Flussonic Media Server

Technical Documentation

Author

July 1, 2025

Contents

0.0.1 Документация Flussonic

Воспользуйтесь нашей документацией, руководствами и справочными материалами, чтобы найти необходимую информацию по использованию и настройке наших продуктов для вашего бизнеса.

Flussonic Media Server

Flussonic Media Server — это многофункциональное программное обеспечение для запуска высоконагруженных видеостриминовых сервисов любого масштаба. Flussonic Media Server может принимать, хранить, транскодировать и доставлять видео до любого количества пользователей и на любые устройства. Flussonic Media Server доставляет видео миллионам зрителей в более чем 100 стран по всему миру. Это комплексное решение для вашего бизнеса.

Более подробно о возможностях, настройке и работе с Flussonic Media Server см. документацию.

Начало работы с Flussonic Media Server Начните работу с Flussonic Media Server с изучения нашего руководства.

Flussonic Media Server API Reference Используйте наш справочник Flussonic Media Server API для интеграции Flussonic Media Server с вашим решением и взаимодействия между ними.

Streaming API Reference Используйте наш справочник Streaming API, чтобы создать свой собственный плеер или другое приложение с использованием всех возможностей проигрывания во Flussonic Media Server.

Flussonic Watcher

Flussonic Watcher — готовое, масштабируемое, веб и мобайл-ориентированное решение для организации онлайн-доступа и записи IP-видеокамер для интеграторов, провайдеров, операторов, различных бизнесов, государственных структур и конечных пользователей.

Более подробно о возможностях Flussonic Watcher см. документацию.

Начало работы с Flussonic Watcher Начните работу с Flussonic Watcher с изучения краткого руководства.

API Flussonic Watcher Flussonic Watcher предоставляет следующие API:

- Watcher Client API для доступа к контенту, в том числе видео с камер, эпизодам аналитики, мозаикам и пр. с учетом системы пользовательских прав.
- Watcher Admin API для управления конфигурацией, пользователями, правами.

Flussonic Central

Flussonic Central — это продукт, позволяющий работать с потоками, Агентами, аналитикой, ONVIF, PTZ на нескольких серверах с *Flussonic Media Server* через единую точку доступа.

API Flussonic Central API Flussonic Central позволяет управлять потоками, Агентами, аналитикой, ONVIF, PTZ на нескольких серверах с *Flussonic Media Server* через единую точку доступа.

Что нового?

Чтобы оставаться в курсе изменений и нововведений в Flussonic Media Server и Flussonic Watcher, посетите наш блог.

Part I

Manuals & Guides

Chapter 1

Manuals & Guides

1.0.1 Быстрый старт Media Server

Эта статья познакомит вас с Flussonic Media Server. Прочитав ее, вы сумеете:

- Установить Flussonic Media Server
- Настроить и посмотреть поток
- Опубликовать видео на сервере
- Загрузить и проиграть файл
- Установка на облачный сервер Selectel

> Далее в документации мы будем приводить IP адрес и URL Flussonic Media Server (например, FLUSSONIC-IP). Вам необходимо заменять их на реальные IP адрес или URL вашего сервера.

Flussonic Media Server

Flussonic Media Server — это серверное программное обеспечение для видео стриминга, способное решать широкий ряд задач от захвата, транскодирования, записи архива и мультипротокольной раздачи видеоконтента (live и on-demand) по всему миру, до управления потреблением контента и видео потоками.

Мы продемонстрируем основные сценарии с помощью веб-интерфейса Flussonic. Однако если вы предпочитаете использовать API, смотрите наш справочник Flussonic API.

Установка Flussonic Media Server

Установка Здесь приводится краткое описание установки, достаточное для быстрого старта Flussonic.

Чтобы попробовать Flussonic Media Server, нужен компьютер с Linux, подключенный к Интернету, и лицензионный ключ. Напишите нашим менеджерам, чтобы приобрести лицензию.

Основное требование — 64-разрядная операционная система. Мы рекомендуем операционную систему Ubuntu Server. Полный список системных требований см. здесь.

> Несмотря на то, что Flussonic Media Server будет работать и на Ubuntu Desktop, мы не рекомендуем ее к использованию, поскольку в Ubuntu Desktop присутствуют особенности с управлением, питанием и энергосбережением, имеется свой Network-manager и фоновые обновления, а также прочие отличия, которые могут сказаться на производительности. Также возможно, что некоторые сторонние ПО и драйвера на ней могут не работать.

Если подходящей системы или свободного сервера под рукой нет, то можно арендовать сервер в Selectel на время, чтобы попробовать Flussonic Media Server. Как это сделать, мы рассказали в подробной инструкции.

В результате вам надо иметь доступ к консоли Linux под пользователем root.

Чтобы установить Flussonic, выполните в командной строке Linux команду:

```
curl -sSf https://flussonic.com/public/install-ru.sh | sh
```

Затем запустите Flussonic Media Server:

Теперь откройте в браузере веб-интерфейс администратора.

Первое открытие веб-интерфейса Flussonic (UI) Веб-интерфейс Flussonic доступен по адресу `http://FLUSSONIC-IP:80/` (замените `FLUSSONIC-IP` на адрес вашего сервера).

На стартовой странице `http://FLUSSONIC-IP:80/` Flussonic просит ввести логин, пароль администратора Flussonic и полученный лицензионный ключ.

Для активации с помощью лицензионного ключа и непрерывного использования *Flussonic Media Server* необходим постоянный доступ в Интернет. Подробнее см. на странице Использование лицензионного ключа.

Логин и пароль не должны содержать символов @, ;, #, [,
, /, =, $

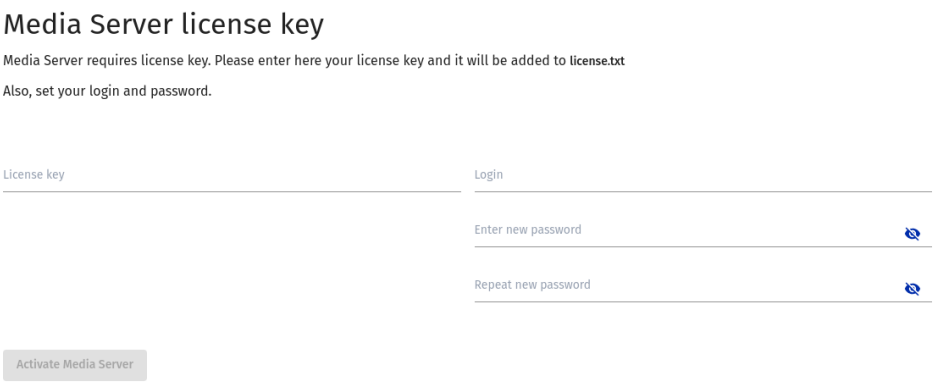


Figure 1.1: Flussonic UI

Проверка установки Проверить правильность установки Flussonic Media Server можно по адресу `http://FLUSSONIC-IP:80/`, где `FLUSSONIC-IP` — адрес того сервера, на который вы поставили ПО. Откроется главная страница веб-интерфейса к Flussonic.

Если веб-интерфейс не открылся, пожалуйста, смотрите более подробное описание установки в разделе Установка или свяжитесь с технической поддержкой Flussonic.

Читайте также:

- Более подробно об установке Flussonic Media Server читайте в разделе про установку.

Получение потокового видео

Flussonic Media Server может получать потоковое видео двумя основными способами: выступая в роли клиента или сервера.

В первом случае Flussonic Media Server сам обращается к источнику для получения с него видео (*захватывает* поток). Во втором – ожидает подключения, чтобы принять видео для *публикации*.

Захват потока Источником видео может быть видеокамера, другой видеостриминговый сервер, специализированная программа, работающая с DVB-картой, и вообще любая программа, умеющая передавать видео по сети. Flussonic поддерживает все основные протоколы передачи видео.

Также Flussonic Media Server может сам генерировать поток `fake://fake`, который можно использовать, например, для проверки работы сервера.

Чтобы добавить поток, перейдите в раздел **Media** > нажмите **Add stream**. Укажите имя потока (`demo`) и URL-адрес источника (`fake://fake`). Нажмите **Create**.

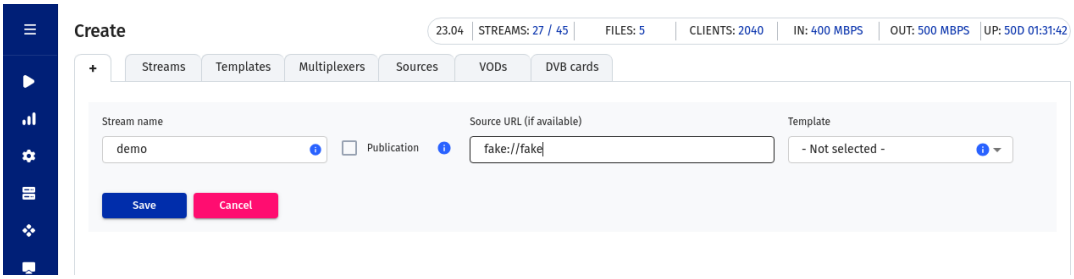


Figure 1.2: Creating live stream

Увидеть результат можно, открыв в браузере страницу `http://FLUSSONIC-IP:80/demo/embed.html`.

Читайте также:

- Подробнее о ссылках для проигрывания см. здесь.
- Подробнее о прямом эфире см. в разделе о потоковом вещании.

Прием публикации Публикацией называется ситуация, когда к Flussonic Media Server подключается другая программа и инициирует передачу ему потокового видео. Чтобы это было возможно, в Flussonic Media Server должно быть сконфигурировано место на сервере, в которое разрешена публикация.

Место публикации может иметь *статическое или динамическое имя*:

- **Статическое** имя используется, если у вас один поток из одного источника, и публикация идет более или менее постоянно. В случае со статическим именем, достаточно указать специальную опцию `publish://` в качестве URL-адреса источника при создании потока во Flussonic. В источнике публикации укажите одну из ссылок **Publish links** со вкладки **Overview** в профиле созданного потока.
- **Динамическое** имя понадобится, если у вас много постоянно меняющихся источников публикации, и вы заранее не знаете сколько и каких потоков нужно будет принять. Для публикации в поток с динамическим именем вам нужно настроить *шаблон (template)* с *префиксом (prefix)* для публикации. В одно место публикации можно будет опубликовать несколько потоков. Префикс будет использоваться для формирования имени потока. Общая схема имени потока такая: `http://FLUSSONIC-IP:80/PREFIX/STREAM_NAME`, причем `STREAM_NAME` задается во внешнем приложении.

Настроим публикацию с динамическим именем:

- Чтобы создать шаблон, перейдите в раздел **Media** > **Templates** > нажмите **Add template**. Укажите имя шаблона (например, `live-mylive`) и специальную опцию `publish://` в качестве URL-адреса источника. Нажмите **Create**.
- Затем нажмите имя созданного шаблона и в разделе **Template settings** укажите префикс (`mylive`). Нажмите **Save and apply to streams**.
- Задайте URL потока в источнике публикации (внешнем приложении). Если вы указали в конфигурации префикс публикации `mylive`, то при настройке источника публикации в URL вы должны указывать имя потока, начинающееся с `mylive/`, например, `mylive/bunny`.

```
/opt/flussonic/bin/ffmpeg -re -i /opt/flussonic/priv/bunny.mp4 -c copy -f
   flv rtmp://FLUSSONIC-IP:1935/mylive/bunny
```

Начнется публикация. На вкладке **Media** появится поток для публикации, который был автоматически сгенерирован из шаблона:

Чтобы посмотреть поток, откройте в браузере адрес:

```
http://FLUSSONIC-IP:80/mylive/bunny/embed.html
```

Читайте также:

- Все о публикации на Flussonic см. в разделе Публикация видео.

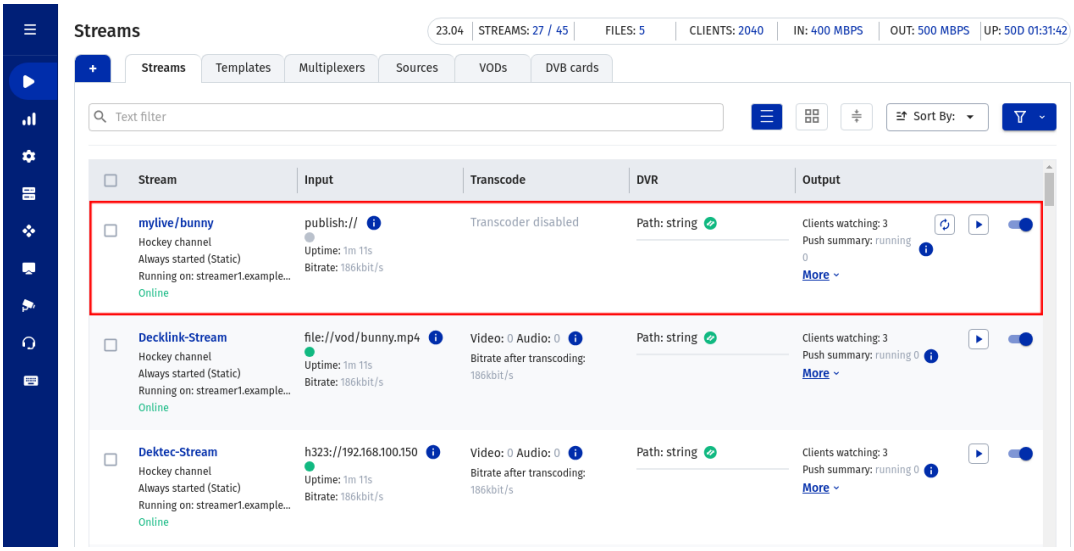


Figure 1.3: Published template stream

Проигрывание файлов

В этом разделе мы научимся проигрывать файлы с помощью Flussonic. Для проигрывания файлов Flussonic использует службу *VOD (Video On Demand)* — неотъемлемую часть услуг, связанных с передачей видео. Чтобы проиграть файл, вам необходимо:

- Создать *VOD-локацию*, чтобы Flussonic Media Server знал, какой путь в запросах на проигрывание файла будет соответствовать файлу на диске или в HTTP хранилище. Чтобы добавить VOD-локацию, перейдите в раздел **Media** > **VODs** > нажмите **Add VOD** > введите **VOD name** (например, `Movies`) и **File directory path** (`/storage`) > нажмите **Create**.
- Теперь можно добавить файл в каталог `/storage`. Перейдите в раздел **Media** > **VODs** > нажмите имя созданной VOD-локации (`Movies`) > **browse** > **Upload Files** > выберите файл для загрузки (`bunny.mp4`).
- Проверить, как проигрывается добавленный файл, можно на странице `http://FLUSSONIC-IP:80/mo`

Чтобы посмотреть все остальные доступные ссылки для проигрывания файла, перейдите в раздел **Media** > **VODs** > нажмите имя созданной VOD-локации (`Movies`) > **browse** > имя файла. Вы увидите встроенный плеер для проигрывания файла, код HTML для использования в плеере на вашем сайте или в приложении и список ссылок для проигрывания файла по разным протоколам.

Читайте также:

- Подробнее о файлах см. в разделе про работу с видеофайлами

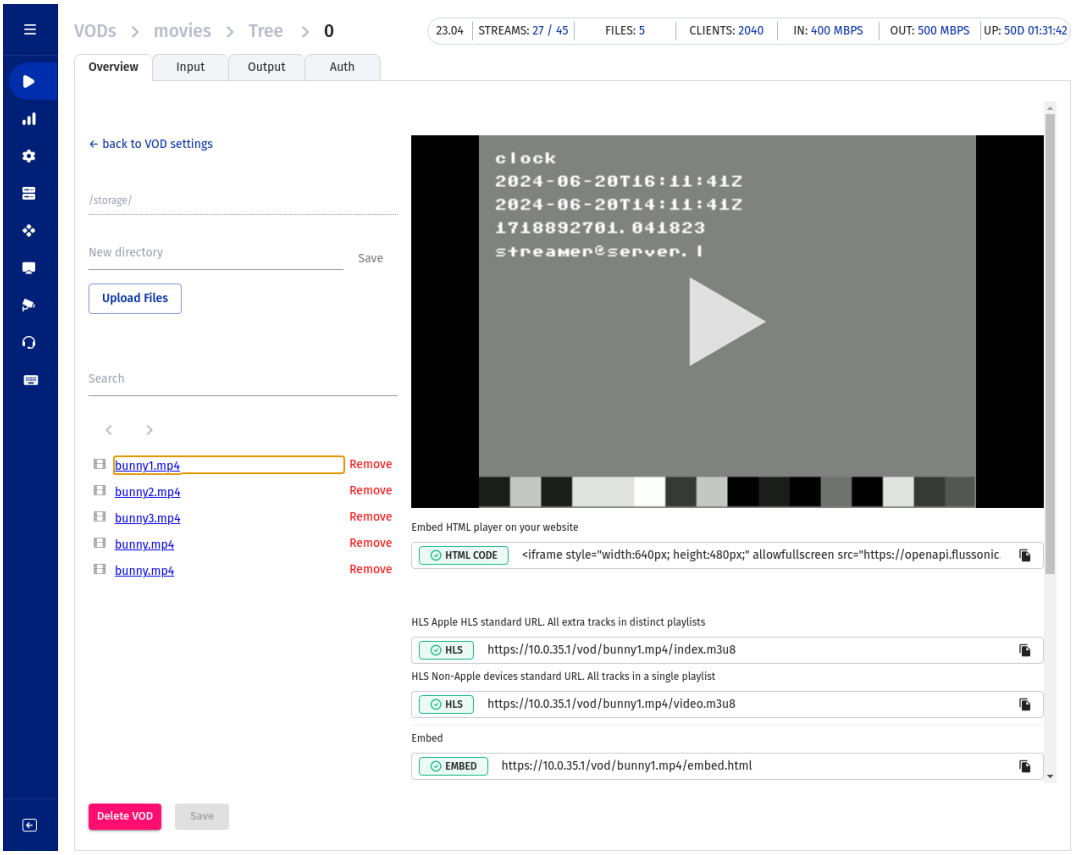


Figure 1.4: Play file

Установка Flussonic Media Server на облачный сервер Selectel

В этом разделе мы расскажем, как создать облачный сервер на платформе Selectel и запустить на нем Flussonic.

Это займет у вас около 10 минут, даже если вы впервые работаете в консоли компьютера.

Создание облачного сервера Создайте аккаунт на сайте Selectel.

В личном кабинете в меню слева выберите **«Облачная платформа»** — **«Серверы»** — **«Создать сервер»**.

Задайте конфигурацию сервера. Для тестирования возможностей Flussonic мы рекомендуем следующие настройки:

* Ubuntu 24.04 LTS 64-bit, * фиксированная конфигурация, Standart, * 2 vCPU, 4 ГБ RAM, * универсальный SSD-диск размером 5 ГБ.

Остальные поля можно оставить по умолчанию.

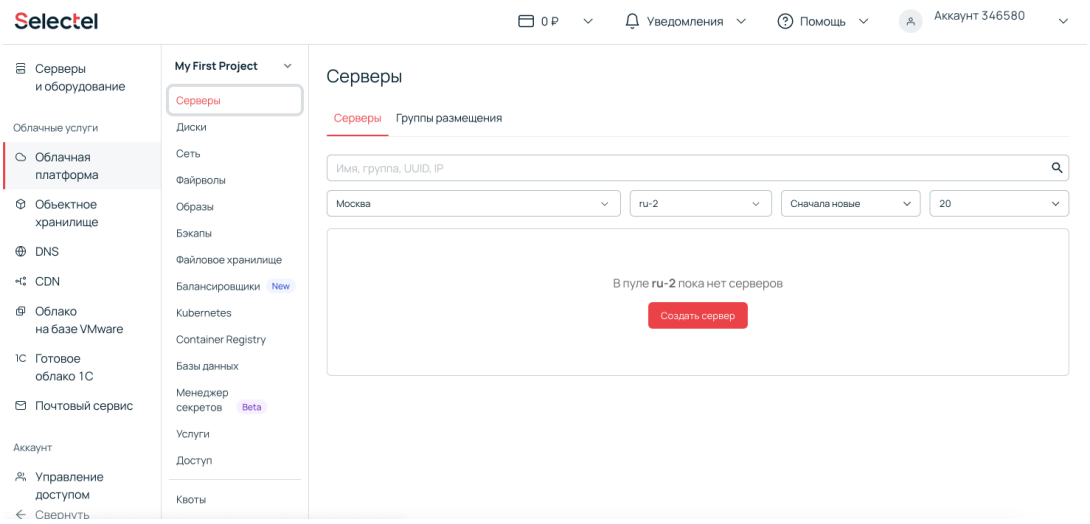


Figure 1.5: Create server

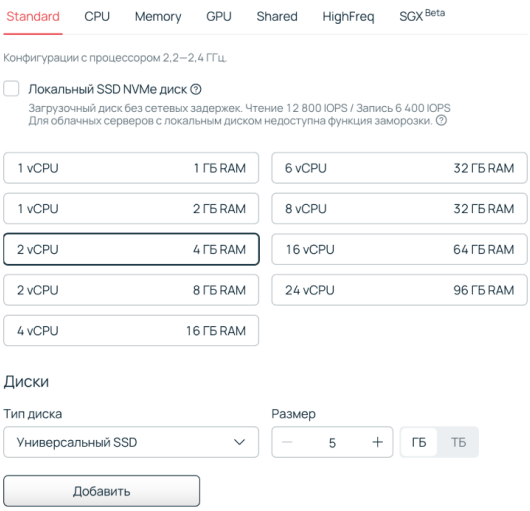


Figure 1.6: Server config

Добавление SSH-ключа При создании сервера вам также нужно добавить SSH-ключ. Если вы уже создавали его ранее, скопируйте и вставьте во всплывающее окно публичный ключ.

Если ключа нет, нужно его создать. Откройте консоль своего компьютера («Терминал» для MacOS или cmd.exe для Windows).

Введите в командной строке:

```
ssh-keygen -t
```

Где <key_type> — тип SSH-ключа: ed25519, rsa, ecdsa или dsa. Например, для rsa:

```
ssh-keygen -t rsa
```

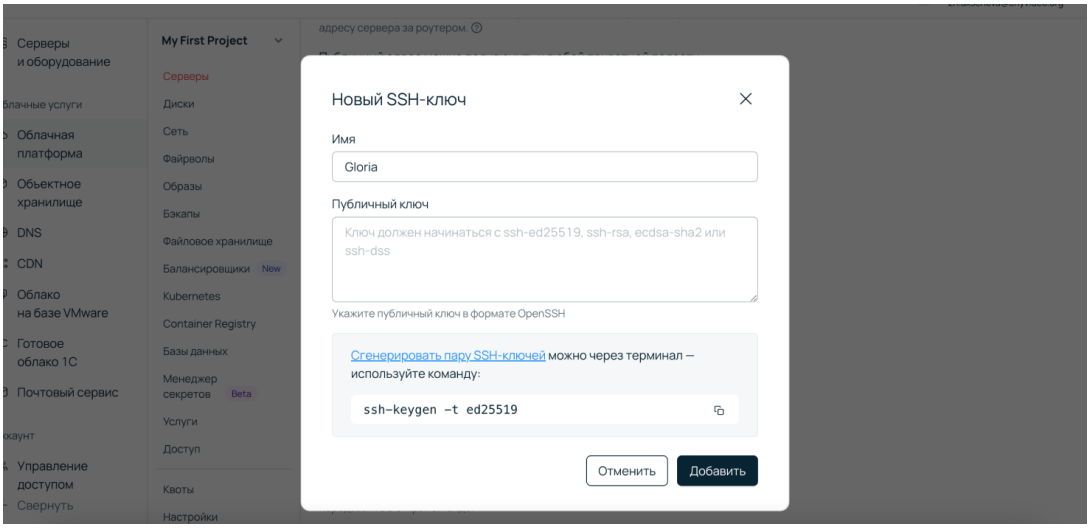


Figure 1.7: SSH

Вы увидите сообщение о выборе директории для хранения пары ключей. Пример для rsa-ключа:

```
Enter file in which to save the key (~/.ssh/id_rsa):
```

Чтобы оставить директорию для хранения ключей по умолчанию, нажмите Enter. Если вы хотите выбрать другую директорию, введите ее в формате `/path/to/id_rsa` и нажмите Enter. Например,

```
/Users/IvanPupkin/.ssh/id_rsa
```

Дождитесь сообщения о том, что ключи сгенерированы.

Будет создано два файла: `id_rsa` (приватный ключ) и `id_rsa.pub` (публичный ключ). В терминале появится отпечаток (fingerprint) ключа и его изображение:

```
Your identification has been saved in ~/.ssh/id_rsa
Your public key has been saved in ~/.ssh/id_rsa.pub
The key fingerprint is:
The key's randomart image is:
```

Теперь вам нужно узнать публичный SSH-ключ, который просит у вас Selectel. Введите команду

```
cat <path>
```

где `<path>` — путь до публичного ключа. В нашем примере это `~/.ssh/id_rsa.pub`.

```
cat ~/.ssh/id_rsa.pub
```

Вы увидите ключ. Скопируйте его от начала до конца и добавьте его для сервера Selectel.

Далее вы увидите стоимость тарифа для сервера с выбранной конфигурацией. Можно посмотреть как цену за час, так и стоимость аренды сервера на месяц. Selectel списывает деньги только

за фактическое использование: если вы создадите сервер, протестируете работу Flussonic в течение двух часов, а потом удалите сервер, деньги спишут только за два часа.

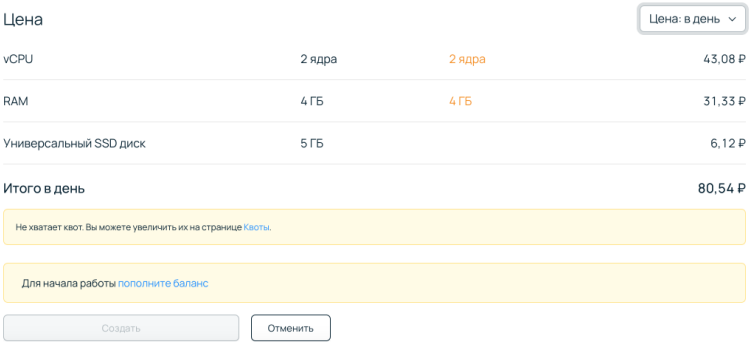


Figure 1.8: Server prices

Пополните кошелек на нужную сумму и создайте сервер.

Скачивание и запуск Flussonic Теперь в разделе **«Серверы»** вы видите новый сервер и его IP-адрес.

Откройте консоль сервера, нажав на кнопку «Открыть консоль» справа от IP-адреса.

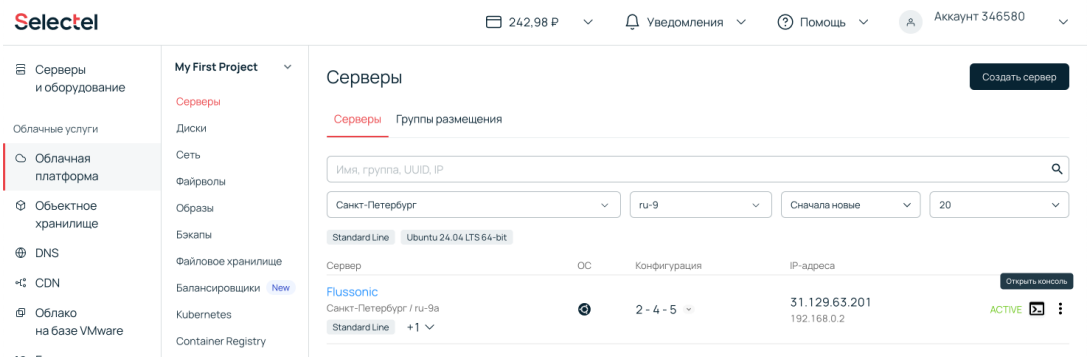


Figure 1.9: Get access

Залогиньтесь под пользователем root. Логин и пароль указаны прямо над консолью.

Введите логин root. Скопируйте и вставьте пароль (он не отобразится, поэтому просто нажмите Enter после вставки).

Далее выполните команду:

```
curl -sSf https://flussonic.com/public/install.sh | sh
```

Начнется загрузка Flussonic.

После появления сообщения `Start Flussonic` введите команду:

```
service flussonic start
```

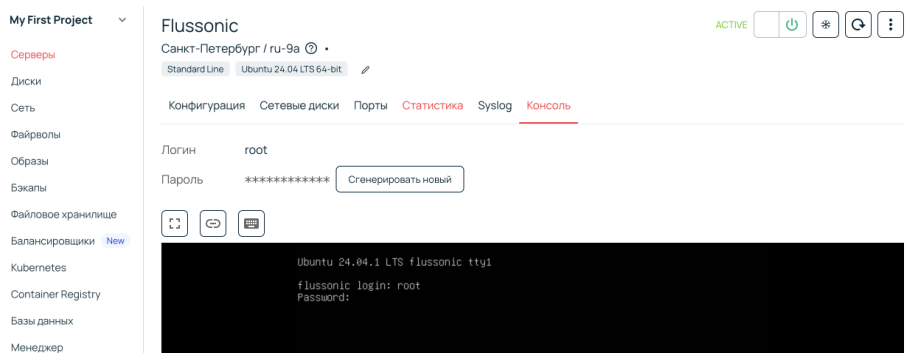


Figure 1.10: Login

Сервер запущен.

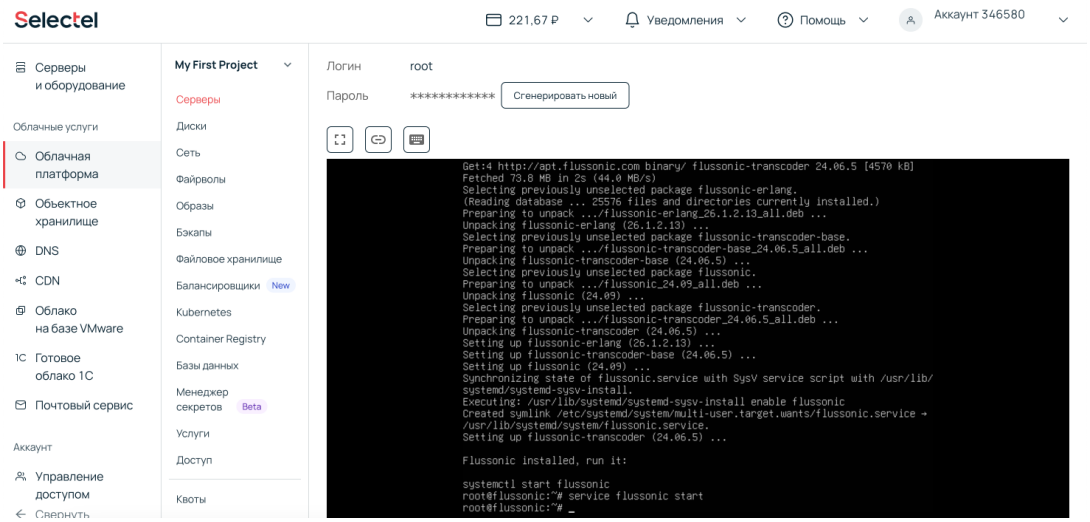


Figure 1.11: Installation

Активация лицензионного ключа Откройте в своем браузере веб-интерфейс Flussonic по адресу `http://FLUSSONIC-IP:80/`, где вместо `FLUSSONIC-IP` необходимо указать IP-адрес вашего сервера.

На открывшейся странице Flussonic просит ввести логин, задать пароль администратора и ввести лицензионный ключ, который вы получили в личном кабинете.

Для запуска первых стримов переходите к настройке и просмотру потоков.

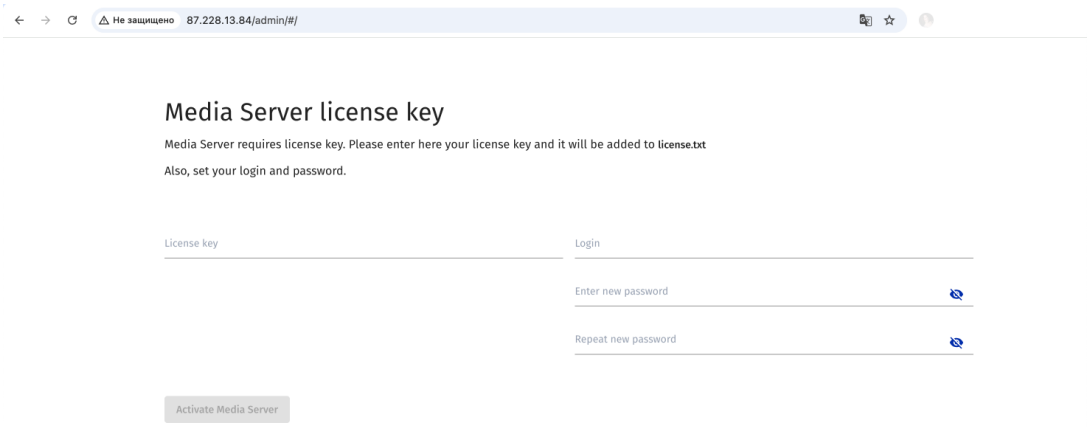


Figure 1.12: License

Chapter 2

Administrator

2.1 Starting

2.1.1 Системные требования

В таблице ниже приведены минимальные системные требования к конфигурации сервера для работы *Flussonic Media Server*. Реальные требования могут различаться в зависимости от количества одновременных подключений, которое будет обслуживать *Flussonic Media Server*.

> Данные рассчеты имеют смысл, если ничего кроме Media Server на этой системе не запущено. Включая виртуализацию.

Минимальные системные требования

Кол-во одновременных подключений	**10**	**100**	**1 000**	
Процессор	Любой	1-ядерный	4-х ядерный (Xeon/Core i7)	2-х проц
Оперативная память	128 Мб	256 Мб	1024 Мб	
Место на жестком диске	40 Мб	40 Мб	40 Мб	
Сетевой адаптер	100 Мбит/с	1 Гбит/с	1 Гбит/с серверный	10
Операционная система	Ubuntu Linux			

Для стабильного воспроизведения видеопотока у клиентов при большом количестве одновременных подключений рекомендуется организовать распределение сетевого трафика по нескольким физическим серверам. Подробная информация о кластеризации серверов *Flussonic Media Server* находится в разделе о кластеризации.

Необходимо учитывать, что при передаче мультимедиа-данных из файлов на жестком диске основная нагрузка ложится на дисковую подсистему. Соответственно, при планировании конфигурации *Flussonic Media Server* особое внимание должно быть уделено производительности используемых жестких дисков. Подробнее можно прочитать в разделе о вещании файлов.

Если на сервере имеется Firewall, лучше удалить его. Если всё-таки вам по каким-то причинам нужен файрвол, обратитесь в поддержку.

Требование к браузеру

Рекомендуемые браузеры для работы в Flussonic Media Server:

- Mozilla Firefox 70 и выше
- Google Chrome 79 и выше

Не рекомендуемые (работоспособность обеспечивается, но возможны ограничения):

- Microsoft Edge 80 и выше

- Safari 13 и выше

Также, убедитесь, что браузер поддерживается его производителем (т.е. жизненный цикл продукта еще не завершен).

2.1.2 Установка Flussonic Media Server

Узнайте, как установить, активировать и запустить Flussonic Media Server.

На этой странице:

Перед установкой Flussonic Media Server

Убедитесь, что удовлетворены следующие условия:

- Ваша система соответствует рекомендуемым системным требованиям.
- У вас открыт HTTP-порт 80 и он не прослушивается никаким другим приложением в вашей ОС. По умолчанию Flussonic Media Server использует HTTP-порт 80.

Если все условия выполнены, то можно переходить к установке.

Установка Flussonic Media Server

Вы можете установить Flussonic Media Server на Ubuntu, CentOS/RedHat и другие RPM-системы.

На Ubuntu **Поддерживаемые архитектуры:** amd64, arm64.

Поддерживаемые версии ОС: Ubuntu LTS 24.04, 22.04.

Установите Flussonic Media Server с помощью утилиты `apt`:

```
curl -sSf https://flussonic.com/public/install-ru.sh | sh
```

После завершения установки, активируйте Flussonic Media Server.

Мы **не** рекомендуем использовать RPM-дистрибутивы
Мы не помогаем с проблемами с RPM-пакетами и RPM-
у которых менее 10 лицензий.

На RPM-системах CentOS/RedHat и подобных им

Установите Flussonic Media Server из Yum-репозитория с помощью следующей команды в терминале:

```
cat > /etc/yum.repos.d/Flussonic.repo <<EOF
[flussonic]
name=Flussonic
baseurl=http://apt.flussonic.ru/rpm
enabled=1
gpgcheck=0
EOF
yum -y install flussonic-erlang flussonic flussonic-transcoder
service flussonic start
```

После завершения установки, активируйте Flussonic Media Server.

Активация Flussonic Media Server

Чтобы активировать Flussonic Media Server:

1) Запустите Flussonic Media Server, используя команду ниже:

```
service flussonic start
```

2) Откройте веб-интерфейс Flussonic Media Server, прописав в адресной строке браузера URL `http://FLUSSONIC-IP:80/`, где `FLUSSONIC-IP` — IP-адрес вашего сервера с Flussonic.

3) Введите полученный лицензионный ключ, а также логин и пароль администратора для управления Flussonic Media Server, которые вы будете использовать. Вы можете найти лицензионный ключ в личном кабинете на my.flussonic.com.

Логин и пароль не должны содержать символов @, ;, #, [,
, /, =, $

Подробнее о лицензировании Flussonic см. на странице Использование лицензионного ключа.

4) Проверьте успешность установки Flussonic Media Server, выполнив следующую команду:

```
service flussonic status
```

Flussonic Media Server готов к работе.

Для большого количества клиентов сделайте тюнинг ОС.

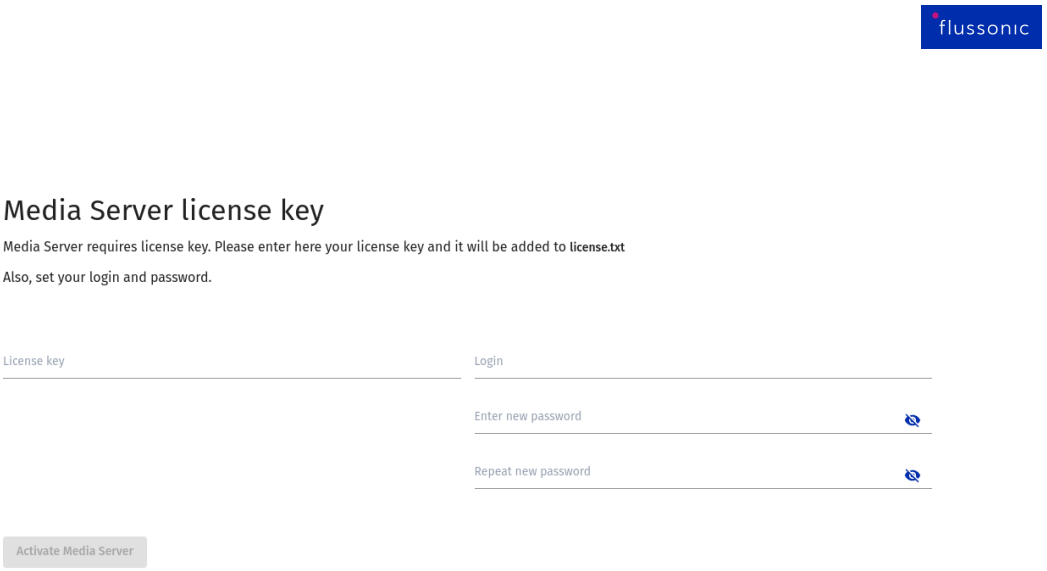


Figure 2.1: Стартовая страница Flussonic

Отключите swap, так как его наличие несовместимо с видеостримингом. Если на сервере не хватает оперативной памяти, её **нельзя** расширять с помощью `swap`.

О конфигурационном файле При первом запуске веб-интерфейса и сохранении введенных логина, пароля и лицензионного ключа автоматически создается конфигурационный файл. Он содержит настройки по умолчанию, такие как путь к базе **Pulse** и лог сессий.

Если у вас есть опыт использования Flussonic, вы можете подготовить этот файл вручную: пропишите логин и пароль в файле и скопируйте его на сервер сразу после установки.

Как изменить пароль администратора?

Изменить пароль администратора можно через редактирование конфигурационного файла либо в веб-интерфейсе.

Редактирование конфигурационного файла Чтобы изменить пароль администратора через конфигурационнный файл:

1) Откройте конфигурационный файл `/etc/flussonic/flussonic.conf` и измените пароль в значении директивы `edit_auth`. 2) Чтобы применить изменения, перечитайте настройки сервера вручную, выполнив команду:

```
service flussonic reload
```

Через веб-интерфейс Чтобы изменить пароль через веб-интерфейс:

1) Перейдите на страницу **Config** в боковом меню. 2) Перейдите на вкладку **Settings** и найдите раздел **Access**. Введите новый пароль в поле *Admin UI password* и повторите пароль в поле ниже. Затем кликните **Save**, чтобы применить изменения.

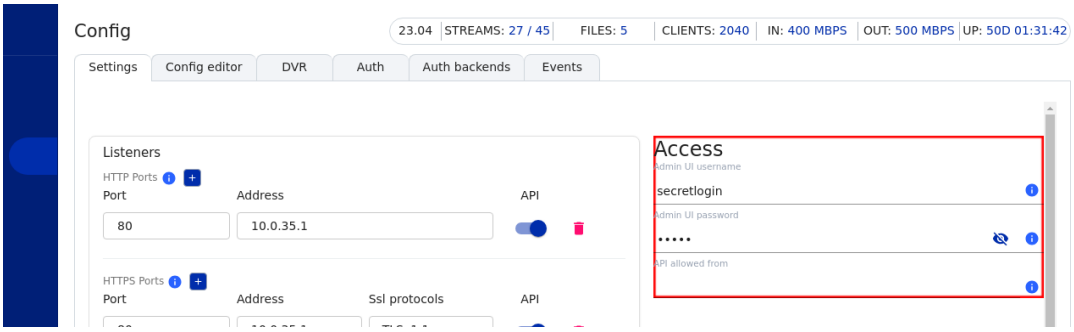


Figure 2.2: Смена пароля администратора

Запуск и остановка Flussonic Media Server

Для управления Flussonic в терминале, используйте следующие команды:

- запустить сервис:

```
service flussonic start
```

- остановить сервис:

```
service flussonic stop
```

- перезапустить сервис:

```
service flussonic restart
```

- переконфигурировать сервис без отключения клиентов:

```
service flussonic reload
```

Запуск Flussonic в Docker контейнере

Flussonic Media Server доступен в качестве Docker контейнера.

Установка в Docker позволит запускать Flussonic в разных операционных системах, если они поддерживают Docker, а не только в Ubuntu. Также такая установка позволит вам использовать все преимущества Docker: изоляцию, безопасность, управление контейнерами, и т.д.

Есть два режима в котором можно использовать Flussonic в докере. Назову их **sysadmin-way** и **devops-way**.

sysadmin-way При такой конфигурации docker используется как аналог виртуалки. Такой режим мы рекомендуем только для тестирования и экспериментов, для небольших сервисов и когда используете только TCP/HTTP протоколы.

Чтобы запустить Flussonic в контейнере:

```
docker run -p 8081:80 -v /some/path/flussonic1:/etc/flussonic flussonic/
   flussonic
```

Или, если у вас есть видеокарты Nvidia, которые вы хотите использовать:

```
docker run --rm -p 8081:80 -v /etc/flussonic:/etc/flussonic --gpus all -e
   NVIDIA_DRIVER_CAPABILITIES='all' flussonic/flussonic
```

Каждому контейнеру можно дать свой путь для конфигурации, замапить на удобный порт. Можно зайти в UI по порту, указанному в строке запуска, создавать и редактировать стримы.

UDP захват, QSV, WebRTC либо не будут работать, либо будут работать с ограничениями из-за особенностей docker.

devops-way В таком режиме инфраструктурные параметры передают через переменные окружения, а конфигурация потоков либо через `ro`-директорию, либо через config-external.

- `STREAMER_HTTP` - позволяет задать порт, который будет слушать контейнер.
- `STREAMER_EDIT_AUTH` - задает логин-пароль для доступа к API и UI.
- `STREAMER_CONFIG_EXTERNAL` - адрес бэкенда конфигурации стримов.
- `LICENSE_KEY` - лицензионный ключ.

Монтировать необходимо директорию `/etc/flussonic/flussonic.conf.d`, Flussonic из нее прочитает все файлы.

В таком режиме через API настраивать сервер нельзя, но можно зайти в UI для проверки.

Как изменить часовой пояс на сервере

Если требуется изменить часовой пояс на сервере, то это можно сделать штатными средствами операционной системы. Смена часового пояса на сервере не влияет на работу Flussonic Media Server, так как все операции внутри проводятся в часовом поясе UTC, включая ведение записей в журналах. Мы рекомендуем обязательно синхронизировать время вашего сервера по NTP.

2.1.3 TLS сертификат с Let's Encrypt

Сервис Let's Encrypt позволяет получать сертификаты для настройки HTTPS в автоматическом режиме.

Flussonic Media Server имеет встроенную поддержку Let's Encrypt, поэтому установки дополнительных пакетов и ручной настройки веб-сервера не требуется.

Достаточно зайти в веб-интерфейс администратора и нажать на кнопку **Issue by Let's Encrypt**.

После этого *Flussonic Media Server* автоматически получит и установит сертификат, а вам нужно будет указать номер порта HTTPS.

Вам не нужно беспокоиться о сроке жизни сертификата и редактировать конфигурацию.

HTTPS нужен, чтобы:

- никто не мог перехватить управление сервером, узнать ваш пароль или ссылки на потоки;
- защитить видео с камер наблюдения;
- вставить ссылку на сайт, работающий по https (иначе браузеры будут выдавать сообщения о незащищенном контенте).

Ниже вы найдете более подробное описание процесса настройки и механизма работы сервиса Let's Encrypt.

Let's Encrypt: как это работает

Подробное описание на официальном сайте: https://letsencrypt.org/how-it-works/.

Чтобы Let's Encrypt выдал вам валидный сертификат, нужно доказать, что вы владеете доменом. Когда вы нажимаете **Issue by Let's Encrypt** в панели администратора, *Flussonic Media Server* сообщает доменное имя, для которого требуется сертификат. В ответ он получает ключ, который нужно будет отдать, когда проверяющий бот обратится к серверу по HTTP (именно на 80-й порт) по адресу `http://your-domain.com/.well-known`.

Проверяющий бот обращается к вашему домену, поэтому домен должен быть делегирован, а DNS записи настроены на IP-адрес, где работает *Flussonic Media Server*. Бот подтверждает владение доменом, а *Flussonic Media Server* сохраняет сертификат.

Чтобы продлить сертификат, придется повторить проверку, а значит *Flussonic Media Server* всегда должен слушать порт `http 80;`. Перенести проверку на другой порт не получится — такие правила у Let's Encrypt. Продление происходит автоматически, когда срок действия сертификата подходит к концу, но можно обновить сертификат и вручную, через панель администратора *Flussonic Media Server*.

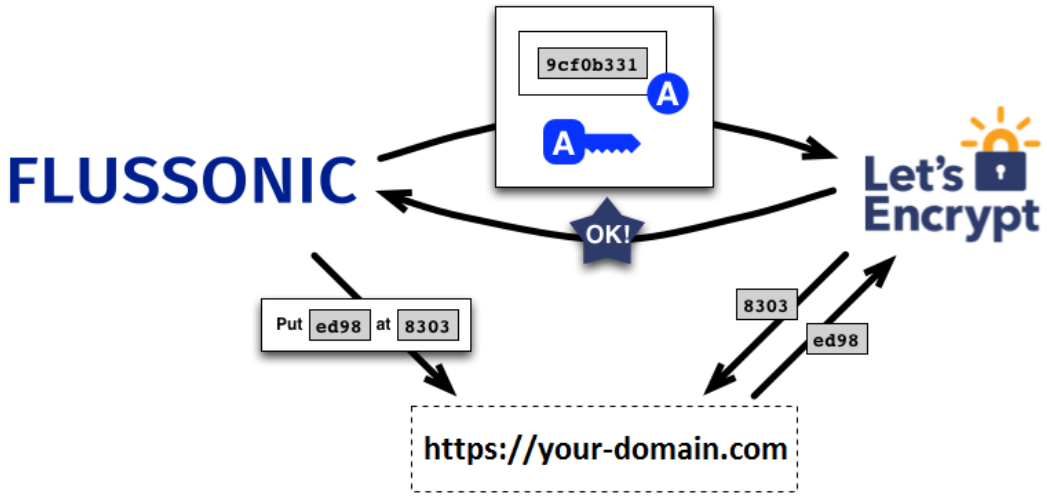


Figure 2.3

Настройка сертификата Let's Encrypt

- Зайдите в панель администратора *Flussonic Media Server* с помощью доменного имени, а не IP-адреса (например, `http://your-domain.com/admin`).
- Перейдите во вкладку **Config**.
- В разделе **TLS-tunneled protocols** нажмите **Issue by LetsEncrypt**. Эта кнопка запускает процесс получения сертификата.
- Дождитесь, когда появится срок действия сертификата (как правило, это занимает до 10 секунд).
- В разделе **Listeners** добавьте номер порта 443 в список **HTTPS ports**. Подробнее о настройках Listeners читайте здесь

Сохраните настройки, нажав кнопку **Save**. *Flussonic Media Server* перенаправит ваш браузер на https:// — теперь можно предоставлять услуги по HTTPS.

Получение сертификата Let's Encrypt для нескольких доменов

Описанная выше процедура позволяет выпустить SSL-сертификат только для одного домена. Но что если вы используете несколько установок Flussonic (например, для доставки нескольких телевизионных каналов), и вам нужно защитить SSL-сертификатом нескольких доменов?

В этом случае воспользуйтесь нашей CLI утилитой Let's Encrypt, которая позволяет получить сертификат для нескольких доменов. Например, если у вас есть домены `domain1.example.com` и `domain2.example.com`, на которых установлен Flussonic, запустите следующую команду:

```
/opt/flussonic/contrib/control.erl letsencrypt -d domain1.example.com -d domain2
```

Figure 2.4

Сертификат Let's Encrypt будет выпущен для обоих доменов.

2.1.4 Использование лицензионного ключа

Flussonic активируется двумя способами:

- Онлайн. Онлайн-лицензия требует постоянного доступа в Интернет на серверах *Flussonic*. Получить бесплатную пробную онлайн-лицензию можно на сайте.
- Офлайн. Если открыть доступ в Интернет на сервере *Flussonic* невозможно, как в государственных объектах, местах с повышенными требованиями к безопасности или без связи с внешним миром, используйте USB-лицензию, защищенную USB-ключом Guardant. USB-лицензия предоставляется только на конкретную версию *Flussonic Media Server*. Для получения USB-ключа или использования других способов оффлайн активации свяжитесь с нашей службой технической поддержки и отделом продаж по адресу support@flussonic.com.

На этой странице:

- Как использовать онлайн лицензионный ключ.
- Как использовать лицензионный USB-ключ.
- Что делать без лицензионного ключа.

Онлайн лицензионный ключ

При первом открытии пользовательского веб-интерфейса *Flussonic* потребует ввести лицензионный ключ, полученный при покупке лицензии или запросе триальной лицензии.

Редактировать ключ можно двумя способами:

- в файле `/etc/flussonic/license.txt`,
- в веб-интерфейсе *Flussonic* в разделе **Config** > **License**.

Ключ выглядит следующим образом:

```
l4|WXHMkfXhFHeNmvDz-M_tb4|r6BzpmVPpjgKpn9IunpFp5lLbCZOp3
```

Чтобы валидировать лицензионный ключ, серверу нужен доступ в Интернет по HTTP и HTTPS.

> Сервер лицензирования не выполняет никаких запросов к серверу *Flussonic*, так что добавлять сервер лицензирования в список исключений при ограничении доступа к серверу *Flussonic* не нужно. Для корректной работы лицензии *Flussonic* разрешите на вашем сервере исходящие подключения для HTTP-портов 80 и 443.

Процесс активации онлайн-лицензии описан в разделе Активация Flussonic Media Server.

Привязка ключа Для привязки ключа к серверу *Flussonic* требуется постоянная связь с сервером лицензий.

Перенос ключа на другой сервер *Flussonic* связывается с сервером лицензий через Интернет. Чтобы перенести лицензионный ключ на другой сервер, выполните следующие шаги:

- Выключите *Flussonic* на первом сервере.
- Включите *Flussonic* на втором сервере и введите лицензионный ключ при первом открытии пользовательского веб-интерфейса.

USB лицензионный ключ

Чтобы активировать USB-ключ, вы получите:

- USB-ключ,
- лицензионный ключ, начинающийся с `g4|`,
- файл активации.

Вы также можете самостоятельно скопировать и скачать их, зайдя в личный кабинет.

Файл активации запускает версию *Flussonic Media Server*, которая указана в названии файла. Если вы хотите запустить несколько разных версий, используйте файлы активации для каждой соответствующей верси.

Чтобы активировать USB-лицензию после установки *Flussonic*, следуйте шагам ниже:

- Не вставляя USB ключ, выполните следующую команду в терминале:
- Вставьте USB-ключ в сервер.
- Запустите *Flussonic*, выполнив следующую команду:

4. Откройте веб-интерфейс *Flussonic Media Server*, по адресу `http://FLUSSONIC-IP:80/`, где `FLUSSONIC-IP` — IP-адрес вашего сервера с *Flussonic*.

- Вы увидите стартовую страницу, на которой нужно ввести лицензионный ключ, начинающийся с `g4|`, а также логин и пароль администратора для управления *Flussonic Media Server*, которые вы будете использовать. Введите данные и нажмите **Activate Media Server**.
- Когда лицензионный ключ пройдет валидацию, вы будете перенаправлены на вкладку **Config > Settings** в веб-интерфейсе администратора *Flussonic*. Нажмите на кнопку **Upload Activation file** и выберите файл активации для версии *Flussonic*, установленной на сервере. Вы можете найти лицензионный ключ и файл активации в личном кабинете на my.flussonic.com.

Веб-интерфейс Flussonic без лицензионного ключа

Если лицензионный ключ не прошел валидацию либо он отсутствует, то веб-интерфейс *Flussonic* откроется в урезанном виде и показывает страницу для ввода лицензионного ключа или загрузки файла активации USB-ключа.

В этом случае доступны только разделы **Config > Settings** и **Support**. Для Coder дополнительно к этим разделам будет доступен раздел **Chassis**.

Перенос ключа на другой сервер

При переустановке операционной системы и продукта Flussonic на этом же сервере или после обновления его железа вы можете повторно использовать свою лицензию на этом же самом железе или после обновления его компонентов, т.к. лицензии никак не привязаны к железу/серверу/ядру/ip адресу и т.д. Достаточно один сервер выключить, другой включить. Надо удалить ключ со старого сервера и остановить там флюссоник.

Мы следим за количеством одновременно работающих серверов.

При смене сервера спокойно запускайте на новом, после чего удаляйте со старого сервера ключ и не забудьте на этом старом сервере выключить сервис флюссоник, чтобы он не перезапускался автоматически. Если забудете, то у вас будет два запущенных сервера, а наши коллеги предложат вам расширить лицензию. Извещать нас о смене сервера не нужно.

2.1.5 Обновление Flussonic

Обновите пакет Flussonic до последней версии, чтобы получить доступ к новым возможностям и исправлениям ошибок более ранних версий. Чтобы избежать проблем с работой сервиса при переходе на последнюю версию Flussonic, обновляйте версию Flussonic каждый месяц или хотя бы раз в три месяца.

О выходе новой версии и изменениях в ней вы можете узнать в веб-интерфейсе Flussonic и блоге.

> Если у вас возникли проблемы с последней версией Flussonic, обратитесь в службу технической поддержки. Если проблему не удалось решить, вы можете вернуться к предыдущей версии.

На этой странице:

Обновление Flussonic через веб-интерфейс

Обновите версию Flussonic в зависимости от типа вашего лицензионного ключа: онлайн-ключ или USB-ключ.

Обновление Flussonic с онлайн-ключом 1) Установите последнюю версию Flussonic, нажав **Upgrade** в боковой панели веб-интерфейса.

2) Завершите установку обновления, перезапустив сервис. Нажмите **Restart** и подождите пока Flussonic перезапустится.

Обновление Flussonic с USB-ключом 1) Скачайте файл активации, зайдя в личный кабинет на my.flussonic.com. Перейдите в **License keys** > **Licenses** > ваша лицензия, выберите нужную версию из выпадающего списка и нажмите **Get Offline Activation File**.

2) Откройте веб-интерфейс Flussonic и перейдите в раздел **Config** > **Settings**. Загрузите файл активации, нажав **Upload Activation File**.

Figure 2.5: Flussonic upgrade in the UI

3) Установите обновление, нажав **Upgrade** в боковой панели веб-интерфейса.

4) Перезапустите сервис. Нажмите **Restart** и подождите пока Flussonic перезапустится.

Обновление Flussonic через командную строку на Ubuntu

```
apt-get update
apt-get -y install flussonic
service flussonic restart
```

Figure 2.6: Activation file in the client area

Чтобы завершить установку обновления, перезапустите Flussonic вручную после установки новой версии. Это сделает третья команда в примере выше.

Обновление Flussonic через командную строку на CentOS

```
yum -y install flussonic flussonic-erlang flussonic-transcoder
```

Менеджер пакетов может создать файл `/etc/init.d/flussonic.rpmnew`. Переименуйте его следующим образом:

```
mv /etc/init.d/flussonic.rpmnew /etc/init.d/flussonic
```

Figure 2.7: Upload activation file in the admin UI

Перезапустите Flussonic после обновления:

```
service flussonic restart
```

Определение версии Flussonic, установленной на компьютере

Чтобы посмотреть текущую версию Flussonic Media Server на вашем компьютере, выполните следующую команду в терминале:

```
dpkg -l | grep flussonic
```

Как откатиться к предыдущей версии

Откатитесь к предыдущей версии Flussonic, имея лицензию с онлайн-ключом или с USB-ключом.

Для онлайн-ключа

Мы не храним версии Flussonic, которые вышли более девяти месяцев назад.

Укажите версию пакета `flussonic` и его зависимостей.

1) Узнайте версии зависимостей, используя `apt-cache` и указав необходимую версию Flussonic:

```
apt-cache show flussonic=24.02  | egrep '^(Depends|Suggests):'
```

Вывод будет примерно следующий:

```
Depends: flussonic-erlang (=26.1.2.3), flussonic-transcoder-base (=23.02.0)
```

2) Установите пакеты с указанием полученных версий:

Перед установкой пакетов сделайте резервную копию конфигурационных файлов из директории `/etc/flussonic` и файлов `.db` из директории `/opt/flussonic/priv` (эта директория используются по умолчанию, в конфигурационном файле вы можете задать произвольный путь).

```
apt-get install flussonic=24.02 flussonic-erlang=26.1.2.3 flussonic-
   transcoder-base=23.02.0
```

Мы не гарантируем работоспособность сервера на дистрибутивах Linux, для которых нет установочных пакетов.

Для USB-ключа

Файлы активации генерируются под конкретную лицензию и версию Flussonic.

Чтобы вернуться к предыдущей версии Flussonic, следуйте инструкции в разделе Обновление Flussonic с USB-ключом.

Промежуточные обновления между релизами

Новая версия Flussonic выходит каждый месяц. У нас есть репозиторий с промежуточными обновлениями. Каждый день он обновляется новой сборкой Flussonic, которая содержит новые функции и исправления ошибок. Промежуточные обновления — это предварительные версии

(release candidate, RC), которые используются в нашей лаборатории. Предварительные версии предлагаются клиентам, которые хотят получить обновления до выхода официального релиза.

Вы можете установить промежуточное обновление и вернуться к официальному релизу.

Установка промежуточного обновления 1) Удалите устаревшую версию Flussonic и её зависимости:

2) Измените используемый репозиторий на репозиторий с промежуточными обновлениями и установите Flussonic:

```
mkdir -p /etc/apt/trusted.gpg.d/
curl -L http://apt.flussonic.ru/repo/master/dev.key > /etc/apt/trusted.gpg.
   d/dev.gpg
echo "deb http://apt.flussonic.ru/branch/flussonic/master repo/" > /etc/apt
   /sources.list.d/flussonic.list;
apt update;
apt install flussonic;
service flussonic restart
```

Возврат к официальному релизу 1) Удалите устаревшую версию Flussonic и ее зависимости:

> Перед установкой пакетов сделайте резервную копию конфигурационных файлов из директории `/etc/flussonic` и файлов `.db` из директории `/opt/flussonic/priv` (эта директория используются по умолчанию, в конфигурационном файле вы можете задать произвольный путь).

```
apt remove flussonic
```

2) Измените используемый репозиторий на репозиторий с официальными релизами и установите Flussonic:

```
echo "deb http://apt.flussonic.ru binary/" > /etc/apt/sources.list.d/
   flussonic.list;
apt update;
apt install flussonic;
service flussonic restart
```

> Если Flussonic не запускается, выполните в терминале команды `service flussonic run` и `journalctl -u flussonic -n 100` и отправьте результат нашей службе технической поддержки.

2.1.6 Обновление Flussonic Coder

Чтобы получать доступ к новым возможностям Flussonic Coder и устранять возможные ошибки в его работе, обновляйте версию прошивки до последней. Если у вас возникли трудности в работе Flussonic Coder после обновления,

]обратитесь в техническую поддержку и откатитесь до рабочей версии.

Проверка на наличие новых версий

- Откройте веб-интерфейс Flussonic Coder и перейдите на страницу **Chassis**.
- В разделе **System Information** нажмите **Check For New Version**.

В выпадающем списке *Firmware Version* вы увидите новые версии прошивки, на которые можете обновиться.

Обновление и откат версии прошивки

- Откройте веб-интерфейс Flussonic Coder и перейдите на страницу **Chassis**.
- (Для обновления) Проверьте наличие новых версий, перейдя в раздел **System Information** и нажав **Check For New Version**.
- В списке *Firmware Version* выберите нужную версию прошивки и нажмите кнопку **Upgrade**. Чтобы обновиться на последнюю версию, нажмите **Upgrade To Latest**.

Вы увидите, что версия прошивки, указанная в *Firmware Version* изменилась. Значит, обновление или откат Flussonic Coder прошли успешно.

2.2 Maintaining

2.2.1 Оптимизация Flussonic Media Server и операционной системы

В этом разделе будут приведены некоторые часто встречающиеся проблемы и возможности детальной настройки ОС и Flussonic Media Server для больших нагрузок.

Настройка UDP захвата

Flussonic автоматически меняет некоторые настройки сети, чтобы оптимизировать захват UDP источников, включая WebRTC и мультикаст. Чтобы отключить эту функцию, вы можете задать переменную окружения DO_NOT_DO_NET_TUNING и выполнить эти настройки вручную:

Чтобы передать переменную окружения DO_NOT_DO_NET_TUNING в Flussonic, выполните команду:

```
systemctl edit flussonic
```

В открывшийся файл добавьте строки:

```
[Service]
Environment="DO_NOT_DO_NET_TUNING=true"
```

Сохраните изменения.

Далее вам нужно увеличить размер памяти под UDP буферы:

```
sysctl -w net.core.rmem_max=1048576
sysctl -w net.core.rmem_default=1048576
sysctl -w net.ipv4.udp_mem="8388608 12582912 16777216"
```

Эти настройки будут работать до перезагрузки. Чтобы сохранить эти параметры навсегда, нужно отредактировать файл `/etc/sysctl.conf`.

В самый конец файла добавить:

```
net.core.rmem_max = 1048576
net.core.rmem_default=1048576
net.ipv4.udp_mem = 8388608 12582912 16777216
```

Чтобы применить изменения из файла, нужно выполнить команду:

```
sudo sysctl -p
```

Работа с большим количеством памяти

При наличии более 60 гигабайт памяти рекомендуется зарезервировать 10 гигабайт под Linux:

```
sysctl vm.min_free_kbytes=10240000
```

Настройка TCP/IP стека

Если вы используете Flussonic Media Server для вещания более чем на 3-4 Гбит/с для вас могут стать ощутимыми тонкости настройки TCP стека в Linux.

Во-первых, требуется увеличить количество доступной памяти для буферов соединений:

```
sysctl -w net.core.wmem_max=16777216
sysctl -w net.ipv4.tcp_wmem="4096 4194394 16777216"
sysctl -w net.ipv4.tcp_congestion_control=htcp
sysctl -w net.ipv4.tcp_slow_start_after_idle=0
```

Эти настройки будут работать до перезагрузки. Чтобы сохранить эти параметры навсегда, нужно отредактировать файл /etc/sysctl.conf, в самый конец файла вставить:

```
net.core.wmem_max = 16777216
net.ipv4.tcp_wmem = 4096 4194394 16777216
```

Чтобы применить изменения из файла, нужно выполнить команду `sudo sysctl -p`.

Так же надо поменять настройки сетевого интерфейса: `ifconfig eth0 txqueuelen 10000`.

Обязательно надо проверить версию драйвера сетевой карты. Желательно использовать самую свежую версию. Выяснить версию драйвера и firmware можно так:

```
ethtool -i eth2
```

```
driver: ixgbe
version: 3.15.1
firmware-version: 0x61c10001
bus-info: 0000:04:00.0
```

> При обновлении файла firmware в каталоге `/lib/firmware` необходимо перезагружать сервер. При этом может остаться старый firmware. Не забудьте запустить утилиту `update-initramfs` перед рестартом сервера.

Настройка сетевой карты

Прерывания сетевой карты Современные 10-гигабитные сетевые адаптеры имеют несколько очередей для входящих и исходящих пакетов, которые иногда приходится вручную привязывать к ядрам процессоров.

Сервер на котором такую оптимизацию не сделали, обрабатывает всю сетевую подсистему на одном ядре. Выглядит это так:

```
cat /proc/interrupts
```

```
           CPU0    CPU1    CPU2    CPU3    CPU4    CPU5    CPU6    CPU7
  0:       2097       0       0       0       0       0       0       0  IR-IO-
   APIC      timer
...
 66: 2072120005       0       0       0       0       0       0       0  IR-PCI-
   MSI      eth2-TxRx-0
 67:    1562779       0       0       0       0       0       0       0  IR-PCI-
   MSI      eth2-TxRx-1
 68:    1830725       0       0       0       0       0       0       0  IR-PCI-
   MSI      eth2-TxRx-2
 69:    1504396       0       0       0       0       0       0       0  IR-PCI-
   MSI      eth2-TxRx-3
 70:    5112538       0       0       0       0       0       0       0  IR-PCI-
   MSI      eth2-TxRx-4
 71:    2229416       0       0       0       0       0       0       0  IR-PCI-
   MSI      eth2-TxRx-5
 72:    1686551       0       0       0       0       0       0       0  IR-PCI-
   MSI      eth2-TxRx-6
 73:    1217916       0       0       0       0       0       0       0  IR-PCI-
   MSI      eth2-TxRx-7
 74:       2358       0       0       0       0       0       0       0  IR-PCI-
   MSI      eth2
```

Для сетевых карт Intel производитель предлагает скрипт `set_irq_affinity` который раскидывает очереди по ядрам. После его запуска статистика прерываний выглядит так:

```
           CPU0       CPU1       CPU2       CPU3       CPU4       CPU5
     CPU6       CPU7
  0:       2097          0          0          0          0          0
         0          0  IR-IO-APIC  timer
...
 66: 2072120005          0          0          0          0          0
         0          0  IR-PCI-MSI  eth2-TxRx-0
 67:    1562779 1162738082          0          0          0          0
         0          0  IR-PCI-MSI  eth2-TxRx-1
 68:    1830725          0 1133908105          0          0          0
         0          0  IR-PCI-MSI  eth2-TxRx-2
 69:    1504396          0     177620 1123678951          0          0
         0          0  IR-PCI-MSI  eth2-TxRx-3
 70:    5112538          0          0          0 1638450740          0
         0          0  IR-PCI-MSI  eth2-TxRx-4
 71:    2229416     130189          0          0          0 1441511712
         0          0  IR-PCI-MSI  eth2-TxRx-5
 72:    1686551          0          0          0          0          0
   1402472725          0  IR-PCI-MSI  eth2-TxRx-6
 73:    1217916          0          0      66145          0          0
         0 1380402032  IR-PCI-MSI  eth2-TxRx-7
 74:       2358          0          0          0          0          0
         0          0  IR-PCI-MSI  eth2
```

Эта настройка становится критичной примерно в районе 3-5 Гбит/с трафика.

Соединение со свичем При соединении сетевой карты сервера со свичем, проверьте, что с обоих сторон установлены совместимые настройки. Т.е. с обеих сторон должно быть либо *auto select*, либо строго одинаковая скорость и дуплекс.

Оптимизация сервера для VOD вещания

Отдельный, более детальный раздел посвящен оптимизации сервера для вещания фильмов.

Перевод процессора в режим высокой производительности

Для экономии энергии в ОС Linux по умолчанию регулятор `scaling_governor` находится в режиме энергосбережения (powersave). В таком случае сервер не будет использовать все свои аппаратные ресурсы. Чтобы сервер работал в режиме высокой производительности, необходимо:

Отключить регулятор ondemand:

```
systemctl disable ondemand
```

Перезагрузить сервер:

```
reboot
```

Проверить текущее значение `scaling_governor`:

```
cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
```

2.2.2 Миграция

При миграции Flussonic Media Server с сервера на сервер нужно перенести файлы конфигурации и лицензии, а также можно перенести архив.

> Исполняемые файлы и установленные библиотеки переносить **нельзя**. Используйте пакетный менеджер для установки на новом сервере.

Перенос конфигурации и лицензии

Порядок переноса конфигурации:

- Установите Flussonic Media Server на новом сервере.
- Перенесите следующие файлы со старого сервере на новый одним из описанных ниже способов:
- Остановите Flussonic Media Server на старом сервере.
- Запустите на новом.

Способы перенести файлы:

- Перенос конфигурации с помощью веб-интерфейса
- Перенос конфигурации с помощью SCP
- Перенос конфигурации с помощью USB-носителя

С помощью веб-интерфейса Зайдите в веб-интерфейс Flussonic Media Server на страницу **Config** -> **Settings** и на старом, и на новом сервере. Чтобы скачать файл конфигурации, на старом сервере нажмите на кнопку **Download Config** в самом низу страницы. Для загрузки файла конфигурации на новый сервер нажмите на кнопку **Upload Config**.

Ваш лицензионный ключ можно посмотреть в личном кабинете на вкладке **Лицензионные ключи**.

С помощью SCP SCP (Secure CoPy) — программа для удаленного копирования файлов по сети между хостами. Она использует SSH для передачи данных, в том числе аутентификацию и меры безопасности, которые реализованы для SSH.

Для копирования файлов конфигурации и лицензии с одного удаленного сервера `remote.host1` на другой удаленный сервер `remote.host2` необходимо выполнить команду:

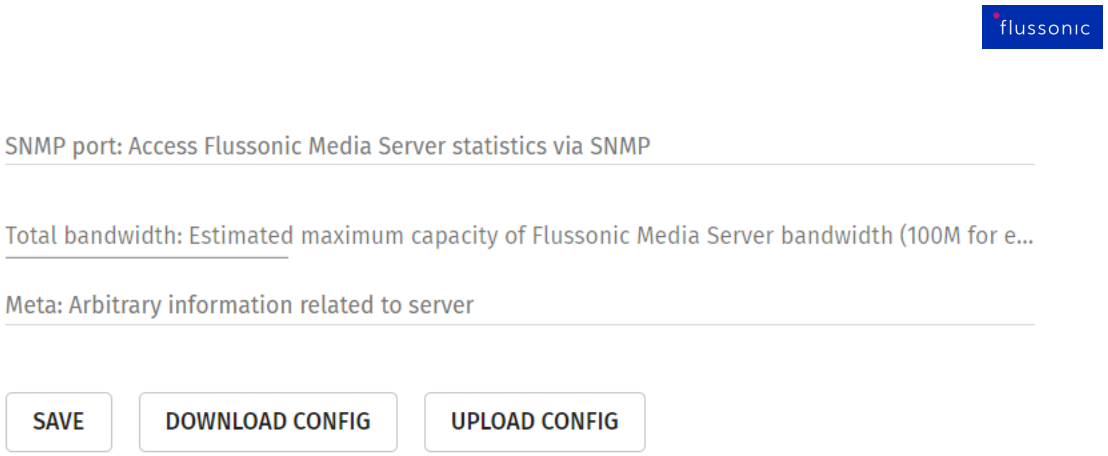


Figure 2.8: flussonic ui

```
scp user@remote.host1:/etc/flussonic/flussonic.conf user@remote.host2:/etc/
   flussonic/
scp user@remote.host1:/etc/flussonic/license.txt user@remote.host2:/etc/
   flussonic/
```

С помощью USB-носителя Если вы хотите перенести файлы конфигурации с помощью какого-либо USB-носителя, то воспользуйтесь следующей инструкцией.

Монтирование USB Создайте директорию, в которую будем монтировать:

```
mkdir -p /mnt/usb
```

Вставьте носитель в USB порт и узнайте имя устройства:

```
fdisk -l
```

Результатом этой команды будет:

```
Disk /dev/sdb: 4008 MB, 4008706048 bytes
118 heads, 53 sectors/track, 1251 cylinders, total 7829504 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x74a37a4d

Device     Boot      Start         End      Blocks   Id  System
/dev/sdb1   *           63     7829503     3914720+   b  W95 FAT32
```

Здесь имя устройства: `/dev/sdb1`.

Смонтируйте носитель:

```
mount /dev/sdb1 /mnt/usb
```

Проверьте подключение:

```
mount
```

Копирование конфигурации

```
cp /etc/flussonic/flussonic.conf /mnt/usb/flussonic.conf
cp /etc/flussonic/license.txt /mnt/usb/license.txt
```

После копирования не забудьте отмонтировать накопитель:

```
sudo umount /dev/sdb1
```

Установка конфигурации на новый сервер Установите *Flussonic Media Server* на новый сервер:

```
curl -sSf https://flussonic.com/public/install.sh | sh
```

Создайте директорию, в которую будем монтировать USB-носитель:

```
mkdir -p /mnt/usb
```

Вставьте носитель в USB порт и узнайте имя устройства:

Смонтируйте:

```
mount /dev/sdb1 /mnt/usb
```

Перенесите файлы конфигурации:

```
cp /mnt/usb/flussonic.conf /etc/flussonic/flussonic.conf
cp /mnt/usb/license.txt /etc/flussonic/license.txt
```

Запустите *Flussonic Media Server*:

```
service flussonic start
```

Готово!

Перенос архива

Для переноса архива используйте механизм репликации.

Перенос архива возможен в плановом режиме, то есть когда старый сервер может некоторое время работать одновременно с новым. Обратите внимание, что для этого у вас должна быть в запасе лицензия хотя бы на один дополнительный сервер. Например, если у вас лицензия только на один сервер, вы не сможете запустить несколько серверов Flussonic Media Server одновременно.

Если миграция происходит из-за неисправности старого сервера, то перенести архив не удастся. Просто начните запись архива на новом сервере заново. Во избежание потери архива в будущем заранее настройте репликацию.

2.2.3 Безопасность Flussonic Media Server

В этом разделе мы расскажем, как ограничить доступ к панели Администратора *Flussonic Media Server*.

> Если злоумышленник получит доступ к *Flussonic Media Server*, то он сможет прочесть и перезаписать любой файл на жестком диске.

Логин и пароль

В конфигурации *Flussonic Media Server* настраиваются два разных уровня доступа: `view_auth` и `edit_auth`.

- `view_auth user password;` — используется для предоставления доступа на чтение к API *Flussonic Media Server*, т.е. получении информации о потоках, состоянии записи и т.п.
- `edit_auth user password;` — используется для предоставления полного доступа Администратору. С этим логином и паролем пользователю предоставляются практически все права на работу с сервером.

Flussonic может хранить пароль в 7Cbody%7Cedit-auth%7Chashtag/config/operation/config-save%7Cbody%7Cedit-auth%7Chashхешированном виде. Мы знаем, что клиенты часто забывают пароль и подсматривают его в конфиг файле, даже когда факта компрометации нет.

Мы рекомендуем пользоваться этой опцией, если сервером пользуется группа людей, а внутри компании используются автоматические средства по отслеживанию паролей, которые хранятся без защиты.

Ограничение доступа к Flussonic UI по IP-адресам и портам

Доступ к *Flussonic UI* огранивается по принципу «белого списка», т.е. вы должны перечислить все порты, на которых должен открываться *Flussonic UI*. Для этого перейдите в раздел **Listeners** на вкладке **Config** и укажите порты, которые должен слушать *Flussonic*. Вы можете оставить поле **Address** пустым, чтобы разрешить все IP-адреса, или указать требуемое значение, чтобы разрешить запросы только по указанному IP-адресу.

Опционально для HTTPS, RTMPS и RTSPS вы можете выбрать в поле **SSL protocol** одну или несколько версий SSL, которые хотите использовать на указанном порту. Клиентам, не поддерживающим выбранные версии, будет запрещен доступ.

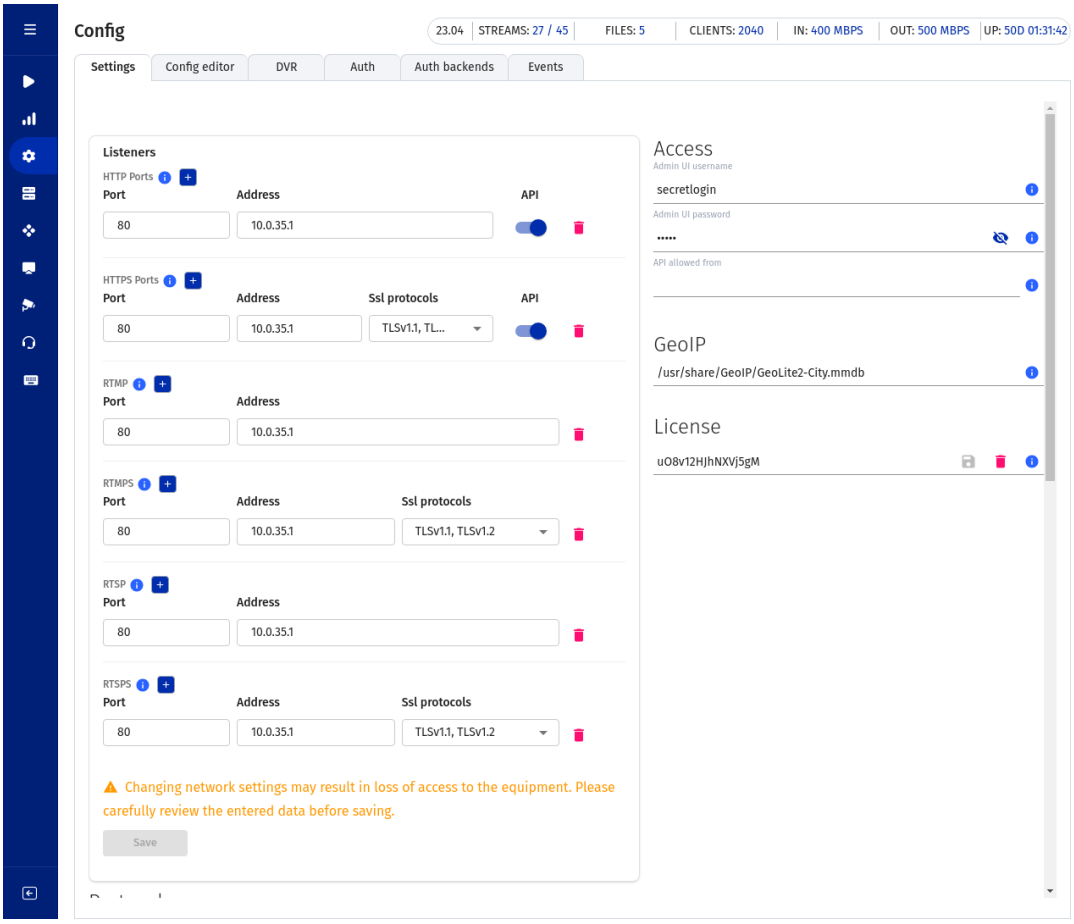


Figure 2.9: Listeners config

Соблюдайте осторожность, изменяя настройку **Адрес**. Некорректное значение может привести к тому, что *Flussonic UI* станет недоступным с вашего компьютера, например, если вы укажете IP-адрес в локальной сети, производя настройку через Интернет. Рекомендуется всегда иметь в запасе альтернативный способ доступа к серверу на случай потери доступа к UI, например, физический доступ или доступ по SSH.

Ограничение вызовов API по IP-адресам и портам

По умолчанию *Flussonic* обрабатывает API-запросы на все HTTP и HTTPS порты, которые вы укажете в настройках.

Вы можете запретить *Flussonic* обрабатывать API-запросы на тех или иных портах одним из следующих способов:

- в *Flussonic UI*, перейдите в раздел **Listeners** на вкладке **Config** и отключите переключатель **API** для тех IP-адресов и портов, по которым не нужно обрабатывать API-запросы:

- в файле конфигурации (`/etc/flussonic/flussonic.conf`) добавьте директиву `api false;` в параметры того IP-адреса и порта, на которых хотите запретить API-запросы:

```
https 443 {
  api false;
}
```

Загрузка SSL сертификатов

Если у вас уже есть SSL сертификат для *Flussonic*, выпущенный сторонним провайдером или сгенерированный вами самостоятельно, его можно загрузить с вашего компьютера на сервер через веб-интерфейс Администратора.

- Настройте порт для HTTPS в конфигурации *Flussonic*. Откройте веб-интерфейс и укажите порт для HTTPS перейдя в **Config** -> **Settings** -> **Listeners**, например, 443.
- Теперь перейдите в **Config** -> **TLS-tunneled protocols**, нажмите **Upload certificates** и выберите файлы сертификата и ключа. Кроме того, можно указать CA-сертификат.

Если при попытке добавить сертификат выводится ошибка, убедитесь в том, что в вашем файле сертификата содержится только один сертификат. В настоящее время Flussonic не поддерживает загрузку через интерфейс файлов, в которых кроме самого сертификата содержится что-либо еще (корневой и/или промежуточный сертификаты, ключ и т.п.). Для загрузки таких файлов используйте другие способы, например, передавайте их по SSH.

Любые SSL сертификаты для *Flussonic* сохраняются в специальной папке — `/etc/flussonic` или `/etc/streamer` (для кластерной установки). При сохранении *Flussonic* заменит названия файлов на `streamer.crt`, `streamer-ca.crt`, и `streamer.key`.

Чтобы удалить загруженные файлы, относящиеся у сертификату, нажмите значок корзины в **Config** -> **TLS-tunneled protocols** напротив списка файлов.

Генерация SSL-сертификатов

Чтобы перевести веб-интерфейс Администратора на HTTPS, нужно включить порт для HTTPS в конфигурации *Flussonic*. Откройте веб-интерфейс и укажите порт для HTTPS в **Config** > **TLS-tunneled protocols**, например, 443.

Можно сгенерировать собственный SSL сертификат сервера *Flussonic*. Для генерации самоподписанных сертификатов *Flussonic Media Server* выполните по порядку команды, приведенные ниже. Каждый раз, когда система будет запрашивать пароль для сертификата, жмите **Enter**, ничего не вводя.

```
cd /etc/flussonic

openssl genrsa -des3 -out streamer.key 1024
```

```
openssl req -new -key streamer.key -out streamer.csr

mv streamer.key streamer.key.org

openssl rsa -in streamer.key.org -out streamer.key

openssl x509 -req -days 365 -in streamer.csr -signkey streamer.key -out
    streamer.crt
```

Файлы нужно поместить в `/etc/flussonic/` (`/etc/flussonic/streamer.crt` и `/etc/flussonic/st`
Загрузить эти файлы можно и через веб-интерфейс. Для этого перейдите в **Config** > **TLS-tunneled protocols** и нажмите **Upload certificates**.

Промежуточный и CA сертификаты будут браться из `/etc/flussonic/streamer.crt`.

Самое актуальное описание команд OpenSSL доступно в manpages в документации OpenSSL.

Let's Encrypt сертификаты

Компания *LetsEncrypt* с апреля 2016 года предлагает бесплатные SSL сертификаты сроком действия на месяц. Создание сертификата происходит в автоматическом режиме.

Мы добавили в *Flussonic Media Server* поддержку *Letsencrypt*. Как получить LetsEncrypt

Защита конфигурационного файла от изменения

Можно запретить изменение конфигурационного файла через API (т.е. веб-интерфейс). Для этого создайте файл `/etc/flussonic/flussonic.conf.locked`, выполнив в командной строке команду:

```
touch /etc/flussonic/flussonic.conf.locked
```

Теперь изменить настройки *Flussonic* через веб-интерфейс нельзя.

Запуск Flussonic от имени непривилегированного пользователя

Чтобы запустить *Flussonic Media Server* под обычным пользователем, выполните следующие настройки:

```
adduser flussonic --home /var/lib/flussonic --disabled-password
chown -R flussonic /etc/flussonic/
chown -R flussonic /var/lib/flussonic/
chown -R flussonic /var/run/flussonic /var/log/flussonic /etc/flussonic/.
    erlang.cookie
setcap cap_net_bind_service=+ep /opt/flussonic/lib/erlang/erts-*/bin/x86_64
    -linux-gnu/beam.smp
```

Затем создайте `override.conf`-файл для юнита `systemd`, используя команду `systemctl edit flusson`

```
[Service]
User=flussonic
Group=flussonic
```

Теперь после перезапуска сервер *Flussonic Media Server* будет работать с правами пользователя `flussonic`.

Чтобы снова запускать *Flussonic Media Server* под суперпользователем, очистите override-файл.

Защита видео от просмотра Администратором

По умолчанию пользователи с правами Администратора *Flussonic* могут проигрывать любой поток в веб-интерфейсе Администратора. Для этого используется специальный авторизационный токен Администратора.

Возможно, вы захотите запретить просмотр некоторых потоков Администратором — тех потоков, для которых нужна авторизация пользователя.

Чтобы запретить Администратору *Flussonic* проигрывать в веб-интерфейсе потоки, защищенные авторизацией:

1) Отредактируйте unit-файл сервиса *Flussonic* (`/lib/systemd/system/flussonic.service`). Следует делать это, используя `systemd override`:

```
systemctl edit flussonic
```

Эта команда откроет текстовый редактор (обычно `nano`).

2) Добавьте строки:

```
[Service]
Environment=STREAMER_ADMIN_VIEW_DISABLE=true
```

Нажмите `Ctrl-X`, затем Y, чтобы сохранить и выйти.

3) Перезапустите *Flussonic*:

```
service flussonic restart
```

Теперь, если поток требует авторизации, плеер в веб-интерфейсе вернет ошибку 403 при попытке проиграть поток с токеном Администратора.

Потоки без настроенной авторизации будут воспроизводиться как обычно.

Защита файловой системы от доступа из UI

В веб-интерфейсе пользователь (Администратор *Flussonic*) указывает расположение директорий для хранения DVR, VOD и кэш. Вы можете ограничить Администратора, указав одну или более

директорий, выше которых *Flussonic* не разрешит размещать хранилища. Это позволит, например, защитить директорию `/root`.

Flussonic проверяет пути внутри `vod vod`, `dvr`, `cache`, `copy`, и в схемах `playlist:///` и `sqlite:///`.

Для настройки добавьте переменную окружения `FLUSSONIC_DATAPATH` и укажите самую верхнюю директорию или несколько директорий, где допускается создавать VOD локации, DVR, кэш и др.

> Чтобы *Flussonic* запустился успешно с новыми настройками, в текущей конфигурации не должно быть путей к директориям, находящимся выше указанных в переменной `FLUSSONIC_DATAPATH`.

Для добавления `FLUSSONIC_DATAPATH` можно использовать возможности `systemd override`:

```
systemctl edit flussonic
```

Эта команда откроет текстовый редактор (обычно это `nano`). В редакторе введите, например, такие директории:

```
[Service]
Environment=FLUSSONIC_DATAPATH=/storage:/mount:/copy
```

Нажмите `Ctrl-X`, затем Y, и `Enter` чтобы сохранить и выйти.

Перезапустите *Flussonic*:

```
service flussonic restart
```

Теперь в настройках можно будет указывать пути для DVR, кэша и VOD файлов только в `/storage`, в `/mount` и `/copy` и вложенных директориях.

2.2.4 Flussonic и файрвол

Не устанавливайте другие программы на сервер с установленным Flussonic. Эти программы могут конфликтовать друг с другом и с Flussonic, а их интеграция выходит за рамки базовой поддержки.

Файрвол (*от англ.* firewall) — это программа для защиты сети от несанкционированного доступа из внешней сети на основе заданных правил. Файрвол ограничивает как входящие, так и исходящие соединения.

В видеостриминге на практике влюченный файрвол не добавляет безопасности серверу, а создает проблемы. На видеостриминговом сервере открыты только те порты, которые требуются для обслуживания клиентов, а файрвол закрывает доступ к серверу по портам и не анализирует сам трафик. Если вы хотите повысить безопасность, то уберите необходимый порт с публичного интерфейса.

Когда вы устанавливаете Flussonic на сервер и задаёте порт администратора, то у вас открыто два (с HTTPS-портом 443 — три) порта: HTTP-порт 80 и порт SSH. Файрволу нечего защищать.

Если вы не можете обойтись без файрвола из-за бумажной безопасности (paper security) и соответствия регламентам, то обратите внимание на следующее:

- Важно различать ограничения на входящие и исходящие соединения. Поскольку на сервере с Flussonic открыты два порта, то ограничивать входящие соединения бессмысленно, как и исходящие. Если злоумышленник подключился к серверу, то поздно защищать сервер.
- Файрвол ограничивает работу стримингового протокола WebRTC, потому что Flussonic случайно выбирает порты для вещания. По этой же причине файрвол может также ограничивать передачу данных по UDP multicast.
- Файрвол, запрещающий исходящие соединениям, закрывает доступ к системе лицензирования. Flussonic сам соединяется с сервером лицензий. Если вы заблокируете исходящие соединения, то потеряете доступ к лицензии. Вопрос о том, какой IP-адрес открыть для нормальной работы лицензии, некорректен, так как неясно, какие типы соединений требуется разрешить: входящие или исходящие. Мы также оставляем за собой право менять IP-адрес сервера лицензий, поэтому, если вы добавите его в правила файрвола, то это временно.
- При обращении в поддержку вас попросят отключить файрвол. В 90% случаев проблемы решаются отключением файрвола.

Узнайте подробнее о защите сервера в статье Безопасность Flussonic Media Server.

2.2.5 Поддержка

Здесь вы узнаете о порядке обращения в техническую поддержку Flussonic, настройке доступа по SSH к вашему серверу и о лог-файлах Flussonic.

Самостоятельное решение проблем

Перед заведением обращения в поддержку, проверьте наличие вашего вопроса в технической документации, возможно ответ уже есть.

Обращения в техническую поддержку

Перед запросом в техническую поддержку необходимо:

- Перейдите на страницу **Config** в веб-интерфейсе Flussonic, прокрутите до группы **Additional** и включите уровень логирования **debug**. Сохраните настройки.
- Если ошибка возникает после каких-то определенных действий, то необходимо попытаться воспроизвести их, чтобы информация попала в лог-файлы.
- Перейдите на страницу **Support** в веб-интерфейсе Flussonic. Для загрузки отладочной информации через интерфейс Watcher, откройте страницу **Состояние** и нажмите кнопку **Загрузить отладочную информацию**
- Опишите последовательность действий, которые привели к ошибке, включая имена потоков, тестовое устройство (ОС, браузер, плеер или приставка) и прочую информацию, которая обязательно понадобится инженерам поддержки. Необходимо описать, что вы ожидали и что вместо этого получили.
- **После окончания загрузки будет показан UUID. Пришлите его нам.**
- Логи, сохранённые в документ Microsoft Word, удаляются. Файл `flussonic.log` также не нужно присылать, он загружается автоматически через форму.
- Если сервис не стартует, выполните **service flussonic run** и пришлите содержимое консоли в текстовом виде. Удобнее, если это будет **.txt** файл, а не скриншот.

Онлайн чат на сайте и форум не являются официальными каналами получения технической поддержки и подходят только для быстрой консультации.

Только обращения, открытые через личный кабинет или support@flussonic.com, с подробным описанием ситуации обрабатываются в первую очередь.

Отправка логов через консоль Если у вас нет доступа к пользовательскому интерфейсу, вы можете загрузить отладочную информацию с помощью следующей команды:

```
service flussonic upload-logs
```

После выполнения команды вы получите UUID, который должны сообщить нам.

Настройка SSH-доступа к вашему серверу

При обращении в поддержку, в некоторых случаях нужно предоставлять рутовый доступ по SSH до вашего сервера. Это нужно, например, при исправлении утечек памяти, поврежденных при сбое архивных файлов на жестком диске, проблем с UDP источниками и так далее.

Для предоставления доступа необходимо положить в домашнюю директорию пользователя root в `/root/.ssh/authorized_keys`

Для добавления ключа можно воспользоваться скриптом. Скачайте и запустите на сервере с правами суперпользователя:

```
sudo -i
curl -s https://flussonic.com/public/ssh-access.sh | sh
```

Сообщите нам IP адрес сервера и порт SSH, если он отличается от стандартного.

Другой способ предоставить нам доступ по SSH — это нажать кнопку **Enable SSH Access** в разделе **Support** пользовательского интерфейса Flussonic.

Утилиты для выявления проблем Мы используем в работе утилиты `screen` и `tcpdump`. Установите их, пожалуйста, если не устанавливали ранее:

```
apt-get -y install screen tcpdump
```

После завершения работ не забудьте удалить наш ключ.

> Не присылайте нам пароль для доступа по SSH. Это небезопасно. Мы не используем программы удаленного доступа, такие как AnyDesk, TeamViewer или VNC для оказания технической поддержки. Для поиска и устранения неполадок нам требуется доступ к вашей системе по SSH. Мы не сможем предоставить вам публичные IP-адреса, которые будут использоваться для доступа к вашему серверу.

Логи

Основой диагностики ошибок в работе Flussonic Media Server являются логи. По умолчанию они пишутся в директорию `/var/log/flussonic`.

Запись идет в файл `flussonic.log`. Когда размер этого файла превышает 40 Мб, выполняется ротация:

- Система упаковывает исходный файл в архив `flussonic.log.1.gz`, а затем продолжает запись логов в файл `flussonic.log`.
- Новый файл логов архивируется в `flussonic.log.1.gz`, а предыдущий архив `flussonic.log.1.g` переименовывается в `flussonic.log.2.gz` и так далее. Система хранит до 40 таких архивов.

> Такая ротация также применяется и к другим, более специфичным типам логов (`crash.log`, `access.log`и т.д.).

Запись идет в файлы вида `flussonic.log`, `flussonic.log.1` и т.п.

Если Flussonic Media Server не запускается или есть какие-то проблемы с записью журнальных файлов, то сначала сервис надо запустить в `foreground` режиме:

```
service flussonic run
```

Довольно часто ошибки во Flussonic Media Server являются лишь следствием других проблем. Изучите и покажите нам логи `/var/log/kern.log` и `/var/log/syslog`.

Записи журналов ведутся в часовом поясе UTC, и Flussonic не предлагает возможности изменить это. Возможно, это неудобно если вы используете только один часовой пояс, но это единственный хороший способ справиться с такими вещами, как переход на летнее время или обслуживание и техническая поддержка серверов, расположенных в разных часовых поясах.

2.3 Monitoring

2.3.1 SNMP

Во *Flussonic Media Server* есть базовая реализация SNMP. Она позволяет отслеживать разные параметры, например, сколько ресурсов потребляют видео-потоки *Flussonic*.

Она включается в конфигурационном файле *Flussonic* следующим способом:

```
snmp 4000;
```

Примените настройки:

```
service flussonic reload
```

Теперь на порту 4000 *Flussonic* будет отвечать по SNMP.

snmpwalk Чтобы прочитать данные SNMP, выполните следующие команды:

```
apt-get -y install snmp snmp-mibs-downloader

snmpwalk -c USERNAME -v 2c -M +/opt/flussonic/lib/mibs -m +STREAMER-MIB
   127.0.0.1:4000 .
```

Замените USERNAME на логин Администратора *Flussonic*.

Здесь `snmpwalk` — утилита для диагностики установленной системы SNMP.

Опция `-c` USERNAME означает "community" в терминах SNMP. SNMP community равен логину Администратора *Flussonic*.

Пример

Если все настроено правильно, то утилита `snmpwalk` выведет ответ следующего вида:

```
# snmpwalk -c flussonic -v 2c -M +/opt/flussonic/lib/mibs/ -m +STREAMER-MIB
    127.0.0.1:4000 .
SNMPv2-SMI::mib-2.1.1.0 = STRING: "Streamer 21.04"
SNMPv2-SMI::mib-2.1.2.0 = OID: STREAMER-MIB::streamerModule
SNMPv2-SMI::mib-2.1.3.0 = Timeticks: (668596) 1:51:25.96
SNMPv2-SMI::mib-2.1.4.0 = STRING: "support@flussonic.com"
SNMPv2-SMI::mib-2.1.5.0 = STRING: "Streamer"
SNMPv2-SMI::mib-2.1.6.0 = STRING: "Erlang"
SNMPv2-SMI::mib-2.1.7.0 = INTEGER: 72
SNMPv2-SMI::mib-2.1.8.0 = Timeticks: (0) 0:00:00.00
SNMPv2-SMI::mib-2.11.1.0 = Counter32: 9
SNMPv2-SMI::mib-2.11.3.0 = Counter32: 0
SNMPv2-SMI::mib-2.11.4.0 = Counter32: 0
SNMPv2-SMI::mib-2.11.5.0 = Counter32: 0
SNMPv2-SMI::mib-2.11.6.0 = Counter32: 0
SNMPv2-SMI::mib-2.11.30.0 = INTEGER: 1
SNMPv2-SMI::mib-2.11.31.0 = Counter32: 0
SNMPv2-SMI::mib-2.11.32.0 = Counter32: 0
STREAMER-MIB::streamsNum.0 = Gauge32: 3
```

```
STREAMER-MIB::sIndex.1 = INTEGER: 1
STREAMER-MIB::sIndex.2 = INTEGER: 2
STREAMER-MIB::sIndex.3 = INTEGER: 3
STREAMER-MIB::sName.1 = STRING: nino
STREAMER-MIB::sName.2 = STRING: 01
STREAMER-MIB::sName.3 = STRING: informer
STREAMER-MIB::sClientCount.1 = Gauge32: 0
STREAMER-MIB::sClientCount.2 = Gauge32: 0
STREAMER-MIB::sClientCount.3 = Gauge32: 1
STREAMER-MIB::sRetryCount.1 = Gauge32: 0
STREAMER-MIB::sRetryCount.2 = Gauge32: 0
STREAMER-MIB::sRetryCount.3 = Gauge32: 0
STREAMER-MIB::sLifeTime.1 = Counter64: 6707983
STREAMER-MIB::sLifeTime.2 = Counter64: 6713981
STREAMER-MIB::sLifeTime.3 = Counter64: 1617905203800
STREAMER-MIB::sBitrate.1 = Counter64: 3571
STREAMER-MIB::sBitrate.2 = Counter64: 2580
STREAMER-MIB::sBitrate.3 = Counter64: 713
STREAMER-MIB::sBytesIn.1 = Counter64: 3043133741
STREAMER-MIB::sBytesIn.2 = Counter64: 2227175658
STREAMER-MIB::sBytesIn.3 = Counter64: 144049422
STREAMER-MIB::sBytesOut.1 = Counter64: 0
STREAMER-MIB::sBytesOut.2 = Counter64: 23894379
STREAMER-MIB::sBytesOut.3 = Counter64: 29263651
STREAMER-MIB::sStatus.1 = INTEGER: active(1)
STREAMER-MIB::sStatus.2 = INTEGER: active(1)
STREAMER-MIB::sStatus.3 = INTEGER: active(1)
STREAMER-MIB::totalClients.0 = Gauge32: 1
STREAMER-MIB::schedulerLoad.0 = Gauge32: 0
SNMPv2-SMI::snmpModules.1.1.6.1.0 = INTEGER: 1091569939
SNMPv2-SMI::snmpModules.10.2.1.1.0 = STRING: "Streamer"
SNMPv2-SMI::snmpModules.10.2.1.2.0 = INTEGER: 1
SNMPv2-SMI::snmpModules.10.2.1.3.0 = INTEGER: 6686
SNMPv2-SMI::snmpModules.10.2.1.4.0 = INTEGER: 484
SNMPv2-SMI::snmpModules.11.2.1.1.0 = Counter32: 0
SNMPv2-SMI::snmpModules.11.2.1.2.0 = Counter32: 0
SNMPv2-SMI::snmpModules.11.2.1.3.0 = Counter32: 0
SNMPv2-SMI::snmpModules.11.2.1.3.0 = No more variables left in this MIB
   View (It is past the end of the MIB tree)
```

В данной SNMP таблице есть объекты, которые относятся к *Flussonic* (`STREAMER-MIB`) и содержат данные о потоках, такие как имя потока (`STREAMER-MIB::sName`), количество запросивших клиентов (`STREAMER-MIB::sClientCount`), время активности потока (`STREAMER-MIB::sLifeTime`) и т.д.

Потоки пронумерованы следующим образом: .1, .2 и т.д.

Поясним объекты, которые могут вызвать вопросы:

- `STREAMER-MIB::sStatus`

Возвращает целые числа, соответствующие следующим значениям:

* active = 1 * notInService = 2 * notReady = 3

- `STREAMER-MIB::schedulerLoad`

Потребление в % ресурса `Erlang scheduler` (среднее за последнюю минуту). Соответствует среднему значению из **Pulse** > *Scheduler utilization for last minute*.

snmptranslate Чтобы получить информацию об объектах и идентификаторах (OID), воспользуйтесь утилитой `snmptranslate` с флагом `-Tz`:

```
snmptranslate -m /opt/flussonic/lib/mibs/STREAMER-MIB.mib -Tz
```

Утилита дает ответ такого вида:

```
"org"                   "1.3"
"dod"                   "1.3.6"
"internet"              "1.3.6.1"
"directory"             "1.3.6.1.1"
"mgmt"                  "1.3.6.1.2"
"mib-2"                 "1.3.6.1.2.1"
"transmission"          "1.3.6.1.2.1.10"
"experimental"          "1.3.6.1.3"
"private"               "1.3.6.1.4"
"enterprises"           "1.3.6.1.4.1"
"streamerModule"        "1.3.6.1.4.1.36342"
"streamer"              "1.3.6.1.4.1.36342.1"
"streams"               "1.3.6.1.4.1.36342.1.1"
"streamsNum"            "1.3.6.1.4.1.36342.1.1.1"
"streamsTable"          "1.3.6.1.4.1.36342.1.1.2"
"streamsEntry"          "1.3.6.1.4.1.36342.1.1.2.1"
"sIndex"                "1.3.6.1.4.1.36342.1.1.2.1.1"
"sName"                 "1.3.6.1.4.1.36342.1.1.2.1.2"
"sClientCount"          "1.3.6.1.4.1.36342.1.1.2.1.3"
"sRetryCount"           "1.3.6.1.4.1.36342.1.1.2.1.4"
"sLifeTime"             "1.3.6.1.4.1.36342.1.1.2.1.5"
"sBitrate"              "1.3.6.1.4.1.36342.1.1.2.1.6"
"sBytesIn"              "1.3.6.1.4.1.36342.1.1.2.1.7"
"sBytesOut"             "1.3.6.1.4.1.36342.1.1.2.1.8"
"sStatus"               "1.3.6.1.4.1.36342.1.1.2.1.9"
"accounting"            "1.3.6.1.4.1.36342.1.2"
"totalClients"          "1.3.6.1.4.1.36342.1.2.1"
"serverStatus"          "1.3.6.1.4.1.36342.1.3"
"schedulerLoad"         "1.3.6.1.4.1.36342.1.3.1"
"streamerConformance"   "1.3.6.1.4.1.36342.2"
"streamGroup"           "1.3.6.1.4.1.36342.2.1"
"statGroup"             "1.3.6.1.4.1.36342.2.2"
"statusGroup"           "1.3.6.1.4.1.36342.2.3"
"security"              "1.3.6.1.5"
```

```
"snmpV2"                "1.3.6.1.6"
"snmpDomains"           "1.3.6.1.6.1"
"snmpProxys"            "1.3.6.1.6.2"
"snmpModules"           "1.3.6.1.6.3"
"zeroDotZero"           "0.0"
```

Описание объектов можно посмотреть в поле DESCRIPTION файла /opt/flussonic/lib/mibs/STREAME

2.3.2 Мониторинг Flussonic в Teledis

Вы можете мониторить состояние Вашего Flussonic сервера, а так же анализировать качество входящего потока с помощью *Teledis*

Как открыть Teledis ?

Для начала, Вам нужно авторизироваться в **личном кабинете**

Раскройте вкладку *Статистика*.

Здесь вы увидите два типа мониторинга: *Мониторинг входящих потоков* и *Состояние серверов*

2.3.3 Мониторинг входящих потоков

Анализирует качество потока, который приходит в Flussonic.

Отображает общий входящий битрейт потока.

Показывает статус входящего потока: - – Нет входящего потока - – Входящий поток присутствует, ошибок нет - – Входящий поток присутствует, но есть ошибки

Отображает топ потоков по количеству ошибок – эти потоки требуют внимания.

Показывает, как записываются сегменты DVR. Типы качества записи сегментов: - – Сегменты записаны быстрее, чем половина продолжительности сегмента - – Записаны медленнее половины, но быстрее 90% продолжительности - – Записаны медленнее 90%, DVR не может записывать этот поток - – Запись не удалась из-за ошибки в хранилище - – Сегмент не был записан из-за задержки хранилища; он мог бы быть записан, если бы хранилище было быстрее, но был удалён из памяти

—

Детали ошибок потока: - – Потерянные пакеты за период - – Повреждённые полезные данные - – Пропущенные кадры - – Перезапуски соединения из-за проблемы с ts_stuck (например, в RTSP) - – Потеря синхронизации MPEG-TS - – Отсутствие PAT - – Потеря TS сервиса - / / – HTTP ошибки от источника - – Крахи источника - – Пакеты, потерянные по разным причинам - – Пропущенные кадры из-за потери сигнала - – Дублирование кадров из-за медленной подачи данных на вход - – Повторяющийся временной штамп кадра - – Потеря кадра из-за высокой загрузки CPU - – Пакеты с полем адаптации больше размера пакета - – Зашифрованные TS пакеты - – PMT не получен после 0,5 секунд - – Несмежные счётчики непрерывности MPEG-TS - – MPEG-TS пакеты с индикатором ошибки передачи - – Повреждённая контрольная сумма PSI - – PES пакеты, начавшиеся не с старт-кода - – Слишком большие буферы ES, отброшенные без формирования кадра - – Крахи внутри декодера MPEG-TS - – Слишком большой скачок DTS, требуется сброс - – Потеря PID MPEG-TS - – RTP пакеты с неправильным типом полезной нагрузки - – Запрещённые NAL единицы в non-interleaved режиме - – Агент не смог установить требуемое соединение. Ошибка может указывать на проблемы с открытием

TCP-сокета на агенте или на недоступность удалённого хоста - – У агента недостаточно памяти для установления соединения с удалённым хостом - – У агента недостаточно буфера для обработки исходящего трафика - – Агент получает некорректные запросы - – Счётчик неизвестных ошибок агента - - Количество случаев, когда источник данных внезапно изменил временные метки без предварительного уведомления, что указывает на непредсказуемые разрывы или сбросы тайминга потока

Входящий битрейт для потока.

Предупреждения, которые сервер исправил: - – Перезапуски для исправления проблемы ts_stuck - – SR пакеты с повторяющимся RTP временным штампом - – Отклонение времени отправителя от времени сервера - – Время на канале пошло назад - – Время на канале пошло вперёд - – Декодер работает в режиме без маркеров - – Нарушение шаблона FU - – FU с установленными битами Start и End - – Применённое исправление FU - – Повторяющийся DTS - – DTS вернулся назад - – DTS прыгнул вперёд

Качество записи DVR для конкретного потока.

Показывает, какой источник используется для потока.

Количество переключений источников для потока.

2.3.4 Состояние серверов

Дает статистику о состоянии каждого сервера, на графиках можно увидеть следующие параметры: CPU Load, Disk utilization %, Disk write metrics, Scheduler Load и т.д.

Выделяет серверы с проблемами производительности, если нагрузка на процессор, память или планировщик превышает 85% в любой момент времени. Обратите внимание, если какой-либо из этих порогов превышен, так как это может свидетельствовать о потенциальной нестабильности.

– Показывает среднюю загрузку процессора сервера – Показывает нагрузку на планировщик на системном уровне – Процент использования оперативной памяти – Ошибки при операциях с сегментами DVR

– Процент использования диска – Метрики операций записи на диск – Метрики операций чтения с диска

– Количество активных потоковых сессий – Количество подключённых клиентов

– Общий входящий трафик для медиа-данных – Общий входящий трафик для системных данных – Общий исходящий трафик для медиа-данных – Общий исходящий трафик для системных данных

– Процент использования GPU – Текущая температура GPU – Нагрузка на GPU, связанная с видеокодированием – Нагрузка на GPU, связанная с декодированием видео – Использование памяти GPU – Нагрузка на GPU, основанная на ядрах CUDA

2.3.5 Сервис статистики Retroview

Retroview — это облачный сервис по сбору статистики и предоставлению доступа к ней. Retroview доступен в личном кабинете пользователя на сайте.

Flussonic загружает данные о сессиях просмотра видео в облачный сервис, чтобы вы могли увидеть отчёты в личном кабинете в реальном времени. Данные сервиса статистики обновляются раз в сутки. Так данные за текущий день вы увидите на следующий день.

Сбор статистики и базовые отчеты доступны пользователям с активными подписками или разовыми лицензиями с действующим доступом к обновлениям.

Чем полезна статистика

На основе данных сервиса статистики вы узнаете:

- Какие устройства используют зрители для просмотра вашего контента. На основе этих данных вы сможете определить, в каком качестве лучше всего предоставлять контент зрителям. Так для мобильных телефонов достаточно видео среднего качества (360p и 480p), а для телевизора — Full HD.
- Количество выходного трафика. Вы сможете делать отчёты для своих клиентов об объёме раздаваемого трафика и числе просмотров контента. Чтобы получить более точные результаты, используйте другие способы оценки статистических данных и сопоставьте полученные результаты с данными сервиса статистики.
- Из каких стран зрители смотрят контент. На основе данных о географическом распределении ваших зрителей вы сможете определить следующее:
- Какие каналы или иные потоки наиболее популярны у ваших зрителей. Эти данные помогут вам определить, какие каналы интересны и неинтересны вашим зрителям. Так вы сможете перестать оплачивать непопулярные каналы и добавить интересные.
- Есть ли у зрителей проблемы при просмотре контента по протоколам на основе HTTP.

Какие данные хранит Retroview

Каждый сервер *Flussonic* передаёт сервису статистики следующие данные о сессиях:

- Уникальный идентификатор сервера, генерируемый произвольно при первоначальном запуске сервера.
- Время создания сессии.
- Время закрытия сессии.

- Имя канала или имя файла.
- IP-адрес пользователя.
- Уникальный идентификатор сессии.
- Авторизационный токен, используемый при создании сессии.
- Стриминговый протокол сессии.
- Количество переданных в сессии байт.
- Идентификатор пользователя, переданный авторизационным бекендом.
- HTTP или RTMP Referer сессии.
- User-Agent сессии.

Геопозиционирование и разбор User-Agent происходит в сервисе при записи сессий.

Количеством трафика между сервером *Flussonic* и сервисом статистики можно пренебречь.

Что вы увидите в статистике

Сервис статистики Retroview показывает:

- Данные по количеству выходного трафика и времени просмотра для 10 серверов и 10 потоков на странице Dashboard.
- Данные по всем сессиям проигрывания, данные сессий, сгруппированные по потокам, серверам, пользователям, а также уникальные сессии на странице Sessions.

Dashboard

Top 10 На странице **Dashboard** вы увидите следующие данные, отсортированные в порядке убывания, за месяц:

- 10 самых просматриваемых по времени потоков,
- 10 самых крупных по выходному трафику потоков,
- 10 самых крупных по выходному трафику серверов,
- 10 самых крупных по времени просмотра серверов.

HTTP-метрики На вкладке **HTTP-метрики** можно проверить, как проигрываются протоколы на основе HTTP (например, HLS, DASH, WebRTC WHIP/WHEP). Фильтр по самым распространенным кодам ошибок 403, 404 и 500 позволит быстро выявить проблемы за заданный период, например, после обновления системы. Данные сгруппированы по нескольким измерениям, чтобы было проще определить источник неполадок:

- Коды ответов по протоколу — покажут, что не проигрывается определенный протокол.
- Коды ответов по названию потока — покажут, что не проигрывается конкретный поток.
- Коды ответов по DVR — покажут, что не проигрывается архив.
- Коды ответов по имени хоста — покажут, что проблема на том или ином сервере.

Sessions На странице **Sessions** показаны данные сессий, распределённые по следующим категориям:

Каждой категории соответствует отдельная вкладка.

- все сессии **Playback all**,
- потоки **By streams**,
- серверы **By servers**,
- пользователи **By users**,
- уникальные сессии **Unique sessions**.

Playback all На вкладке **Playback all** показан список всех сессий со всех серверов *Flussonic*, отсортированных по дате создания: от самой ранней к самой поздней. **Playback all** показывает следующие данные:

- имя потока **Stream name**,
- стриминговый протокол **Protocol**,
- имя хоста **Hostname**,
- страна и IP-адрес **Country,IP**,
- общее количество выходного трафика **Traffic**,
- дата и время создания сессии **Creation Date**,
- общее время просмотра потока **View time**.

By streams На вкладке **By streams** показана сводная статистика по потокам со всех серверов *Flussonic*. В таблице приведены следующие данные:

- имя потока **Stream name**,
- общее количество выходного трафика **Traffic**,
- общее число просмотров **Views**,
- общее время просмотра **View time**.

By servers На вкладке **By servers** показана сводная статистика для каждого сервера *Flussonic*.

> Если у вас в таблице дублируются имя хоста **Hostname** и IP-адрес **Ip**, это значит, что у вашего сервера *Flussonic* изменился идентификатор сервера. Сводная статистика сгруппирована по уникальному идентификатору сервера, который обновляется при изменении аппаратной конфигурации сервера или лицензии.

В таблице приведены следующие данные:

- имя хоста **Hostname**,
- IP-адрес **Ip**,
- общее количество выходного трафика **Traffic**,
- общее число просмотров **Views**,
- общее время просмотра **View time**.

By users На вкладке **By users** показана сводная статистика по зрителям со всех серверов *Flussonic*. В таблице приведены следующие данные:

- идентификатор клиента **User ID**,
- общее число просмотров **Views**,
- страна и IP-адрес **Country,IP**,
- общее количество потребленного зрителем трафика **Traffic**,
- общее время просмотра **View time**,
- User-Agent **Useragent**,
- Referer **Referer**.

Unique sessions С помощью **Unique sessions** вы можете обнаружить проблемы с учетом сессий или сетевым подключением клиента. Например, если вы видите, что клиент часто переподключается, то у зрителя могут быть проблемы с сетевым подключением.

На вкладке **Unique sessions** отображается список уникальных сессий со следующими данными для каждой сессии:

- имя потока **Stream name**,
- стриминговый протокол **Protocol**,
- страна и IP-адрес **Country,IP**,
- общее количество выходного трафика **Traffic**,
- дата и время создания сессии **Creation Date**,
- общее время просмотра **View time**,
- User-Agent **Useragent**,
- Referer **Referer**.

Данные о схожих сессиях клиента, зафиксированные в течение дня, агрегируются. Если в течение дня есть сессии, у которых совпадает IP-адрес, имя потока, стриминговый протокол, идентификатор клиента, Referer и User-Agent, то данные о таких сессиях объединяются и представляются в виде уникальных сессий.

Фильтры Чтобы ограничить вывод данных на странице, используйте фильтры. Например, вы можете использовать фильтр, чтобы вывести трафик определённого IP-адреса, сессии с определённой продолжительностью и др.

Чтобы использовать фильтры, следуйте этим шагам:

- Откройте фильтры, кликнув на иконку фильтра справа от поля *Stream title here*.
- Задайте необходимые значения для фильтров. Список фильтров указан ниже:
- Чтобы применить фильтры, нажмите **Apply**.

2.4 Devops

2.4.1 Flussonic в Kubernetes

В этой статье представлена базовая информация по работе с *Flussonic* в *Kubernetes*:

- основные термины и понятия,
- особенности запуска Flussonic в Kubernetes.

Чтобы протестировать *Flussonic* в *Kubernetes*, воспользуйтесь инструкцией по работе с media-server-operator

Этот документ актуален как для тех, кто только начинает своё знакомство с *Kubernetes* и хочет попробовать запустить *Flussonic* в этой среде, так и для тех, кто уже активно использует *Kubernetes* в своём бизнесе.

> В этом документе не рассматриваются основы *Kubernetes*, его настройки и использования. Для этого ознакомьтесь с Kubernetes Overview и Начало работы.

Основные термины и понятия

Этот глоссарий предназначен для тех, кто не знаком с *Kubernetes* и только начинает своё знакомство с ним. Мы постараемся объяснить вам некоторые термины и понятия *Kubernetes* так, чтобы вы могли посмотреть на них с другой стороны:

- **Kubernetes** (также известный как *k8s*) — это программа управления кластером, набор стандартов и правил, который позволяет унифицированным способом управлять сложным и динамичным кластером микросервисов. Можно сказать, что *Kubernetes* — кластерная операционная система.
- **Node (также нода, узел)** — машина (виртуальная или физическая) в кластере *Kubernetes*, на которой и запускаются контейнеры.
- **Pod** — это экземпляр программы, запущенной в ОС *Kubernetes*. Эта программа может состоять из одного или нескольких контейнеров. **Pod** — мельчайшая единица развёртывания экосистемы *Kubernetes*. Pod — это группа из одного или нескольких контейнеров, работающих на нодах в кластере *Kubernetes*. Эта программа может состоять из одного или нескольких контейнеров.
- **Deployment** — объект, декларирующий тип развёртывания в *Kubernetes*. Он следит, чтобы в *Kubernetes* было запущено необходимое кол-во единообразных Pod. Так, если бы вы захотели запустить несколько Pod, то без Deployment вам бы пришлось вручную определять каждый Pod. Deployment используется для запуска stateless-приложений, т.е. приложений, без отслеживания состояния. Pod'ы в Deployment являются эфемерными, т.е. непостоянными. Это значит, что если Pod упал и перестал работать, то вместо него будет запущен новый Pod, который не будет знать ничего о том Pod, после которого он был запущен. В Deployment Pod'ы являются взаимозаменяемыми.

- **DaemonSet** — объект, декларирующий тип развёртывания в *Kubernetes*. Он следит, чтобы на каждом Node был запущен ровно один Pod. Его отличие от Deployment состоит в том, что DaemonSet гарантирует уникальность каждого Pod. В DaemonSet Pod'ы имеют свои уникальные идентификаторы, которые совпадают с именем хоста Pod'а. Таким образом, Pod'ы не являются взаимозаменяемыми. DaemonSet подразумевает, что одно и то же имя хоста присваивается Pod независимо от количества перезапусков Pod. Знание об уникальности запущенного Pod для большого количества Pod в кластере является крайне полезным. В контексте *Flussonic* DaemonSet означает, что будет запущен экземпляр *Flussonic* со своим лицензионным ключом, на своём имени хоста и со своей конфигурацией.
- **Volume** — каталог, смонтированный в контейнере Pod.
- **PersistentVolume** (также называемый PV) — дисковое пространство, используемое для хранения данных. Жизненный цикл PersistentVolume не зависит от жизненного цикла использующего его Pod. Поэтому, даже если Pod упадёт и остановит или прекратит свою работу, то PV выживет. В контексте *Flussonic*, PersistentVolume отлично подходит для записи и хранения архива. В случае с облаком, в качестве PersistentVolume предоставляется облачное хранилище. В случае размещения на своих серверах, PersistentVolume — это диски на конкретной ноде.
- **PersistentVolumeClaim** (также называемый PVC) — механизм запроса Pod на PersistentVolume. Pod запрашивает хранилище с помощью PVC. Затем PersisentVolumeClaim пытается найти подходящее хранилище в кластере.
- **ConfigMap** — тип volume, содержащий неконфиденциальные данные в виде пар ключ-значение. В контексте *Flussonic*, в ConfigMap будет храниться статическая конфигурация *Flussonic*. ConfigMap может быть определён как файл на диске либо через переменные окружения.
- **Secret** — тип volume (хранилища), содержащего конфиденциальные данные, такие как имя пользователя, пароль, ключ активации лицензии и т.д. В контексте *Flussonic*, в Secret будут храниться данные `edit_auth`: логин и пароль администратора. Secret может быть определён как файл на диске либо через переменные окружения. В отличие от ConfigMap, данные в Secret надёжно спрятаны.
- **Service** — объект, позволяющий агрегировать много Pod в одном месте сразу. Он даёт доступ через единую точку входа для разных Pod. Service даёт возможность через себя получить доступ к группе Pod.

Особенности запуска Flussonic в Kubernetes

Для запуска *Flussonic* использует данные, указанные в переменных окружения поля `env` в конфигурационном файле Pod `publish.yaml`. Когда речь идёт о конфиденциальных данных, таких как логин и пароль администратора (`edit_auth` в *Flussonic*), *Kubernetes* рекомендует помещать эту информацию в Secrets в виде Base64-строк. Учётные данные извлекаются из Secret и попадают в окружение (поле `env`).

```
kind: Secret
metadata:
  name: test-secret
data:
# root:password
  edit_auth: cm9vdDpwYXNzd29yZA==
```

Для того, чтобы запустить Pod со своими настройками, вам необходимо вместо значений по умолчанию (`root:password`) подставить в переменную `edit_auth` ваши логин и пароль администратора в формате Base64-строки.

Концепции конфигурационных файлов для *Flussonic Media Server* и *Kubernetes* значительно отличаются. *Kubernetes* позволяет создать конфигурационный файл *Flussonic* (`flussonic.conf`) из каталога конфигурационных файлов, в котором содержатся несколько файлов `.conf`, представляющих из себя кусочек конфигурационного файла *Flussonic* (`flussonic.conf`). Давайте посмотрим, как это выглядит на примере:

```
kind: ConfigMap
metadata:
  name: streamer-presets
data:
  ports: |
    rtmp 1935;
  vod: |
    file vod {
      storage /opt/flussonic/priv;
    }
  publish: |
    template pub {
      prefix pub;
      url publish://;
    }
```

В объекте `ConfigMap` в поле `data` содержатся кусочки конфига *Flussonic*, разделённые по смысловым частям (`ports`, `vod` и `publish`). Вы можете заменить эту конфигурацию на свою собственную. Вы также можете задавать части, на которые разбита конфигурация. Затем эти кусочки конфигурации записываются в файлы `.conf`. Это определяется в поле `volumes -> configMap` в секции `items`.

```
volumes:
      - name: config-templates
        configMap:
          name: streamer-presets
          items:
          - key: ports
            path: ports.conf
          - key: vod
            path: vod.conf
          - key: publish
            path: publish.conf
```

Каждая часть записывается в отдельный файл `.conf` и размещается далее в директории конфигурационных файлов `/etc/flussonic/flussonic.conf.d`.

2.4.2 Flussonic k8s Media Server Operator

media-server-operator это расширение для Kubernetes, позволяющее упростить запуск нескольких экземпляров media server с одинаковой конфигурацией. Можно запустить без оператора, прописав все необходимые настройки руками, после чего придется отслеживать все изменения и поддерживать новые возможности Flussonic вручную.

media-server-operator берет на себя следующие вопросы, специфичные для Kubernetes:

- организация хранения лицензионных данных, позволяющая перезапускать медиасервер при потере связи с сервером лицензий
- запуск строго по одному экземпляру на одном сервере с возможностью использования прямого подключения к сети (HostPort)
- настройка всех механизмов мониторинга и слежения за запуском сервера
- удобная настройка конфигурационного файла, защищенного от перезаписи в процессе работы
- создание сервиса для балансировки проигрывания с пула медиасерверов

Работа media-server-operator подразумевается вместе с central-operator, который обеспечит доставку конфигурации потоков до каждого экземпляра медиа сервера.

Запуск оператора

Для того, чтобы воспользоваться оператором, достаточно просто поставить в систему yml файл с CRD (описанием нового типа).

Вам нужен запущенный Kubernetes и настроенный на него kubectl

```
kubectl apply -f https://flussonic.github.io/media-server-operator/latest/
   operator.yaml
```

Для продакшн инсталяции рекомендуем жестко указывать версию, полный список которых доступен в репозитории, например:

```
kubectl apply -f https://flussonic.github.io/media-server-operator/24.3.3/
   operator.yaml
```

После этого можно запускать медиасервера. Пример, который создаст секрет с лицензией и запустит медиасервер ниже:

```
kubectl label nodes streamer flussonic.com/streamer=true
kubectl create secret generic flussonic-license \
    --from-literal=license_key="${LICENSE_KEY}" \
    --from-literal=edit_auth="root:password"
kubectl apply -f https://raw.githubusercontent.com/flussonic/media-server-
   operator/master/config/samples/media_v1alpha1_mediaserver.yaml
```

Вместо `${LICENSE_KEY}` надо подставить ваш лицензионный ключ.

После выполнения этих действий на ноде с именем `streamer` запустится медиасервер.

Поставка

Мы упаковали оператор так, чтобы им было максимально удобно пользоваться как в подключенном к интернету облаке, так и в приватном. Он состоит из двух частей:

- yml файл с описанием k8s CRD (за этими абревиатурами прячется описание нового типа объектов для кубернетеса)
- docker образ с нашим кодом на https://hub.docker.com/r/flussonic/media-server-controller

Их можно скачать и хранить у себя локально.

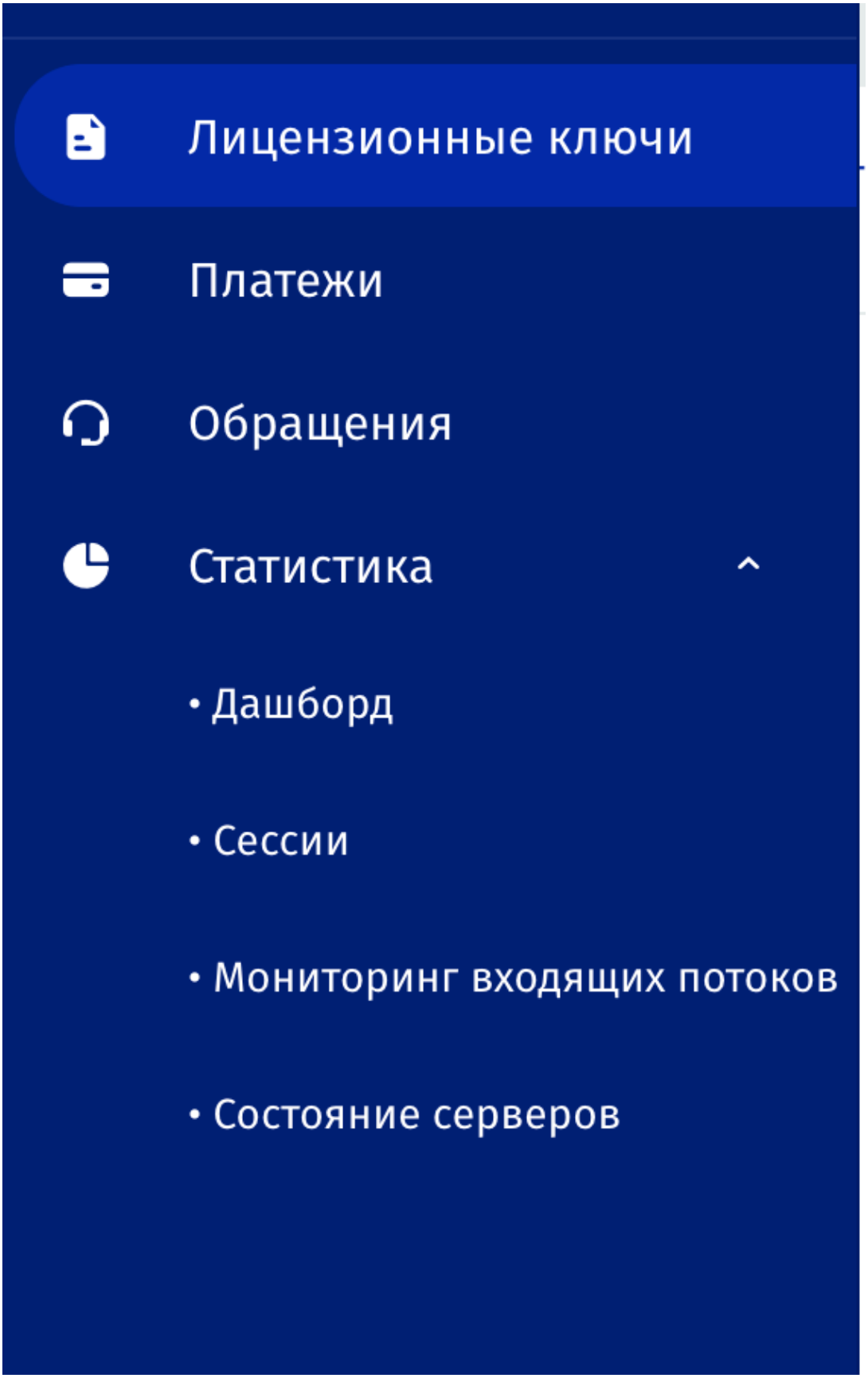


Figure 2.10. Статистика

Figure 2.11: Мониторинг входящих потоков

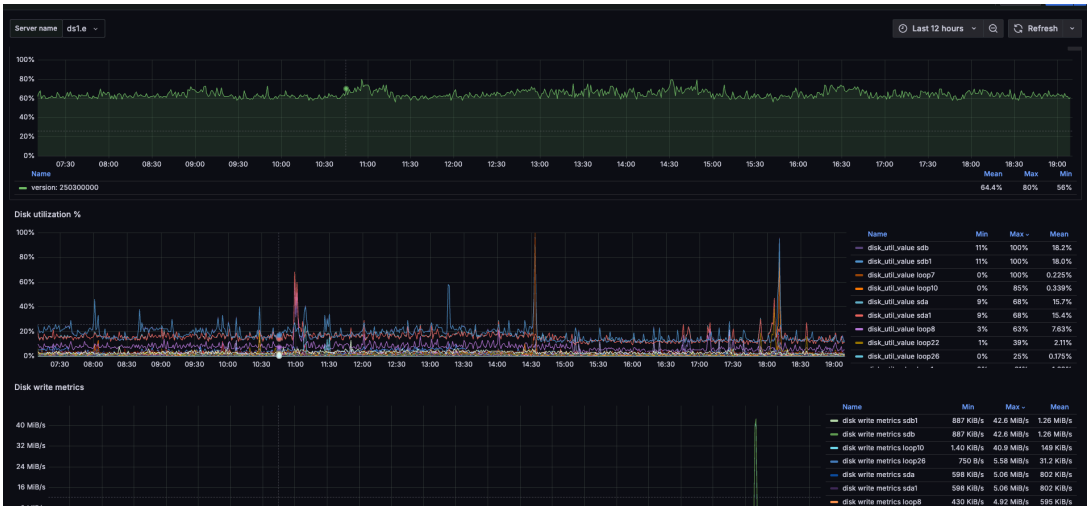


Figure 2.12: Состояние серверов

Figure 2.13: Dashboard Retroview

Figure 2.14: HTTP metrics Retroview

Figure 2.15: Sessions Retroview

Figure 2.16: Filters

Chapter 3

Developers

3.1 TV

3.1.1 Авторизация в Flussonic через middleware

Middleware

Очень важная задача, которую надо решить при запуске OTT IPTV сервиса — ограничение доступа к стриминговым серверам. По нашей статистике многие люди вообще не обращают на это внимание и, как следствие, переплачивают за трафик — их потоки попросту воруют.

Видео можно раздавать всем, но хитро зашифрованное и не всем отдавать ключи, это называется DRM. Другой способ защиты — ограничивать раздачу самого видео, это уже авторизация.

В Flussonic Media Server реализована очень гибкая схема авторизации, требующая определенных действий со стороны Middleware.

Схема работы такая:

- клиентская приставка обращается за адресом потока;
- Middleware отдает адрес с уникальным токеном;
- Flussonic Media Server использует этот токен для идентификации сессии;
- при открытии сессии Flussonic Media Server проверяет этот токен у Middleware.

Такая трехсвязная схема нужна для того, что бы не встраивать авторизацию в Flussonic Media Server. В свою очередь Flussonic Media Server не на каждый запрос клиента ходит к Middleware, а только раз в определенное время.

Вопрос правильного выбора токена открытый и мы можем предложить пару вариантов по его генерации.

Share nothing токен

Можно генерировать токены, упаковывая в них всю необходимую для авторизации информацию. Например, токен можно сгенерировать таким образом:

```
token=sha1(secret_key + ip + stream_name)
```

Такой токен можно проверить, зная лишь `secret_key`. При этом, если злоумышленник попробует воспользоваться этим токеном, у него ничего не получится, потому что IP будет другим.

Однако такой токен можно сохранить и пользоваться им бесконечно. Если пользователь один раз оплатил подписку на сервис, то потом с таким токеном ему можно уже не платить.

В токен можно вставить время:

```
time = utc()
token=sha1(secret_key + ip + stream_name + time)+":"+time
```

Теперь Middleware при проверке может проверить время жизни токена и, если ему больше суток, может его отключать. На практике почти никто (кроме публичных телевизоров и поклонников 24 Le Mans) не в состоянии смотреть эфир больше суток подряд.

Токены в базе

Можно совместить авторизацию с учетом просмотра и под каждый просмотр создавать пользователю новый уникальный токен, помещая его в базу:

```
token=uuid()
```

Потом при повторных обращениях Flussonic Media Server к Middleware можно обновлять статистику по этой сессии, сохраняя информацию о том, кто сколько просмотрел.

3.1.2 Получение EPG из MPEG-TS потоков

Об экспорте EPG

EPG (электронный телегид) — это важный компонент услуг цифрового телевидения. Есть разные способы предоставить EPG пользователям. Например, спутниковое ТВ передает EPG для вещаемых каналов в MPEG-TS потоках, и эта услуга бесплатна. Но можно монетизировать ее.

Flussonic может извлекать расписание передач из метаданных MPEG-TS потоков, полученных со спутникового приемника мультикастом по UDP. Он экспортирует данные о EPG в файлы, которые вы можете получать через HTTP API. По мере поступления новых данных расписание на Flussonic обновляется, и эти обновления можно отслеживать и получать.

Расписание можно использовать на стороннем middleware для приставок, чтобы отдавать его абонентам. Также EPG в JSON-формате можно использовать для интеграции с приложениями в ваших сервисах. Все это значит, что абоненты могут получать расписание через Интернет как часть платных услуг.

Flussonic экспортирует телепрограмму в два формата. Они удобны каждый для своих целей:

- XMLTV. Это стандартный формат для описания телевизионных передач, который вы можете загружать в IPTV middleware. Позволяет просматривать расписание и формировать ссылки на передачи из архива.
- JSON. Файл, имеющий собственную структуру Flussonic. JSON файлы удобно использовать на веб-страницах.

Flussonic формирует расписание для отдельных каналов, для всех каналов и для группы каналов, например, Спорт.

Как получить EPG

Начиная с версии 20.03, необходимо явно включить получение EPG потока в настройках потока при помощи опции `epg on`:

```
stream channel5 {
  input tshttp://trancoder-5:9000/;
  input file://vod/epg.ts;
  epg on;
}
```

Другой способ — включить получение EPG в UI:

- Найдите и кликните поток в **Media**
- В настройках потока перейдите на вкладку **EPG**

Figure 3.1: Flussonic EPG

- Отметьте **EPG** и сохраните настройки.

Теперь можно:

- Получать EPG в виде XMLTV или JSON файлов, чтобы затем использовать их в ваших сервисах.
- Подписаться на событие `epg_changed`, чтобы получать обновленное расписание по мере его обновления.

Внимание. Начиная с версии Flussonic 20.03, для получения EPG достаточно обратиться к потоку по специальному URL. IPTV плагин больше не используется.

***Чтобы получить EPG в формате XMLTV, используйте URL вида*:**

- `/ИМЯ_КАНАЛА/epg.xml` — загружает EPG для канала с указанным названием.
- (устаревший URL) `/tv/channel/ИМЯ_КАНАЛА/epg.xml` — загружает EPG для канала с указанным названием (для версий до 20.03).
- (устаревший URL) `/tv/all/epg.xml` — загружает EPG для всех каналов (для версий до 20.03).

Формат ссылки для загрузки программы передач в XMLTV:

```
http://FLUSSONIC-IP/CHANNEL_NAME/epg.xml
```

***Для получения EPG в формате JSON, используйте URL вида*:**

- `/ИМЯ_КАНАЛА/epg.json` — загружает EPG для канала с указанным названием.
- (устаревший URL) `/tv/channel/ИМЯ_КАНАЛА/epg.json` — загружает EPG для канала с указанным названием (для версий до 20.03).
- (устаревший URL) `/tv/all/epg.json` — загружает EPG для всех каналов (для версий до 20.03).

Формат ссылки для загрузки программы передач в JSON:

```
http://FLUSSONIC-IP/CHANNEL_NAME/epg.json
```

3.1.3 IPTV

Содержание

- Что такое IPTV и IPTV/OTT?
- IPTV и его структура
- Комплексное решение на базе Flussonic Media Server

Что такое IPTV и IPTV/OTT?

Телевидение уже настолько плотно вошло в нашу жизнь, что сложно представить себе дом или квартиру, где жители не смотрят ТВ. На данный момент существует несколько видов телевидения: спутниковое, кабельное, эфирное и относительно новые – **IPTV** и **IPTV/OTT**. Данная статья посвящена рассмотрению классической технологии **IPTV**, способу доставки телевизионного сигнала до зрителя, а также тому, как *Flussonic Media Server* может помочь во внедрении традиционного **IPTV**.

IPTV и его структура

IPTV или *Телевидение по протоколу интернета* (*англ.* Internet Protocol Television) – технология транслирования телевизионных программ с помощью IP сетей (от *англ.* Internet Protocol). Эта технология пришла в конце 90-х годов прошлого века на смену традиционным способам передачи телевизионного сигнала.

IPTV конкурирует с эфирным телевидением, кабельным, а также спутниковым.

Спутниковое, эфирное и кабельное телевидение являются достаточно недорогими и простыми в своей организации решениями, но уступают **IPTV** по ряду характеристик, о которых мы расскажем чуть позже. Рассмотрение схемы трансляции сигнала до конечного пользователя, т.е. зрителя, в рамках спутникового телевидения, кабельного и эфирного находится за рамками нашей статьи.

Классический вариант услуги **IPTV** предоставляется, как правило, тем же оператором или провайдером, что и интернет, в границах сети этого оператора. Обычно провайдер подключается ко всему многоквартирному дому и затем к своим абонентам. Так в одном доме может быть несколько провайдеров, допустим, *Ростелеком*, *SkyNet* и *МТС*, житель этого многоквартирного дома может выбрать одного из этих 3-ёх операторов для подключения услуги **IPTV**. Если же, например, у жителя возникло желание подключить услугу **IPTV** у другого оператора, например, *Дом.ru*, то сначала этому оператору необходимо подключиться ко всему дому, а уже потом к конкретному абоненту. Важно понимать, что в данном случае потоки транслируются не через открытую сеть Интернет, а через кабель. Таким образом, качество предоставляемых услуг зависит не от мировой сети, а от провайдера, который полностью контролирует весь процесс доставки потока данных до абонента.

Традиционно термин **IPTV** описывает конкретный перечень технических решений по получению телевизионных каналов и их ретрансляции абонентам. Классическая **IPTV** архитектура выглядит следующим образом (см. *схему 1.1*):

Схема 1.1. Классическая IPTV архитектура

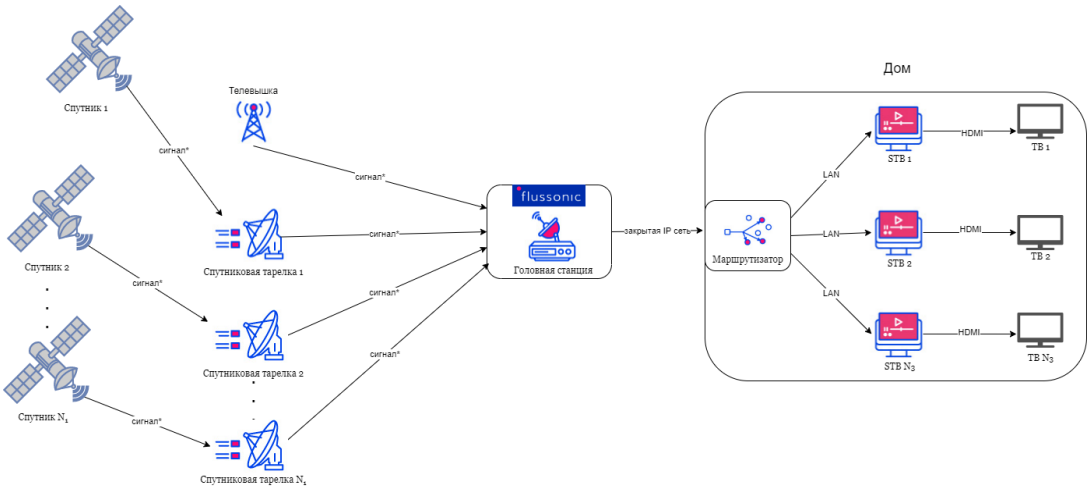


Figure 3.2

***Примечание 1.** *Схема трансляции сигнала по IPTV**, приведённая выше, является классической и общей, в каждом конкретном случае она может претерпевать некоторые изменения.

***Примечание 2.** *сигнал может передаваться в разных стандартах цифрового телевизионного вещания:* DVB, ATSC или ISDB*.

Далее в статье мы будем использовать термин *видеоконтент*. Давайте условимся, что под *видеоконтентом* мы будем понимать не только видеопоток, но ещё и поток аудио, а также файл(ы) с субтитрами и др., если таковые имеются.

В самом простом случае схема **IPTV** сервиса включает в себя спутниковую тарелку, *головную станцию* и набор приставок, по набору сервисов не сильно отличаясь от первого советского телевизора *Рубин*.

Введём некоторые понятия, необходимые для дальнейшего понимания процесса трансляции видеоконтента:

Головная станция – это профессиональный термин для обозначения спутникового приёмника, который способен захватывать очень много телеканалов одновременно. У головной станции три основных задачи:

- Преобразовать цифровой сигнал с тарелки в байты.
- *Дескремблировать*, т.е. расшифровать, его.
- Отправить этот поток байт по *UDP* (*англ.* User Datagram Protocol – протокол пользовательских датаграмм) мультикастом в сеть.

Мультикаст (или *Мультивещание*) – это такой способ передачи данных, когда источник отдаёт данные в приватную IP сеть в широковещательном режиме, не зная кто именно получит данные, так сказать, “распараллеливая” поток данных (отношение: один — ко многим). Заметим, что мультикаст имеет место быть только в контексте приватной сети или сети закрытого доступа. В общем случае телевизионная приставка получателя может быть опционально настроена получать или не получать необходимые данные. Этим мультикаст отличается от броадкаста.

Подробнее про рассылку мультикаста, в том числе с использованием Flussonic Media Server см. Рассылка мультикаста.

Телевизионная приставка или *STB* (от *англ.* set-top-box, «коробка, лежащая наверху телевизора») – это такой небольшой компьютер, который берёт видео из приватной или закрытой IP сети и показывает его на экране телевизора. Основное устройство для управления приставкой – пульт.

Захват сигнала Для захвата контента большинством **IPTV** операторов используется спутниковая тарелка, но это не единственный возможный источник сигнала. На самом деле источников может быть несколько причём разного рода. Например, головная станция может захватывать сигнал как со спутниковых тарелок, так и с телевышки одновременно (см. *схему 1.1*)

Подробнее про захват спутникового видео, в том числе с использованием Flussonic Media Server см. Захват спутникового видео.

Спутниковый канал передачи телевидения существенно дешевле с точки зрения операционных расходов, нежели интернет. Примерную стоимость захвата одного канала профессиональным оборудованием можно оценить в 100-1000$ единоразово. Выделенный же интернет-канал под один телеканал с гарантированным качеством будет стоить столько же, но уже ежемесячно, поэтому зачастую операторы обходятся каналами без гарантированного качества.

Итак, каким же образом происходит передача сигнала в **IPTV**? Телевизионные каналы захватываются со спутника, *дескремблируются*, т.е. расшифровываются, и вещаются *мультикастом*, т.е. в широковещательном режиме, на всех абонентов.Транспортировка сигнала осуществляется в соответствии с определёнными правилами, т.е. это своего рода “упаковка” сигнала перед его отправкой получателю. Такие правила называются *протоколами*.

Сигнал со спутника транслируется по протоколу *DVB-S/S2* (протокол передачи спутникового сигнала) на спутниковую тарелку. Захватив сигнал со спутниковой тарелки, *головная станция*, т.е. спутниковый приёмник, осуществляет преобразование сигнала из протокола *DVB-S/S2* в IP протокол, т.е. “переупаковывает” его, чтобы маршрутизатор смог его принять. Затем сигнал из маршрутизатора поступает в *STB*, т.е. телевизионную приставку или просто приставку, по тому же IP протоколу. Следующим шагом *STB* преобразует сигнал таким образом, чтобы телевизор смог его обработать и воспроизвести желаемый контент. Заключительным этапом этот обработанный сигнал передаётся уже в телевизор по кабелю *HDMI*, где зритель осуществляет просмотр контента.

Может возникнуть вопрос: чем же **IPTV** лучше простой тарелки (*DVB-S/S2*), если оператор всё равно ставит тарелку? Во-первых, оператор ставит не одну тарелку, а 5-6 и более, захватывая все каналы, до которых только может добраться, соответственно, абонент получает гораздо большее количество разных каналов. Во-вторых, **IPTV** предоставляет больше разных удобных

услуг. В-третьих, существенная часть жителей многоквартирных домов в условиях городской застройки не имеет возможности поставить себе спутниковую тарелку ввиду того, что сигнал со спутника просто не достигает тарелки. Это может произойти по следующим причинам:

- характерна для районов плотной застройки, где расстояние между зданиями крайне мало. В таком случае дома преграждают путь сигналу со спутника и тарелка никак не может его принять.
- окна многоквартирных жилых домов выходят на север. Спутники размещаются на геостационарной орбите над экватором и в северном полушарии видны только на юге, соответственно, в таком случае спутниковая тарелка просто-напросто не сможет поймать сигнал.

Конечно, чисто технически установить спутниковую тарелку всё равно можно, но никакого смысла это иметь не будет.

STB-приставка Некоторые приставки *STB* имеют возможность записи передач или постановки эфира на паузу. Важно понимать, что в вопросах записи телевизионного эфира возникают проблемы с точки зрения права. Немало десятилетий прошло прежде чем юристы контент-провайдеров согласились на использование видеомагнитофона пользователями. Из-за этого в современных телевизионных приставках зачастую просто скопирована бессмысленная и неудобная функциональность старых видеомагнитофонов: запись одного канала по предварительному расписанию. В таком случае пользователь должен заранее настроить приставку на запись в нужное время.

Первые приставки были достаточно примитивным прибором, который мог только переключать каналы по предзагруженному плейлисту. Современные приставки зачастую идут с веб-браузерами *Opera* или что-то на базе *Webkit* (свободный движок для отображения веб-страниц), которые модифицированы для обработки сигнала с пульта и специфичных для видео задач. Использование веб-браузера позволяет облегчать процедуру изменения интерфейса и добавления новых сервисов (например, покупки контента одной кнопкой с пульта). Однако, веб-браузеры на медленных процессорах приставок работают медленнее, чем специализированные приложения, поэтому на рынке всё равно есть устройства без веб-браузеров.

Middleware Для предоставления чего-то поинтереснее, чем банальный список из 300 каналов, который необходимо пролистывать с первого до последнего, в инфраструктуру включается новый компонент – *Middleware*.

Middleware – отдельный компонент всей инфраструктуры, который представляет из себя сетевой сервис, предоставляющий доп. сервисы пользователям через приставки. Надо отметить, что и на сегодняшний день не все IPTV сервисы используют *Middleware*, в некоторых случаях приставки получают фиксированный список каналов.

С помощью *Middleware* можно оперативно менять список каналов, предлагать классификацию каналов по жанрам, предоставлять доступ к записанным передачам, фильмам, настроить показ различной информации типа курса валют, прогноза погоды и т.п.

Подробнее см. Middleware

Таким образом выглядит первая классическая модель **IPTV**, созданная ещё в конце прошлого века. Однако технологии не стоят на месте и классическая архитектура **IPTV** претерпела изменения, вследствие чего появилась **IPTV/OTT**.

Подробнее об IPTV/OTT в статье IPTV/OTT.

Комплексное решение на базе Flussonic Media Server

Итак, мы рассмотрели что такое **IPTV** и схему доставки сигнала до абонентов. Какое же место занимает в этой системе *Flussonic Media Server* и как на его базе можно реализовать **IPTV**?

Возможности головной станции можно реализовать с помощью *Flussonic Media Server*, т.е. захват сигнала со спутниковой тарелки и/или телевышки, его дескремблирование и отправка данных на маршрутизатор по IP-сети. Кроме того, *Flussonic* поддерживает DVB-карты захвата, а значит, может напрямую с них принимать потоки. Небольшой сервис в количестве примерно 100 каналов вы можете собрать всего лишь на одном сервере *Flussonic*.

Так вы можете сосредоточиться на опыте контент-мейкера и зрителя, пока *Flussonic* возьмёт на себя всё остальное.

Наше решение позволяет осуществлять доставку сигнала наиболее эффективным способом без потерь качества для пользователя.

Если у Вас возникли вопросы по реализации сервиса **IPTV** на базе *Flussonic Media Server* в рамках вашего проекта или Вы хотите попробовать наш продукт, то, пожалуйста, заполните форму для получения бесплатного тестового ключа.

Если вы не получите от нас письмо в течение часа, то проверьте папку ”Спам” и добавьте наш адрес в ”Избранные контакты”.

Email: support@flussonic.com Тел.: +7 (800) 777-84-13, +7 (495) 481-37-63

3.1.4 Описание IPTV/OTT

Содержание

- Что такое IPTV и IPTV/OTT?

1. IPTV/OTT и его структура 2. Переход от классического IPTV к IPTV/OTT 2. Ключевые особенности реализации IPTV/OTT сервиса 1. Нюансы захвата и транскодирования 2. От catch up (запись архива) к Интерактивному телевидению 3. Линейное телевещание в Wi-Fi сетях 4. Георасределённая доставка видео 3. Комплексное решение на базе Flussonic Media Server

IPTV/OTT и его структура **IPTV/OTT** (от *англ.* Over the Top) – технология передачи видеоконтента через открытую сеть Интернет. Отметим, что предоставление услуги **IPTV/OTT** не осуществляется провайдером сети Интернет и им не контролируется, в отличие от **IPTV**.

Например, захват каналов может осуществляться в Армении, а сам пользователь может находиться в США и смотреть родные каналы, а его провайдер в США даже не будет знать списка предоставляемых каналов.

Эта модель передачи телевизионного сигнала начала своё распространение около 10 лет назад. На данный момент главные провайдеры и операторы связи переходят на **IPTV/OTT** в связи с большей гибкостью технологии и её удобством. Однако модель классического **IPTV** всё ещё используется, но в основном в гостинично-ресторанном бизнесе.

Отличительной чертой **IPTV/OTT** является предоставление контента пользователю напрямую по сетям передачи данных без контакта с оператором связи, в отличие от услуг **IPTV**, которые предоставляются через управляемую оператором сеть.

Архитектура **IPTV/OTT** выглядит следующим образом (см. *схему 1.2*):

Схема 1.2. IPTV/OTT архитектура

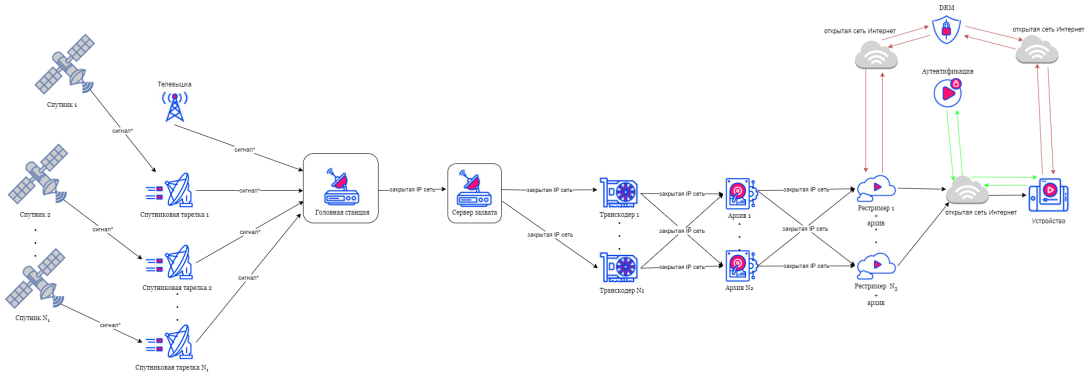


Figure 3.3

Передача сигнала в модели **IPTV/OTT** осуществляется следующим образом:

Первый этап такой же как и в **IPTV** – захват *головной станцией* телевизионного сигнала с источника или нескольких источников разного рода. Дальнейшие этапы прохождения сигнала будут отличаться. Затем преобразованный в IP протокол сигнал передаётся на *сервер захвата*. После этого по закрытой IP сети сигнал передаётся в транскодер (см. Транскодирование), где происходит “распараллеливание” видеодорожки на 3 формата или более (в зависимости от качества входного сигнала): *Full HD* (*1920×1080 пикселей*) , *HD* (*1280×720 пикселей*), *SD* (*720x576 пикселей*). Следующим шагом этот поток передаётся в *DVR*. *DVR* представляет собой хранилище или архив, куда записывается *видеоконтент* и хранится до востребования. Затем сигнал проходит через *рестример* в Интернет. В *рестримере* сигнал зашифровывается, с целью его защиты от сторонних пользователей. Важно отметить следующее: до того, как потоковый сигнал достигает открытой сети Интернет он передаётся по закрытой сети. Перед тем, как начать проигрывать *видеоконтент* на каком-либо устройстве (смартфон, ПК, ТВ) необходимо пройти аутентификацию и получить к нему доступ. Доступ к *видеоконтенту* осуществляется через систему *DRM*, которая предоставляет пользователю ключ для расшифровки. После завершения прохождения всех процедур по расшифровке и аутентификации пользователь может наслаждаться просмотром видеоконтента.

В модели **IPTV/OTT** зрителю стали доступны следующие услуги:

+ *VoD* или *“видео по запросу”* (*англ.* Video on Demand). Услуга по индивидуальному предоставлению видеоконтента пользователю по его требованию или запросу. Так пользователь может выбрать фильм из медиа-библиотеки и посмотреть его. + *nVoD* или *“почти видео по запросу”* (*англ.* Near Video on Demand). Это сервис вещания контента по заранее составленному расписанию для ограниченного круга абонентов (тех, кто оплатил подписку). Таким образом, пользователь может просматривать расписание и смотреть интересующий его контент. + *Time-shifted TV*. Услуга просмотра телепрограмм, при которой можно ставить вещание на паузу и осуществлять перемотку к нужному моменту. + *TVoD* (*англ.* Transactional Video On Demand). Услуга записи эфиров выбранных телеканалов с целью их последующего просмотра в течение некоторого ограниченного промежутка времени (например, 7 дней).

Примерами **IPTV/OTT** могут служить *Netflix*, *Hulu* или *Disney* +.

Переход от классического IPTV к IPTV/OTT Важно понимать, что **IPTV** и **IPTV/OTT** — это два вида доставки контента до конечного пользователя. **IPTV/OTT** считается видом и, в некотором смысле, продолжением **IPTV**. Схематично этот путь можно представить так (см. *схема 2*):

Схема 2. IPTV/OTT передача данных

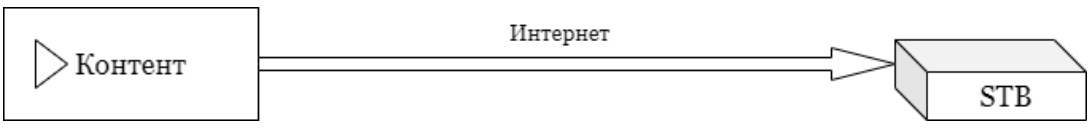


Figure 3.4

Подробнее об IPTV в статье IPTV.

Изначально **IPTV** сигнал передавался по сети закрытого доступа, а в **IPTV/OTT** передача сигнала уже осуществляется через сеть открытого доступа. Отсюда вытекает первое различие — *доступ*

к сети. Изначально (**IPTV**) — закрытый, позже (**IPTV/OTT**) — открытый. Сигнал по **IPTV** практически невозможно "перехватить", поэтому уровень пиратства в таком случае гораздо ниже, чем в случае **IPTV/OTT**. Поскольку это сеть открытого доступа, то и сигнал там "перехватить" гораздо проще.

Следующее — это *контроль за каналом передачи сигнала*. В **IPTV** владельцем канала является оператор-поставщик услуг интернета, который контролирует весь процесс, т. е. ему известно сколько у него пользователей и какой контент они потребляют. Таким образом, имеет место обратная связь. В **IPTV/OTT** же никакого надзора и контроля за каналом передачи сигнала нет, непонятно кто и что смотрит, соответственно, обратной связи нет. В **IPTV** потребитель контента взаимодействует *напрямую со своим оператором*, в то время как в **IPTV/OTT** потребитель взаимодействует уже *напрямую с производителем контента*.

Следующее отличие — *качество передаваемого материала*. В модели **IPTV** сигнал доставляется практически бесперебойно и он достаточно устойчив, что обеспечивает отличное качество контента, в то время как по **IPTV/OTT** сигнал идёт неровно, что не лучшим образом сказывается на качестве. Здесь следует сказать о таком явлении как *адаптивный битрейт* или *ABR* (от *англ.* Adaptive Bitrate). Главной задачей в **IPTV** и **IPTV/OTT** является доставка сигнала без видимых сбоев и задержек для конечного пользователя. Таким образом, учитывая тот факт, что в модели **IPTV/OTT** сигнал может быть неустойчив (в зависимости от скорости и качества интернет-подключения), то качество видео адаптируется под текущее интернет-соединение. Например, имея стабильный и быстрый интернет дома, пользователь спокойно смотрит *Первый канал* со своего смартфона в максимальном качестве, но затем ему понадобилось выйти из дома и качество интернет-соединения резко ухудшилось, что, в свою очередь, привело к ухудшению качества просматриваемого видеоконтента, однако разрыва соединения не произошло.

Одно из основных свойств классического **IPTV** — *географическая привязка*. Имеется ввиду то, что поставляемый контент локален и характерен только для места его распространения. Например, если пользователь подключает себе услугу **IPTV** от *МТС* в Москве, то и смотреть он будет то, что транслирует *МТС* в Москве. **IPTV/OTT** же в этом плане уже более гибкий. С его приходом пользователь, находясь в Новосибирске, спокойно может смотреть контент США.

Говоря о *цене* необходимо заметить следующее: сама *стоимость услуг* и *из чего она складывается*. Начнём со стоимости: **IPTV** дороже **IPTV/OTT**. Стоимость **IPTV** обычно складывается из стоимости пакета: доступ в интернет + сама услуга **IPTV** (т. е. подключение приставки *STB* и её обслуживания). Стоимость же **IPTV/OTT** равна стоимости услуги доступа в интернет.

Следует также отметить, что *скорость запуска нового контента* выше в случае **IPTV/OTT**, нежели в **IPTV**.

Всё вышеперечисленное можно свести в таблицу (см. *табл.1*):

Табл.1. IPTV и IPTV/OTT

Характеристики| IPTV | IPTV/OTT ——|:———:|—— Качество контента | **+** (высокое) | **+/-**(зависит от качества интернет-соединения)| Контроль за потреблением контента | **+** | **-** Скорость запуска нового контента | **-** | **+** Стоимость | **-** | **+** Компоненты стоимости | интернет (если IPTV уже входит в стоимость)/интернет + IPTV | интернет + подписка Надёжность соединения | **+** | **-** Доступ к сигналу | **-** (закрытый доступ) | **+**(открытый доступ)

В заключение необходимо отметить, что нами были рассмотрены классическая схема **IPTV** и её обновлённая версия — **IPTV/OTT**. Главное же отличие между *IPTV* и *IPTV/OTT* состоит в *способе доставки сигнала до конечного пользователя (заключительный этап)*. В случае **IPTV** передача сигнала осуществляется по закрытой сети, а в **IPTV/OTT** — по открытой.

Ключевые особенности реализации IPTV/OTT сервиса

На сегодняшний день операторы **IPTV/OTT** сталкиваются с новыми проблемами, которых не было 5-10 лет назад. Давайте разберёмся в чём же дело и какие есть особенности реализации сервиса **IPTV/OTT**.

Нюансы захвата и транскодирования Спутниковое оборудование крайне инертно. Традиционно в спутниковом телевидении используется кодек *MPEG-2* видео и, условно назовём его, *MPEG-2* аудио. Внедрение кодека *H.264* в спутниковом вещании идёт уже не первый год и до сих пор не завершено.

Ни тот, ни другой кодеки не подходят для iPhone и прочих устройств. Более того, тот сигнал *H.264*, который захватывается сегодня со спутника, также не подходит для iPhone из-за используемого режима intra-refresh (кодирование без опорных кадров).

Кодек *MPEG-2* можно смело заменять на *H.264*, выигрывая почти в 3-4 раза по битрейту, а, следовательно, по трафику и счетам за канал.

Если захватывать *HD-сигнал* со спутника и раздавать его пользователям вне локальной сети, то надо быть готовым к тому, что у большинства пользователей не хватит пропускной способности интернета, а, значит, возникает необходимость кодировать сигнал в несколько качеств для адаптивного переключения битрейта.

В итоге, видео и аудио со спутника приходится транскодировать в *H.264/AAC*, потому что на iPhone и других устройствах видео в таком кодеке не показывается, полосу интернета потребляет больше, чем стоит и *HD* тогда необходимо доставлять с помощью *мультибитрейта*, т. е. в нескольких качествах одновременно в зависимости от пропускной способности интернет-соединения пользователя. Делается это затем, чтобы зритель мог выбрать соответствующий его требованиям сигнал.

Каким образом вопросы захвата и транскодирования сигнала решаются *Flussonic Media Server*?

Flussonic Media Server способен осуществлять захват сигнала по IP протоколу не только с любых устройств и систем приёма сигналов цифрового телевидения (*англ.* Integrated Receiver Decoder — IRD), но и напрямую с плат *DVB-S* и др., что характеризует *Flussonic* как гибкое решение.

Кроме того, *Flussonic Media Server* может декодировать видео из *UDP/HTTP*, *MPEG-TS*, *RTMP* источников и кодировать его в несколько качеств и размеров, что позволяет показывать видео не только на приставках, но и на планшетах и iPhone.

Подробнее см. Захват и Транскодирование.

От catch up (запись архива) к интерактивному телевидению Как мы уже ранее замечали, исторически в приставках есть возможность записи одного канала по запросу. Этот подход неэффективен, потому что люди забывают поставить приставку на запись, а потом злятся: зачем покупать дорогущую приставку, если она ничем не лучше старого видеомагнитофона.

Современный подход к предоставлению доступа к архиву телепередач заключается в следующем: осуществлять запись всего телевизионного эфира на стороне провайдера и предоставлять возможность управления просмотром самому зрителю, а именно:

- просматривать передачи из архива используя расписание телепередач или *EPG* (от *англ.* Electronic Program Guide),
- перематывать видео назад (и вперёд по возможности),
- поставить просмотр на паузу.

Для оказания сервиса *Интерактивного телевидения* необходимо:

- реализовать запись архива на стороне провайдера,
- доработать проигрыватели на стороне зрителя.

Flussonic Media Server предоставляет широкий спектр возможностей для работы с архивом, используя технологию *DVR* (*от англ.* Digital Video Recording), среди которых: удобная навигация и доступ к архиву, неограниченный объём для записи, быстрый предпросмотр кадров из записи без необходимости перематывать саму запись и др.

Подробнее см. DVR

Линейное телевещание в Wi-Fi сетях Традиционный метод мультикаст раздачи столкнулся с *Wi-Fi*. Как мы помним, мультикаст имеет место только в контексте закрытой сети для передачи данных. Так в условиях внедрения *HD* вещания (*6-15 Мбит/с вместо 1-3 на SD*) и распространения *Wi-Fi* сети в домах традиционный мультикаст начинает давать сбой: у зрителей на дорогом телевизоре вместо кристально чистой картинки видны характерные зеленые квадраты. Причиной тому являются потери пакетов по пути от головной станции к приставке.

Flussonic Media Server может выполнять функцию рестримера и осуществлять многопотоковое вещание, давая возможность указать несколько серверов-источников сигнала и построить отказоустойчивую конфигурацию.

Подробнее см. Ретрансляция потоков

Георазпределённая доставка видео В процессе роста количества абонентов **IPTV/OTT** сервиса может возникнуть ситуация, когда обслуживать клиентов с одного центрального сервера становится сложно или попросту невозможно.

Классическая ситуация — открытие филиала оператора в другом городе или появление большого количества абонентов в другой стране/на другом континенте.

В такой ситуации может стать неоправданной раздача видео с центрального сервера, особенно если возникают кластеры/группы пользователей, которые смотрят один и тот же канал, находясь при этом близко друг к другу.

Для того, чтобы сэкономить трафик, применяются ретрансляторы: канал попадает с центрального сервера на удаленный, а с него раздаётся пользователям, которые находятся рядом.

Такая конфигурация становится всё более сложной с ростом количества ретрансляторов и каналов, ведь если каждый канал настраивается вручную, то администратору необходимо вручную обрабатывать огромное количество каналов.

При геораспределённой доставке видео возникает вопрос доступа к архиву: имеет ли смысл писать все каналы на всех серверах? Конечно же нет. Редко используемые каналы имеет смысл писать только в одном месте, но при этом необходимо иметь возможность предоставлять доступ к такому архиву.

Flussonic Media Server имеет ряд механизмов для упрощения решения этих задач.

Подробнее см.DVR и Ретрансляция потоков

Комплексное решение на базе Flussonic Media Server

Итак, мы рассмотрели что такое **IPTV/OTT**, схему доставки сигнала до абонентов, переход от классического **IPTV** к **IPTV/OTT**, а также уделили внимание ключевым особенностям реализации этой технологии. Какое же место занимает в этой системе *Flussonic Media Server* и как с его помощью можно реализовать подобное решение?

Работа *Flussonic Media Server* начинается с этапа захвата головной станцией сигнала со спутниковой тарелки и/или телевышки и заканчивается его доставкой до конечного пользователя. Таким образом, весь этот отрезок пути можно реализовать на нескольких *Flussonic Media Server* (см. *схему 3.1*).

Схема 3.1. Flussonic Media Server в IPTV/OTT

Так вы можете сосредоточиться на опыте контент-мейкера и зрителя, пока *Flussonic* возьмёт на себя всё остальное.

В случае с **IPTV/OTT** каждый отдельный компонент пути (*головная станция, сервер захвата, транскодер, DVR и рестример с функцией DVR*) может быть реализован с помощью *Flussonic*. Наше решение позволяет осуществлять доставку сигнала наиболее эффективным способом и с минимальными потерями качества для пользователя.

Если у Вас возникли вопросы по реализации сервиса **IPTV/OTT** на базе *Flussonic Media Server* в рамках вашего проекта или Вы хотите попробовать наш продукт, то, пожалуйста, заполните форму для получения бесплатного тестового ключа.

Если вы не получите от нас письмо в течение часа, то проверьте папку ”Спам” и добавьте наш

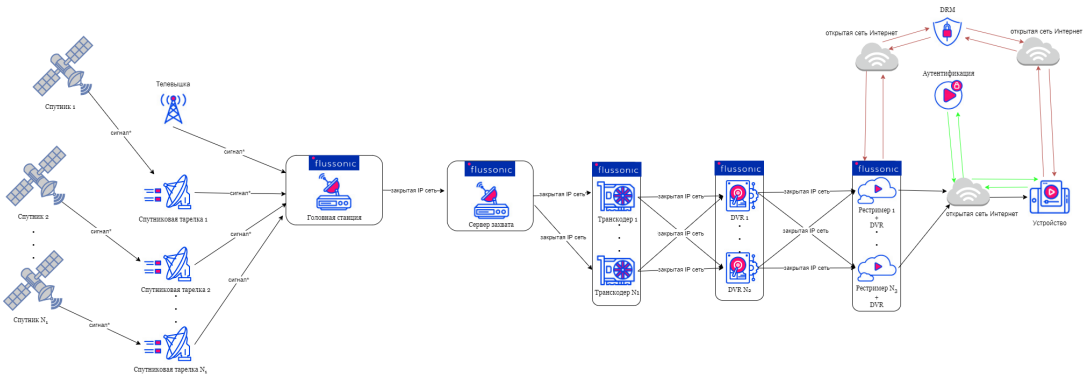


Figure 3.5

адрес в ”Избранные контакты”.

Email: support@flussonic.com

Тел.: +7 (800) 777-84-13, +7 (495) 481-37-63

3.2 Internet streaming

3.2.1 Организация CDN

Из этой статьи вы узнаете, как оптимизировать доставку видеоконтента на другие континенты. Видеоконтент — это и живое видео, ретранслируемое с сервера захвата, и видео, хранящееся на диске origin-сервера (например, DVR или VOD).

При передаче такого контента на большие расстояния по публичному интернету могут возникнуть следующие проблемы:

- **Нестабильность**. Сигналу нужно пройти много маршрутизаторов, чтобы попасть с сервера захвата/origin-сервера к конечному пользователю, при этом стабильность передачи данных никем не гарантируется. Могут пройти десятки секунд, прежде чем пользователь получит запрошенное видео.
- **Стоимость**. Каналы между континентами, странами, регионами обычно дороже местных/городских каналов, поэтому обращение отдаленных пользователей напрямую к серверам захвата и origin-серверам нерентабельно.
- **Нагрузка**. Одновременный запрос видео большим количеством пользователей может вызвать перегрузку сервера.

Для решения этих проблем обычно применяют **CDN** (Content Delivery Network) — сеть доставки контента. Это множество серверов со специализированным ПО, физически расположенных в тех регионах, где пользователи будут получать контент. На местных серверах CDN можно либо полностью дублировать, либо кешировать наиболее часто запрашиваемое содержимое DVR, а также распределять между этими серверами запросы на проигрывание живого видео. Это значительно ускоряет доставку и отдачу контента конечному пользователю, а также позволяет снизить затраты на каналы связи и оптимизировать нагрузку на сервера.

Flussonic Media Server обладает рядом возможностей, позволяющих без труда организовать CDN необходимого масштаба для доставки видео. Использование *Flussonic Media Server* как специализированного ПО на серверах CDN дает ряд преимуществ, например:

- Встроенный механизм кеширования, специализированный под задачи доставки видео с персонализированной рекламой (см. Монетизация контента через врезку рекламы).
- Отсутствие необходимости отправлять на edge-сервера сразу несколько протоколов по дорогим каналам, ведь *Flussonic* на edge-сервере сам упакует видео в нужный формат.

В данной статье рассмотрим пример построения небольшой сети CDN из серверов, вещающих прямые эфиры.

Далее будут приведены конфигурации всех серверов в пайплайне видео от захвата до edge-серверов (CDN). Если вас интересуют только сервера CDN, переходите в раздел Раздача.

Пайплайн

Схема организации пайплайна:

* В регионе захвата будет минимум два резервирующих друг друга сервера. * В регионе вещания серверы будут захватывать видео с одного из двух источников. * Каждый канал будет передаваться между регионами только один раз, чтобы не создавать лишний трафик. * В регионе захвата будет вестись транскодирование и запись, чтобы видео не потерялось при провалах канала.

В рассматриваемых далее примерах каждый сервер будет выполнять определенную роль — захват и транскодирование, запись, раздача. Вы можете объединять сервера в регионе захвата произвольным образом в зависимости от доступных аппаратных ресурсов, например выполнять захват, транскодирование и запись на одном и том же сервере. На схеме ниже показаны потоки видео в предлагаемой системе, пунктирными стрелками обозначены резервные каналы.

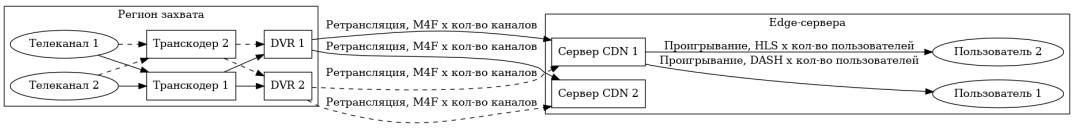


Figure 3.6

На такой схеме продемонстрируем возможности *Flussonic Media Server*.

См. также подробную статью в нашем блоге о пайплайне видео.

Захват и транскодирование

Можно сделать разные конфигурации захвата потоков в сеть доставки в зависимости от того, можно ли брать видео из источника несколько раз или нет. В самом простом варианте, если видео приходит с головной станции мультикастом по UDP, и можно просто на разных серверах захвата настроить захват одного и того же видео (см. Прием мультикаста). Мы же рассмотрим пример с кластерным захватом, который подойдет для захвата из источника с дорогим/медленным каналом.

Чтобы клиенты с разной пропускной способностью каналов могли стабильно получать видео, нужно транскодировать его в мультибитрейтный поток. Тогда пользователь может выбрать тот битрейт (качество), который будет для него комфортен. Вы можете выполнять транскодирование на тех же серверах, где происходит захват, или выделить для этого отдельный пул серверов. Все зависит о доступных ресурсов, ведь транскодирование — очень ресурсоемкий процесс. В нашем примере транскодирование будет происходить на тех же серверах, что и захват.

Рассмотрим пул из двух транскодеров (`transcoder1.example.com` и `transcoder2.example.com`), которые резервируют друг друга посредством кластерного захвата.

Транскодер 1

```
cluster_key mysecretkey;
```

```
# Remote sources:
peer transcoder1.example.com {}
peer transcoder2.example.com {}

# Ingest streams:
stream tvchannel1 {
  input udp://239.0.1.1:5500;
  transcoder vb=2048k preset=veryfast ab=128k vb=1024k preset=veryfast ab
   =64k;
  cluster_ingest;
}
stream tvchannel2 {
  input udp://239.0.1.2:5500;
  transcoder vb=2048k preset=veryfast ab=128k vb=1024k preset=veryfast ab
   =64k;
  cluster_ingest;
}
```

Транскодер 2

```
cluster_key mysecretkey;

# Remote sources:
peer transcoder1.example.com {}
peer transcoder2.example.com {}

# Ingest streams:
stream tvchannel1 {
  input udp://239.0.1.1:5500;
  transcoder vb=2048k preset=veryfast ab=128k;
  cluster_ingest;
}
stream tvchannel2 {
  input udp://239.0.1.2:5500;
  transcoder vb=2048k preset=veryfast ab=128k;
  cluster_ingest;
}
```

Здесь и далее мы подразумеваем, что у серверов правильные хостнеймы и все они доступны.

На всех серверах должен быть прописан единый ключ кластера. Здесь мы выбрали `mysecretkey`, но его можно поменять.

Обратите внимание, что конфигурация обоих серверов абсолютно идентична. Это обязательное требование при кластерном захвате: все потоки должны быть объявлены на всех серверах.

Запись архива

Серверы, на которых будет храниться архив — это origin-серверы (`origin1.example.com` и `origin2.example.com`). В нашем примере они будут выполнять ретрансляцию потоков от транскодера к edge-серверам CDN, одновременно записывая идентичный архив, чтобы в случае отказа одного из серверов сохранилась резервная копия на втором.

DVR 1

```
cluster_key mysecretkey;

# Remote sources:
source transcoder1.example.com {
  dvr dvr/origin1 2d;
}
source transcoder2.example.com {
  dvr dvr/origin1 2d;
}
```

DVR 2

```
cluster_key mysecretkey;

# Remote sources:
source origin1.example.com {
}
source transcoder1.example.com {
  dvr dvr/origin2 2d;
}
source transcoder2.example.com {
  dvr dvr/origin2 2d;
}
```

При такой конфигурации Flussonic Media Server будет забирать все каналы с одного или со второго транскодера и писать их локально в архив.

Раздача

Сервера в CDN будут выступать в роли рестримеров, ретранслируя все потоки с предыдущего звена пайплайна (в нашем случае серверов DVR) и кешируя недавние запросы к DVR:

Сервер CDN 1

```
cluster_key mysecretkey;

source origin1.example.com {
  cache /storage/cache 2d;
}

source origin2.example.com {
```

```
  cache /storage/cache 2d;
}
```

Сервер CDN 2

```
cluster_key mysecretkey;

source origin1.example.com {
  cache /storage/cache 2d;
}

source origin2.example.com {
  cache /storage/cache 2d;
}
```

Реализуемый Flussonic Media Server механизм рестриминга гарантирует, что каждый поток будет передаваться на каждый edge-сервер только один раз.

Запросы к архиву будут прозрачно проксироваться на соответствующий сервер DVR, и полученное видео будет кешироваться на edge-сервере. Благодаря кешу последующие запросы к тому же содержимому будут выполняться быстрее. Подробнее о кешировании читайте здесь. См. также Кластеризация DVR об особенностях доступа к архиву в распределенной среде видеодоставки.

Поскольку CDN — последний узел пайплайна перед отдачей потока клиентам, здесь можно добавить авторизацию и/или определить список стран, из которых разрешен просмотр.

Масштабирование CDN При раздаче большого количества видео у вас скорее всего будет несколько серверов CDN в каждом регионе, и нужно будет решать вопрос с распределением нагрузки. Для этого можно настроить балансировщик нагрузки для каждого из регионов, например по такой схеме:

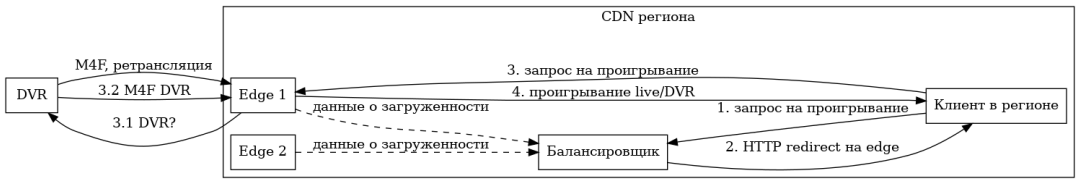


Figure 3.7

На схеме показано, что с сервера DVR идет ретрансляция живого видео, а когда пользователь обращается к незакешированному архиву, то отправляется и архив. Клиент получает все видео с edge-серверов и не знает о существовании DVR и других звеньев пайплайна, а балансировщик только выполняет редиректы, т.е. видео через него не передается.

3.2.2 Руководство по созданию UGC-платформы или сервиса на базе Flussonic

Из этого документа вы узнаете, как спроектировать и внедрить UGC-платформу или сервис для стриминга с помощью *Flussonic Media Server*. *Flussonic Media Server* — многофункциональный медиасервер для создания высоконагруженных проектов по стримингу видео любого масштаба. Этот документ предназначен для:

- архитекторов ПО,
- технических директоров,
- продуктовых или проектных менеджеров

компаний, которые хотят построить собственную стриминговую UGC-платформу. В этом документе не содержится информация об установке или настройке *Flussonic Media Server*. Информацию об установке *Flussonic Media Server* и начале работы с ним читайте в разделах Установка и Быстрый старт документации.

Раздел Введение фокусируется на задачах, решаемых *Flussonic Media Server* для создания стриминговой UGC-платформы, и сферах применения стриминговой UGC-платформы. В разделе Типовое решение для стриминговой UGC-платформы рассматриваются архитектура UGC-платформы, её составляющие, процесс доставка контента до зрителя и методы резервирования и масштабирования для обеспечения надёжности сервиса. Раздел Анализ требований посвящен изучению требований для сервиса или платформы. В разделе Проектирование архитектуры показано как считать пропускную способность сети и что требуется для построения схемы архитектуры сети. Пример реализации решения на базе *Flussonic Media Server* приведён в разделе Пример разработки стратегии внедрения UGC-сервиса.

Содержание

Введение

В понятие UGC входят:

- трансляции с ультранизкой задержкой менее одной секунды, например, групповые разговоры в Google Meets или Zoom,
- трансляции в соцсети с задержкой в пределах нескольких секунд.

Видеостриминговые UGC-платформы используются для потокового вещания разного контента: от прямых трансляций прохождения видеоигр до деловых мероприятий и лекций.

Диапазон сфер применения UGC-платформ ограничивается лишь доступностью Интернета. Они используются в таких сферах, как:

- **Гейминг**. Для вещания прямых трансляций турниров и соревнований по киберспорту, прохождений и обзоров видеоигр.
- **Спорт**. Для организации прямых трансляций и записи турниров, соревнований по разным видам спорта.
- **Обучение**. Для вещания прямых трансляций и записи лекций, семинаров, вебинаров, записи курсов и специализаций.
- **Религия**. Для вещания прямых трансляций и записи богослужений.
- **Мероприятия**. Для вещания презентаций новой модели смартфона, трансляций культурно-массовых мероприятий.

Работа над проектом начинается с определения конечной цели и задач.

Цель проекта — конечный результат, который вы хотите получить, выгодный бизнесу; то, ради чего будет создаваться проект. Пример цели — увеличить доход компании от продаж или снизить издержки. **Задачи** проекта — действия, которые необходимо предпринять, чтобы достичь цели.

Flussonic Media Server позволяет решать следующие задачи при создании UGC-сервиса или платформы:

- Приём публикаций от авторов контента с максимальным качеством и стабильностью.
- Обеспечение использования серверного оборудования при неравномерной нагрузке в зависимости времени суток.
- Обеспечение равномерного распределения сетевого трафика между серверами.
- Взимание денег с авторов за сервис по ретрансляции и хранению записей.
- Запись, хранение и проигрывание зрителям архива трансляции.
- Ретрансляция контента в социальные сети.
- Обеспечение комфортной среды для общения группы людей в Интернете в формате видео- и аудиоконференций без установки дополнительных приложений.

- Взимание денег со зрителей, уплата авторам роялти (royalty).

Определение целей и задач позволяют понять ожидаемый результат и начать проектировать архитектуру решения.

Типовое решение для стриминговой UGC-платформы

В этой части вы узнаете:

- Из каких элементов строится стриминговая UGC-платформа и зачем они нужны,
- Как устроен процесс доставки потока от автора к зрителю,
- Как обеспечить отказоустойчивость и масштабируемость всей системы.

Ниже представлена схема типового решения для стриминговой UGC-платформы, построенной на *Flussonic Media Server*:

Схема 1. Схема типового решения для стриминговой UGC-платформы

где:

- Publish — сервер публикации,
- Transcoder — сервер транскодирования,
- DVR — DVR-сервер,
- Egress — egress-сервер,
- Central — управляющий сервер *Flussonic*,
- Content authors — устройства авторов контента,
- Viewers — устройства зрителей.

Составляющие Основные элементы, из которых состоит стриминговый UGC-сервис:

- Сервер публикации — сервер, который принимает поток от автора.
- Сервер транскодирования — сервер, который преобразовывает входящий поток в поток с несколькими видеодорожками разного разрешения и битрейта. Необязательный компонент.
- DVR-сервер — сервер, который принимает входящие потоки, записывает и хранит их, а затем отправляет по запросу. Необязательный компонент.
- Egress-сервер — сервер, раздающий контент.
- Central — управляющий сервер, обеспечивающий единую точку доступа для управления конфигурацией серверов. Необязательный компонент.

Сервер публикации Сервер публикации позволяет решать следующие задачи:

- Приём публикации от авторов контента с максимально возможным качеством и стабильностью.
- Ретрансляция контента в социальные сети.

Flussonic принимает публикации потоков в режиме реального времени по следующим протоколам:

- WebRTC
- SRT
- RTMP

Сервер публикации переупаковывает входящий поток во внутренний *Flussonic*-протокол M4S* и отправляет его одному из следующих серверов:

- серверу транскодирования,
- DVR-серверу,
- egress-серверу.

Это зависит от выстроенной архитектуры сервиса и поставленных задач.

> M4S — протокол потоковой передачи в реальном времени между серверами *Flussonic*, обеспечивающий низкую задержку при передаче данных. Этот протокол поддерживает все кодеки *Flussonic*.

Читайте также:

- Публикация потока из OBS Studio в Flussonic Media Server.

Сервер транскодирования Сервер транскодирования решает задачу подготовки контента для доставки зрителю с максимально возможным качеством.

Чтобы решить эту задачу, требуется следующее:

- Подготовить мультибитрейтный поток.
- Балансировать нагрузку серверов транскодирования.
- Мониторить объём транскодированного трафика.

Сервер транскодирования принимает поток и транскодирует его, получая на выходе мультибитрейтный поток.

Подготовка мультибитрейтного потока Подготовка контента для доставки предполагает формирование мультибитрейтных потоков — синхронизированных видеодорожек с разным разрешением и битрейтом. Чтобы транскодер качественно подготовил поток для всех зрителей, входной поток должен быть с максимальными настройками битрейта.

Ниже приведены оптимальные профили для доставки контента зрителям:

- 1920x1080p,
- 1280x720p,
- 896x504p.

Выбор профиля для доставки контента зависит также от конечного устройства зрителя. Так для большинства смартфонов оптимальным профилем качества является 896x504p, а для ТВ — 1920x1080p. Транскодер может как понизить, так и повысить разрешение потока для доставки зрителям. Например, автор публикует поток в разрешении 720p, а транскодер увеличивает разрешение до 1080p для доставки зрителям.

Чем выше разрешение и битрейт потока, тем больше задержка сигнала.

Если вы получаете потоки из разных источников сразу, например, браузера, веб-камеры, OBS или видеоэнкодера по протоколам WebRTC, RTMP и SRT, то транскодируйте эти потоки, а не переупаковывайте их. Видеостриминговые протоколы WebRTC, RTMP и SRT поддерживают разные и не до конца совместимые кодеки.

Транскодер вносит следующие задержки:

- От трёх до десяти секунд, чтобы добиться максимального качества при минимальном битрейте.
- От одной до двух секунд, чтобы добиться приемлемого качества с сильно разными битрейтами.

Балансировка нагрузки серверов транскодирования Чтобы балансировать нагрузку на серверы транскодирования и предотвратить остановку сервиса, используйте внешнее управление потоками через механизм config_external. Это исключит ручное управление конфигурацией потоков.

Мониторинг объёма транскодированного трафика Объём транскодированного трафика — это сумма битрейтов дорожек в профиле. Имея это значение, можно определить, во сколько вам обойдётся транскодирование для каждого автора.

В зависимости от битрейта исходного потока можно использовать комбинацию профилей. Например, если исходный поток 1920x1080p имеет битрейт 5 Мбит/с, то рекомендуемыми профилями являются 1920x1080p при 4 Мбит/с, 1280x720p при 2 Мбит/с и 896x504p при 900 Кбит/с. Тогда объемы транскодированного трафика составляют:

- Для 1920x1080p: 4000 Кбит/с + 192 Кбит/с = 4192 Кбит/с
- Для 1280x720p: 2000 Мбит/с + 128 Кбит/с = 2128 Кбит/с
- Для 896x504p: 900 Кбит/с + 96 Кбит/с = 996 Кбит/с

Для транскодирования используются специализированные выделенные устройства, центральный процессор или видеокарта: внешняя или встроенная в процессор. В *Flussonic Media Server* есть встроенный транскодер. Он поддерживает транскодирование на графическом процессоре GPU и с использованием центрального процессора CPU. Для транскодирования вы можете использовать медиасервер или Flussonic Coder.

Транскодирование — вычислительно дорогой процесс, создающий большую нагрузку на CPU и GPU. Изменение кодека, разрешения или битрейта видео сопровождается высоким потреблением ресурсов системы, в отличие от трансмуксинга, или пакетайзинга. Для трансмуксинга не требуется транскодер.

Транскодирование и трансмуксинг работают на разных уровнях: транскодирование — на уровне контента и данных, трансмуксинг — на уровне контейнера и протокола доставки.

> Если вам необходима консультация и профессиональная помощь в подборе оборудования для транскодирования, обратитесь в нашу службу технической поддержки support@flussonic.com.

Подробнее о транскодировании, поддерживаемых кодеках и протоколах см.:

- Транскодирование,
- Поддерживаемые протоколы и кодеки.

DVR-сервер DVR-сервер решает задачу записи и хранения архива трансляций.

С использованием DVR-сервера у ваших зрителей появляется возможность ставить на паузу, перематывать назад и вперёд, пересматривать и смотреть прямой эфир в записи.

Flussonic Media Server предоставляет инструменты для работы с архивом, например:

- live-to-VOD,
- catch-up,
- таймшифт и др.

Flussonic Media Server умеет работать как с локальными, так и с облачными хранилищами. Он может загружать записи в облачное хранилище и выгружать их. В качестве места для выгрузки архива можно указать как каталог на сервере *Flussonic*, так и, например, бакет Amazon S3. Кроме того, *Flussonic Media Server* позволяет экспортировать фрагменты архива в формате MP4-файлов. Подробнее об экспорте фрагментов архива в формате MP4-файлов см.:

- Экспорт фрагмента записи архива в виде MP4-файла,
- Скачивание фрагмента записи архива в виде файла MP4 или MPEG-TS.

Чтобы снизить нагрузку на хранилище и ускорить раздачу VOD-контента, настройте кэширование на SSD.

Подробнее о DVR и его возможностях см. DVR.

> Если вам необходима квалифицированная помощь по организации распределенного DVR-сервиса, обратитесь в нашу службу технической поддержки support@flussonic.com.

Egress-сервер Egress-сервер позволяет решать задачи:

- Обеспечения комфортной среды для общения людей в Интернете в формате видео- и аудиоконференций в режиме реального времени.
- Проигрывания зрителям архива трансляции.

Задержка видео или аудио при трансляции в режиме реального времени критична. У зрителей могут возникнуть проблемы со звуком и видео, что оставит их недовольными качеством сервиса. Чтобы обеспечить комфортную среду для онлайн-общения по видео и аудио в режиме реального времени, используйте WebRTC. WebRTC позволяет:

- Создать ощущение живого общения между участниками беседы за счёт задержки в пределах одной секунды.
- Автоматически подстраивать качество видеоизображения под скорость интернет-соединения благодаря алгоритму адаптивного битрейта (ABR). Это даёт возможность расширить аудиторию приватных чатов и смотреть трансляцию зрителям с нестабильным или медленным интернет-соединением.
- Смотреть трансляцию на любом устройстве.
- Не использовать дополнительное ПО. Достаточно браузера и выхода в Интернет.

Трансляции с задержкой, которой достаточно, чтобы отреагировать на чат, но недостаточно, чтобы общаться со зрителями в прямом эфире, встречаются чаще всего. В таких случаях небольшая задержка некритична, поэтому используйте протоколы HLS, DASH и MSS для доставки потоков зрителям. Это даёт возможность:

- Смотреть трансляции на любых устройствах.
- Расширить аудиторию, которой не нужна ультранизкая задержка.

- Увеличить эффективность воронки продаж.

Egress-сервер забирает потоки с одного из следующих серверов:

- сервера транскодирования,
- DVR-сервера,
- сервера публикации.

Затем egress-сервер доставляет контент зрителям напрямую либо через сторонний сервис CDN. Это зависит от поставленных задач и выстроенной архитектуры сети.

Подробнее см. Отправка потоков на другие серверы.

Чтобы показывать зрителям контент, вы можете использовать веб-плеер *Flussonic* и вставить его на свой сайт. Чтобы интегрировать плеер с вашим веб-сайтом, используйте Flussonic Streaming API.

Central *Central* помогает решать такие задачи, как:

- Обеспечение использования серверного оборудования при неравномерной нагрузке в зависимости времени суток.
- Обеспечение равномерного распределения сетевого трафика между серверами.

Central выполняет роль управляющего сервера, обеспечивая единую точку доступа для:

- управления конфигурацией серверов в процессе доставки контента,
- аутентификации и авторизации авторов и зрителей.

Central позволяет сделать доступной всю конфигурацию со всех серверов в процессе доставки видео. Не требуется настраивать конфигурацию для каждого сервера вручную. Взаимодействие между *Central* и другими серверами осуществляется с помощью внутреннего механизма *Flussonic* — config_external.

Central также отвечает за балансировку нагрузки между серверами в кластере, перенаправляя запросы на наименее загруженные серверы.

Flussonic предоставляет API для взаимодействия с *Central* — Flussonic Central API.

Авторизацию авторов можно настроить напрямую во *Flussonic* либо с помощью внешнего бэкенда, к которому будет обращаться сервер *Flussonic*.

Если у вас нет собственного бэкенда, который занимается авторизацией авторов и зрителей, или не хочется его создавать, воспользуйтесь следующими встроенными инструментами *Flussonic*:

- Авторизация по паролю при публикации потока: система проверяет пароль при публикации потока.
- ‘passphrase‘ для защиты публикации по SRT: система проверяет пароль при публикации потока по SRT.
- Конфигуратор бэкендов: с его помощью можно создавать авторизационные бэкенды в файле конфигурации без написания скриптов.
- ‘passphrase‘ для защиты проигрывания по SRT: система проверяет пароль при проигрывании потока по SRT.

Если у вас есть собственный бэкенд для авторизации зрителей, то *Flussonic* может обращаться к этому бэкенду для авторизации зрителей. *Flussonic* предоставляет следующие способы авторизации зрителей через бэкенд:

- Авторизация по токену: система обращается к указанному бэкенду и проверяет полученный токен.
- Авторизация сессий проигрывания через внешний бэкенд: система обращается к указанному бэкенду, чтобы получить разрешение или запрет на проигрывание контента зрителем.
- Авторизации сессий публикации через внешний бэкенд: система обращается к указанному бэкенду, чтобы получить разрешение или запрет на публикацию контента автором.

Flussonic также предоставляет API для работы с авторизационным бэкендом — Flussonic Authorization Backend API.

Для вашего удобства мы свели всю информацию о составляющих и их функциях в Табл.1:

Табл.1. Составляющие процесса доставки потоков и их функции

Составляющая	
Сервер публикации	- при
Сервер транскодирования	- изменение
DVR-сервер	- запись и хранение потоков,- возможность работать с архивом и использовать
Egress-сервер	доставка до зрителей:- тран
Central	- управление конфигурацией серверов в

Процесс доставки контента от автора до зрителя В этой части описывается процесс работы стриминговой UGC-платформы: какие этапы проходит контент, чтобы оказаться на экране зрителя.

Для этого мы снова обратимся к схеме типового решения для стриминговой UGC-платформы:

Направление чёрных стрелок на схеме указывает направление передачи потоков, а розовых — направление отправки запросов.

1. Автор <-> Сервер публикации <-> Central Чтобы начать трансляцию, автору необходимо авторизоваться и инициировать публикацию потока на сервер публикации по SRT, WebRTC (WHIP) или RTMP в зависимости от источника. Вот как происходит публикация потока автором:

- Запрос автора уходит DNS-серверу.
- DNS-сервер перенаправляет запрос на адрес одного из доступных серверов публикации в кластере. Подробнее о DNS-балансировке см. ниже.
- Сервер публикации отправляет запрос *Central* для авторизации автора.
- Как только сервер публикации получает положительный ответ, он запрашивает конфигурацию для потока у *Central*.
- *Central* возвращает необходимую конфигурацию серверу публикации.
- Автор публикует поток.

2. Сервер публикации -> Central -> Сервер транскодирования

- Сервер публикации перепаковывает входящий поток в собственный Flussonic-to-Flussonic протокол M4S и отправляет его кластеру серверов транскодирования.
- Балансировщик нагрузки *Central* с учётом состояния и загрузки серверов направляет поток серверу с наименьшей загрузкой.
- Сервер транскодирования получает M4S-поток, распаковывает и декодирует его, получая необработанное видео и аудио.
- Транскодер преобразует поток в поток с несколькими видеодорожками, адаптированными под разные разрешения и битрейт.

Особенность *Flussonic* – переупаковка потоков. При получении потока *Flussonic Media Server* распаковывает поток и снова запаковывает. Так стабильность выходного потока повышается.

3. Сервер транскодирования -> Central <-> DVR-сервер DVR-сервер:

- Запрашивает конфигурацию потока у *Central*.
- Захватывает M4S-потоки у сервера транскодирования.
- Записывает потоки и хранит копии этих потоков.

DVR-сервер также сообщается с другим DVR-сервером в кластере для репликации архива. Таким образом архив копируется на другие серверы, обеспечивая надёжность и автовосстановление данных после сбоев. Подробнее о резервировании архива см. ниже.

4. DVR-сервер <-> Central <-> Egress-сервер <-> Зрители Перед началом просмотра трансляции зрители проходят аутентификацию и авторизацию.

- Зритель заходит на сайт со своего устройства и вводит свои имя пользователя и пароль.
- Запрос на аутентификацию отправляется серверу *Central*.
- *Central* проверяет зрителя и аутентифицирует его.
- После успешной аутентификации зритель отправляет запрос балансировщику нагрузки *Central*.
- *Central* перенаправляет запрос на минимально загруженный и доступный egress-сервер в кластере.
- Egress-сервер посылает запрос серверу *Central* для авторизации зрителя.
- *Central* проверяет права зрителя и либо разрешает, либо запрещает ему проигрывать поток.
- На основе полученного ответа egress-сервер либо отправляет поток зрителю, либо нет.

В зависимости от того, прямой это эфир или запись, доставка потоков до зрителей может осуществляться с egress-серверов одним из двух способов:

- Напрямую.
- Через CDN.

Методы резервирования и масштабирования сервиса В этом разделе приведены методы обеспечения надёжности и отказоустойчивости системы на этапе проектирования архитектуры сервиса. Надёжный сервис должен отвечать следующим требованиям:

- Продолжать отвечать на запросы клиентов, несмотря на высокую нагрузку на сервис или события технического обслуживания.
- Справляться со сбоями.
- Уметь распределять нагрузку в ответ на изменения рабочей нагрузки и количества запросов пользователей.

Кластер серверов публикации и DNS-балансировка В приведённой нами схеме используется кластер из нескольких серверов публикации с DNS-балансировкой. Такой подход позволяет:

- Распределить нагрузку на несколько серверов.
- Обеспечить автору возможность подключиться хотя бы к одному активному серверу.

Чтобы вы могли оказывать отказоустойчивый сервис при публикации потоков по протоколам SRT и RTMP, не поддерживаемым на прикладном уровне (application layer), используйте DNS-балансировку. Для балансировки публикаций по WebRTC (WHIP) используйте встроенный балансировщик *Flussonic*.

Все серверы публикации в кластере одинаковые, то есть равноправны и взаимозаменяемы, и каждый из них может принять любой поток. Если один из серверов в кластере выходит из строя, то другой сервер возьмёт на себя его работу.

При DNS-балансировке сервер, на который будет отправлен запрос, выбирается на основе определённого алгоритма, например, round robin. Узнать адрес сервера, на который необходимо направить запрос автора, можно разными способами: запросить через API (рекомендуется) или вернуть подходящий сетевыми средствами. Это зависит от структуры вашей сети.

Однако у DNS есть существенный минус — он не учитывает статус загрузки серверов в кластере. По этой причине авторы могут быть перенаправлены на сервер, который в данный момент может быть перегружен или выключен.

Нужно также учитывать, что:

- DNS кешируется до трёх дней. Если в кэше окажутся серверы, которые выключены или находятся в закрытой сети, то таймаут подключения может быть очень долгим.
- В RTMP не встроен механизм редиректов. Временные решения, сделанные для обхода, не работают. Load balancing with WHIP publications #live-webrtc-publish-load_balancing

В нашем случае необходим балансировщик, который перенаправляет запросы, а не проксирует потоки через себя. При выходе из строя одного сервера публикации запросы авторов перенаправляются на другие активные серверы.

Рассмотрим использование DNS-балансировки на примере облачной платформы, которая предоставляет UGC-сервис для реализации различных проектов.

Чтобы автор всегда подключался хотя бы к одному активному серверу, выделяется минимум две DNS-записи. Так для каждого проекта предоставляется отдельная группа серверов с учётом загрузки серверов или, например, географического расположения авторов.

Так вы получаете:

- множество DNS-записей для резервирования,
- отдельный домен для каждого проекта для более точной балансировки запросов на публикацию потоков и выделения ресурсов.

Кластер транскодеров с балансировкой нагрузки Аварийные ситуации в работе сервера транскодирования могут привести к остановке или прекращению работы вашего сервиса. Чтобы этого избежать:

- Установите как минимум два сервера транскодирования, если вам позволяет бюджет.

- Предусмотрите возможность аварийного переключения на другой сервер в случае нарушения работы одного из серверов.
- Распределите нагрузку между серверами.

В нашей схеме используется кластеризация с балансировкой нагрузки через *Central*.

Кластер из двух и более серверов транскодирования обеспечивает бесперебойную работу транскодеров. В случае отказа одного из серверов транскодирования другие серверы смогут запустить процессы на своей стороне. Таким образом, сервис будет продолжать работать. Подробнее о кластеризации см. Кластер.

Чтобы избежать перегрузки серверов транскодирования, необходимо равномерно распределять запросы к кластеру серверов транскодирования. Так *Central* выполняет балансировку нагрузки следующим образом:

- Оценивает статус загрузки серверов транскодирования в кластере.
- Направляет запросы на наименее загруженные серверы.

Подробнее о *Central* и балансировке нагрузки см. описание балансировки нагрузки в Flussonic.

Кластер DVR с репликацией Внезапное отключение, сбой системы или другая аварийная ситуация могут стать причиной потери записей архива. Чтобы обезопасить свой сервис и обеспечить надёжность архива, важно сделать резервную копию архива DVR на другие серверы. Для этого мы предлагаем использовать кластер из двух и более DVR-серверов с функцией репликации.

Репликация позволяет копировать архив на другие серверы для:

- обеспечения надёжности,
- автовосстановления после сбоев.

Архив хранится на двух или более серверах, где один является основным, а другие — вторичными. Репликация работает следующим образом:

- Основной сервер захватывает и хранит потоки из источника.
- Вторичный сервер захватывает потоки из основного сервера (см. схему).

Flussonic предоставляет и другие инструменты для резервного копирования архива DVR и обеспечения его доступности:

- кросс-репликация,
- Flussonic RAID,
- кэширование сегментов.

CDN С увеличением числа зрителей и объёма выходного трафика egress-сервер в какой-то момент перестанет справляться с доставкой потоков в одиночку. По мере расширения зоны распространения контента системе станет всё сложнее предоставлять зрителям качественный сервис.

CDN (сеть доставки контента) даёт возможность:

- Увеличить число зрителей без приобретения дополнительного оборудования.
- Равномерно распределять нагрузку между серверами.
- Обеспечить зрителям доступность контента.

Имея множество серверов, распределённых по различным географическим точкам, CDN:

- Сокращает расстояние между egress-сервером и зрителем.
- Минимизирует задержку.
- Обеспечивает быстрый доступ к контенту.
- Разгружает egress-сервер, поддерживая его работоспособность.

Чтобы обеспечить эффективную доставку контента по всему миру, CDN использует следующие методы:

- Кэширует контент в различных PoP (point of presence, точка присутствия).
- Распределяет рабочую нагрузку на несколько серверов.

Вы можете построить свой CDN или использовать существующие решения, например, Akamai, CDNvideo и др. Это зависит от числа зрителей, их географического положения, выделенного бюджета и др.

Анализ требований

Чтобы определить требования:

- Определитесь с нишей.
- Изучите целевую аудиторию.
- Определите контент и его авторов.
- Определите основные возможности платформы или сервиса, изучив представленные на рынке.

Определите нишу Выберите нишу, в которой будет работать сервис. Будет ли это здоровье и фитнес, онлайн-обучение, искусство или игры? Будет сервис или платформа охватывать только один вариант использования или несколько?

Определение ниши очень важно для определения задач стримингового UGC-сервиса или платформы, изучения целевой аудитории, выработки стратегии развития и дальнейшей работы. Изучите рынок, посмотрите что предлагают конкуренты.

Изучите целевую аудиторию Определите, кто ваша целевая аудитория и что она хочет. Ответьте на такие вопросы, как:

- Что объединяет вашу аудиторию?
- С какой целью аудитория смотрит контент (образование, развлечения, спорт и др.)?
- Какие устройства используют зрители для просмотра контента (ПК, смартфон на базе Android или iOS и др.)?
- Какое примерное число одновременных подключений зрителей должен выдерживать ваш сервис или платформа?

Это даст вам понимание вашей аудитории и того, что она ожидает от вашего сервиса или платформы.

Например, целевой аудитории игровой индустрии интересны: видеоигры, их обзоры и прохождения, киберсоревнования и др. Зрители обычно смотрят такой контент на различных устройствах: ПК, ноутбуках, смартфонах (Android, iOS), планшетах. Это пример развлекательного контента, который смотрят сотни тысяч зрителей.

Определите контент и профили авторов Подумайте, какой вид контента вы хотите предоставлять и кто будет его создавать. Определите, что нужно целевой аудитории и какие у неё предпочтения, чтобы понять какой контент точно будут смотреть.

Рассмотрим, например, видеостриминговый сервис *Twitch*. *Twitch* был создан для геймеров и ориентирован на них. Поэтому основную часть контента сервиса составляют прямые трансляции прохождения видеоигр и геймплея, а также киберспортивные турниры. Такой контент создают геймеры и организаторы турниров.

При определении контента для платформы или сервиса необходимо исходить из целевой аудитории и её предпочтений.

Определите основные возможности платформы или сервиса При выборе стриминговой UGC-платформы или сервиса авторы и зрители обращают внимание на возможности, которые предоставляет платформа. Jпределитt набор функций, которые отвечают потребностям и запросам авторов и зрителей. Посмотрите, что предлагают конкуренты.

Ниже перечислены некоторые возможности стриминговых UGC-платформ:

- авторам: проведение трансляций с помощью разных устройств, таких как VMix, OBS, Atemi, HDMI-RTMP конвертер, пульт видеомонтажа и браузеров;
- зрителям: просмотр контента с любого устройства, например, ПК, смартфон на базе Android или iOS, ноутбук;
- запись, хранение и проигрывание онлайн-трансляций;
- одновременная трансляция в социальные сети и на другие платформы, например, YouTube, VK, Одноклассники, RUTUBE, Twitch;
- непрерывное и стабильное проигрывание контента;
- загрузка предварительно записанных потоков;
- онлайн-трансляции с минимальной задержкой: трансляция с задержкой, которой достаточно, чтобы отреагировать на чат, но недостаточно, чтобы общаться со зрителями в прямом эфире;
- встроенный медиаплеер;
- система авторизации;
- система подписок и др.

В зависимости от целей и задач платформы или сервиса, выделенного бюджета, доступных программно-аппаратных ресурсов, набор возможностей может варьироваться.

Проектирование архитектуры сети

Эта часть посвящена:

- Оценке необходимой пропускной способности сети на входе и на выходе для сервера публикации, сервера транскодирования, DVR-сервера, egress-сервера.
- Построению схемы архитектуры сети.

Чтобы посчитать необходимую пропускную способность сети на входе и на выходе и построить схему архитектуры сети, ответьте на следующие вопросы:

Табл.2. Уточняющие вопросы для определения требований к сети

Тема	
Источник (вход)	1) Какой тип источника сигнала (камера, программный энкодер, аппаратн
Транскодер	1) Какие профили выходного видеопотока (кодек, битрей
DVR архив (запись и хранение)	1) Есть ли необходимость в записи потоков? Если да, то как вы план
Egress-сервер (выход)	1) Какие типы конечных устройств (мобильные устройс
Безопасность и защита	1)
Другие требования	

Чтобы посчитать пропускную способность сети, используйте следующую формулу:

число потоков * (битрейт видео + битрейт аудио)

На основе полученных ответов на вопросы из *Табл.2*, вы можете определить число серверов публикации, серверов транскодирования, DVR-серверов, egress-серверов и построить схему.

Затем переходите к настройке и конфигурации серверов.

Пример разработки стратегии внедрения UGC-сервиса

Эта глава посвящена разработке стратегии для небольшого UGC-сервиса — платформы для трансляции мероприятий. Следуйте шагам ниже:

- Анализ требований.
- Проектирование архитектуры сети.

> Эта глава не предполагает предоставление какого-либо программного кода или конфигурации *Flussonic Media Server* для этого проекта.

Анализ требований Допустим, вы хотите создать платформу для трансляции мероприятий. Цель проекта — снизить издержки компании на текущий сервис.

- Ниша: организация вещания мероприятий.
- Целевая аудитория: небольшие и крупные компании.
- Тип контента: виртуальные конференции и саммиты, вебинары, онлайн-семинары, виртуальные корпоративные мероприятия, конференции.
- Возможности сервиса:

Проектирование архитектуры сети Вы ответили на вопросы из Табл. 2 следующим образом:

Тема	
Источник (вход)	1) Тип источника: а
Транскодер	
DVR архив (запись и хранение)	1) Требуется запись и хранение архива. У зрителей будет возможность по
Egress-сервер (выход)	
Безопасность и защита	
Другие требования	

Чтобы вычислить пропускную способность сети, воспользуемся следующей формулой:

число потоков * (битрейт видео + битрейт аудио)

Для сети рассчитаем следующие параметры:

- пропускная способность сети на входе:

10 потоков * (5000 + 192)Кбит/с ≈ 52 Мбит/с

- пропускная способность сети на выходе транскодера. При транскодировании десяти Full HD (1080p, 5000 Кбит/с, H.264) потоков в три профиля:

Получаем:

30 * (4000 + 2000 + 1000 + 192 + 128 + 96)Кбит/с ≈ 223 Мбит/с

- пропускная способность сети на выходе для раздачи. Берём максимальное количество зрителей (100 тыс.) и самое высокое качество потока (Full HD):

100,000 * (4,000 + 192)Кбит/с ≈ 420 Гбит/с

На основе полученных ответов составим схему сети:

Схема 3. Схема архитектуры сети для платформы трансляции онлайн-мероприятий

Для реализации проекта понадобится четыре сервера *Flussonic*:

- Transcoding server №1 и transcoding server №2, чтобы принимать публикации и транскодировать видео.
- Egress server №1 и Egress server №2, чтобы вести запись архива и доставлять контент зрителям.

Авторизация Для авторизации зрителей в платформе используется авторизация по токену:

- Внешняя система генерирует персональный токен для каждого зрителя, чтобы у зрителя не было возможности передавать его другим.
- Затем *Flussonic* сверяет токен с IP-адресом и некоторыми другими параметрами.

Этот метод отлично подходит для закрытых мероприятий.

Серверы транскодирования В нашей схеме серверы *transcoding server №1* и *№2* занимаются приёмом публикации по RTMP и SRT и транскодированием.

Transcoding server №1 будет основным, а *transcoding server №2* — резервным. Так мы можем реализовать следующие сценарии:

- Если основной сервер будет недоступен, то произойдёт бесшовное переключение на резервный сервер.
- Если оба сервера (основной и резервный) будут недоступны, то включится файл-заглушка и в архив будет записываться заглушка. Такая логика работы обеспечивает отказоустойчивость системы в случае возникновения трудностей с серверами публикации.

Egress-серверы Egress server №1 и egress server №2 выполняют следующие задачи:

- Запись трансляций в режиме реального времени и их запись в архив.
- Доставка потоков зрителям.

Для записи и хранения архива:

- Egress server №1 и egress server №2 захватывают M4S-потоки с сервера транскодирования и записывают онлайн-трансляции.
- После окончания мероприятия *Flussonic* выгружает записи в формате MP4-файлов в облачное хранилище S3, чтобы зрители могли посмотреть трансляцию мероприятия в записи.

Чтобы ускорить доставку VOD-контента, используем кеширование на SSD. Это будет работать следующим образом:

- При первом запросе MP4-файла с записью онлайн-трансляции файл будет кеширован локально на SSD.
- Все последующие запросы будут обслуживаться уже с этого SSD, чтобы каждый раз не обращаться к основному хранилищу S3.

- Если к файлу кеша не обращаются в течение суток, то файл удаляется.
- Если суммарный объем файлов в кеше превышает 100 Гб, то *Flussonic* удаляет файлы из кеша, начиная с самого старого.

Чтобы равномерно распределить нагрузку между серверами, используем гибридную схему работы балансировщика нагрузки. При такой схеме:

- Для массовых мероприятий используется CDN Akamai с балансировщиком на стороне CDN Akamai. CDN Akamai захватывает контент из egress server №1 и egress server №2 по протоколам HLS и LL-HLS.
- Для камерных мероприятий используется балансировщик *Flussonic* с распределением между двумя egress-серверами. В таком случае egress server №1 выступает также в качестве балансировщика нагрузки.

Раздача контента осуществляется по протоколам HLS и LL-HLS:

- LL-HLS — для доставки трансляций в режиме реального времени,
- HLS — для записей трансляций.

Глоссарий

User-generated Content (UGC)

Пользовательский контент, т.е. любая форма контента, например, видео, аудио или фото, созданная людьми и опубликованная в Интернете.

UGC-платформа

SaaS (*англ.* software-as-a-service — программное обеспечение как услуга), которое используется для сбора и управления пользовательским контентом. UGC-платформа соединяет автора и зрителя через контент. Особенность UGC-платформы состоит в том, что она использует авторов для производства контента, в отличие от IPTV/OTT-платформ, которые сами являются провайдерами контента.

Сервер публикации

Сервер, который принимает поток от автора.

Сервер транскодирования

Сервер, который преобразовывает входящий поток в поток с несколькими видеодорожками разного разрешения и битрейта.

DVR-сервер

Сервер, который принимает входящие потоки, записывает и хранит их, а затем отправляет по запросу.

Egress-сервер

Сервер, раздающий контент.

Кластер

Группа серверов, работающих вместе для выполнения общих задач.

DNS-балансировка

Выделение нескольких IP-адресов на одно доменное имя, что позволяет направлять запросы авторов на разные серверы при обращении к одному и тому же домену.

3.2.3 Проигрывание VOD

Содержание:

- Как вставить плеер на сайт
- Как проиграть файл по разным протоколам
- Мониторинг проигрывания VOD-файлов
- Мультиязыковой стриминг
- Экспорт трека с субтитрами в виде SRT
- Адаптивный стриминг (мультибитрейт)
- Рестриминг VOD

Как вставить плеер на сайт

В *Flussonic Media Server* есть специальная страница — **embed.html**, с помощью которой можно вставить VOD видео на сайт или просмотреть его через браузер. Она доступна по ссылке:

```
http://FLUSSONIC-IP/myvod/bunny.mp4/embed.html
```

Страница автоматически определяет браузер и выбирает поддерживаемый протокол. Для большинства устройств на сегодня — HLS.

Подробнее в статье Вставка видео на сайт (embed.html).

Как проиграть файл по разным протоколам

Здесь мы покажем, как проиграть файл по различным видео-протоколам. Список всех поддерживаемых протоколов вместе с URL для проигрывания вы можете увидеть в веб-интерфейсе. Здесь же можно воспроизвести VOD-файл. Перейдите в **Media** -> **VODs** -> ваш VOD -> **browse** и выберите файл. Справа отобразится плеер и список адресов для проигрывания:

> Обратите внимание, что веб-интерфейс может воспроизводить только файлы, находящиеся в VOD-локациях.

Также можно сформировать ссылку для проигрывания вручную. Рассмотрим пример, в котором проиграем файл `/movies/example/s01e02.mp4`. Предварительно мы настроили VOD-локацию:

```
vod myvod {
  storage /movies;
}
```

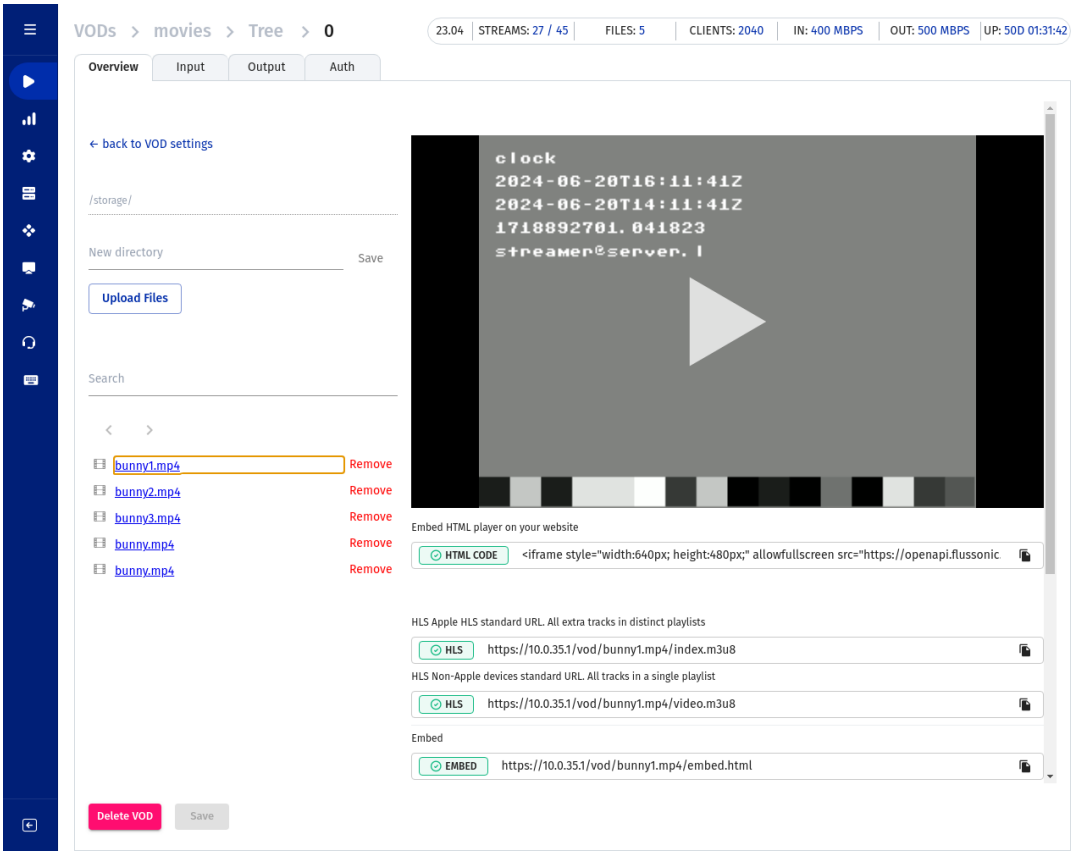


Figure 3.8: Flussonic VOD

Для того чтобы проиграть файл, лежащий на диске по пути `/movies/example/s01e02.mp4` надо указать следующие источники для плееров:

- **HLS (iOS, Android, STB)**

```
http://FLUSSONIC-IP:80/myvod/example/s01e02.mp4/index.m3u8
```

- **MSS**

```
http://FLUSSONIC-IP:80/myvod/example/s01e02.mp4.isml/manifest
```

- **DASH**

```
https://FLUSSONIC-IP:80/myvod/example/s01e02.mp4/index.mpd
```

- **RTSP**

```
rtsp://FLUSSONIC-IP:80/myvod/example/s01e02.mp4
```

Чтобы использовать RTSP, задайте RTSP-порт в разделе **Config** -> **Settings** -> **Protocols**.

Мониторинг проигрывания VOD-файлов

На вкладке **Media** — **VODs** всегда отображается статистика по файлам, которые открыты на данный момент. Также количество открытых файлов указано на информационной панели в правом верхнем углу экрана.

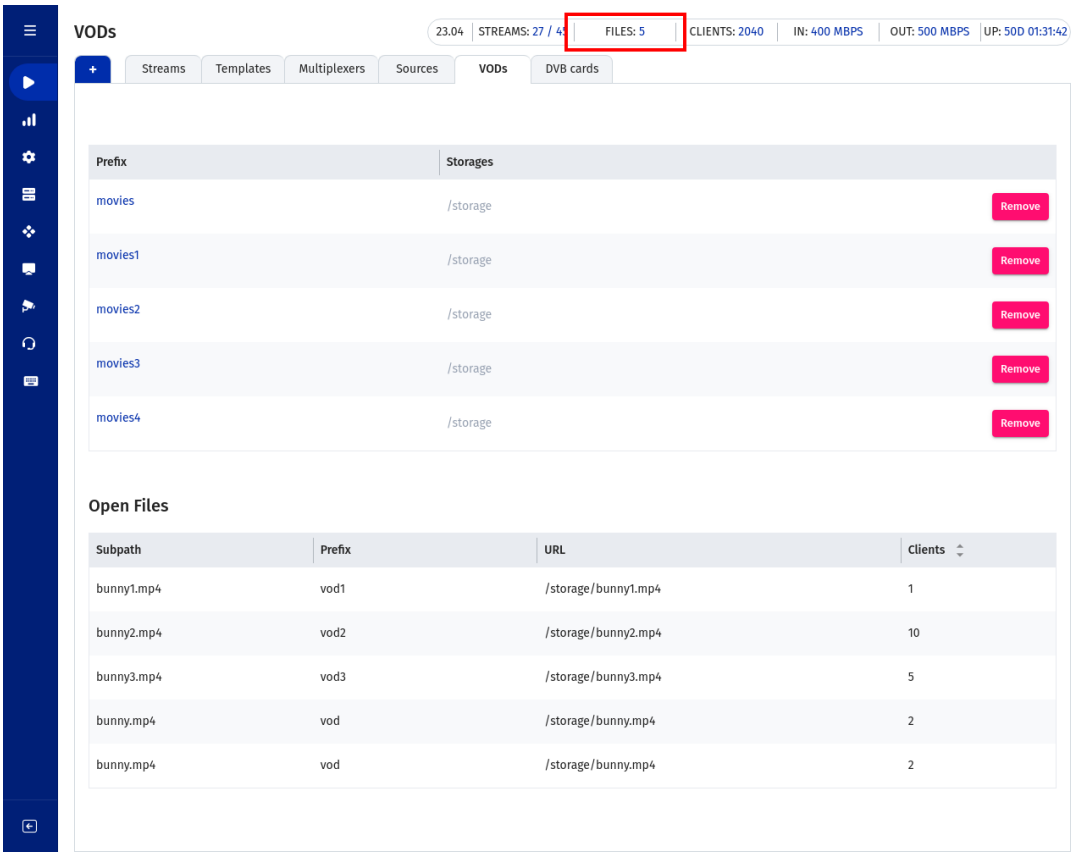


Figure 3.9: Flussonic VOD

Мультиязыковой стриминг

Протокол HLS даёт возможность переключать языки. *Flussonic Media Server* включит эту опцию автоматически, если вы просто добавить дополнительные языковые дорожки в mp4 файл.

Для включения субтитров, надо также просто добавить субтитры в формате tx3g в виде дорожек в MP4 файл.

Экспорт трека с субтитрами в виде SRT

Flussonic Media Server может отдать дорожку с субтитрами в формате SRT (SubRip Text), необходимом для некоторых flash-плееров. Получить такую дорожку можно с помощью протокола HTTP:

```
http://FLUSSONIC-IP:80/myvod/video.mp4/track-t1.srt
```

Адаптивный стриминг (мультибитрейт)

Для того, чтобы обеспечить комфортный просмотр пользователям, подключенным на разных скоростях к интернету, можно воспользоваться *адаптивным стримингом.* Flussonic поддерживает два способа:

- Использование нескольких файлов с одинаковым содержимым, но с разными качеством.
- Использование одного файла, содержащего дорожки разного качества.

Рестриминг VOD

Библиотека VOD имеет большой объём, и её копирование между серверами дорого по ресурсам, но Flussonic может выполнять рестриминг видеофайлов на соседние серверы Flussonic. Это сэкономит не только время, но и место, требуемое для хранения VOD контента. Сэкономленные ресурсы можно использовать для включения кэширования VOD контента, что увеличит производительность VOD-рестримера.

Пример конфигурации VOD-каталога:

- основной сервер VOD:

```
vod source_vod {
  storage /storage;
  download;
}
```

- рестример VOD:

```
vod restream_vod {
  storage http://FLUSSONIC-IP:8081/source_vod;
  cache /mount/cache 500G misses=2;
}
```

3.3 API

3.3.1 Список API Flussonic

Добро пожаловать в раздел API references!

В этом разделе перечислены все ссылки на API Reference, доступные в настоящее время для технологий *Flussonic*.

Доступны следующие публичные справочники API:

- API Flussonic Media Server
- API для проигрывания потоков
- API Flussonic Central
- API для авторизационного бэкенда
- API для config_external
- API Watcher Client
- API Watcher Admin
- API для создания собственного менеджера раскладок потоков

3.3.2 Принципы проектирования дизайна Flussonic API

Содержание:

- Общая информация по дизайну API
- Свойства API
- Аутентификация и авторизация
- Поддержка OpenAPI
- Коллекции
- Создание и обновление (upsert)
- Чтение объекта
- Удаление объектов

Общая информация по дизайну API

Мы (компания ”Эрливидео”) предоставляем клиентам разные продукты и сервисы, такие как *Media Server*, *Watcher*, *Iris*, *Retroview* и т.д.

На этой странице Вы найдёте актуальную информацию о принципах организации HTTP API к нашим системам.

Свойства API

Мы проектируем HTTP API для доступа других программ к нашим системам. Основные принципы, которые мы стараемся соблюдать:

- Ориентация на индустриальные стандарты и общепринятые практики с сохранением принципа разумности.
- Использование REST + JSON при построении API.
- Спецификация в формате OpenAPI 3.1 (бывший Swagger) для машинопригодного описания API.
- Подход API-first — первичность API по отношению к коду. Это означает, что сначала мы проектируем API, потом генерируем по нему код с помощью разработанного нами в рамках этого подхода openapi_handler, который мы предоставляем в публичное пользование.
- Строгое соответствие схемы API возвращаемым в payload данным. Никакие отсутствующие в схеме поля не могут быть получены от сервиса.

- Гибкое соответствие схемы API передаваемым в payload данным. Строгая валидация известных полей. Попытка отправить известное поле в неправильном формате приведет к ошибке. Неизвестные поля в payload будут проигнорированы.
- Удобство и простота создания как лёгких, так сложных запросов (например, простой запрос списка потоков и сложный запрос с несколькими фильтрами).
- Обеспечение единообразия терминологии между разными системами (имя сущности в одной системе должно быть одинаковым во всех системах).
- Обеспечение единообразия методов доступа к разным системам. На сервере может быть коллекция из одного бэкенда авторизации, а в сервисе хранения истории просмотров могут лежать триллионы записей о сессиях. Доступ к этим системам должен обеспечиваться единообразным API.
- Предпочтение HTTP вебсокетам. Доступ к данным по вебсокетам плохо стандартизован и трудно мониторится.
- Получение разумно ограниченного объёма данных по каждой выборке. Так клиент не будет получать огромное количество записей по умолчанию.
- Учёт работы с динамичным типом данных.
- Поддержка GET-запросов и задекларированных параметров в строке запроса (query string) для создания простых запросов доступа к большим коллекциям и отказ от сверхсложных правил парсинга строки запроса с переходом к JSON-языку запроса, передаваемому с помощью метода POST.
- Идемпотентность API, т.е. одинаковый результат при многократном повторе одних и тех же запросов.
- Обеспечение работы со сложными объектами, вложенными подобъектами и коллекциями.

Аутентификация и авторизация

Flussonic Media Server предоставляет возможность получать информацию и управлять определённой функциональностью при помощи HTTP.

Запросы на **получение информации** можно защитить с помощью директивы `view_auth user password;`, а на **модификацию состояния** *Flussonic Media Server* — с помощью директивы `edit_auth user password;` в конфигурационном файле `/etc/flussonic/flussonic.conf`.

Подробнее о настройке авторизации во *Flussonic* см. Авторизация.

Для доступа к HTTP API при настроенной авторизации необходимо указать логин и пароль в формате HTTP Basic Auth.

Так, например, для `edit_auth user password;` авторизация будет выглядеть следующим образом:

```
GET /streamer/api/v3/streams HTTP/1.1
Host: FLUSSONIC-IP
Authorization: Basic dXNlcjpwYXNzd29yZA==

HTTP 200 OK
Date: Sun, 19 Sep 2021 19:40:22 GMT
Content-Type: application/json
...
```

Можно также использовать `Bearer`-авторизацию, используя тот же логин и пароль:

```
GET /streamer/api/v3/streams HTTP/1.1
Host: FLUSSONIC-IP
Authorization: Bearer dXNlcjpwYXNzd29yZA==

HTTP 200 OK
Date: Sun, 19 Sep 2021 19:40:22 GMT
Content-Type: application/json
...
```

Для других систем авторизация может отличаться, но чаще всего используется `Bearer`-авторизация с токенами.

Поддержка OpenAPI

Все системы, поддерживающие описанные в этом документе принципы, предоставляют описание сервисов в машиночитаемом формате OpenAPI 3.1. Этот формат является развитием таких форматов описания API, как *Swagger* и *JSON Schema*.

OpenAPI предусматривает получение формализованной схемы сервиса, т.е. списка доступных методов и описание входных и выходных данных.

Эту OpenAPI-схему API медиасервера можно получить с работающего сервера *Flussonic* по следующему URL:

```
curl http://FLUSSONIC-IP:8080/streamer/api/v3/schema
```

Кроме того, доступны публичные справочники API, перечисленные на странице со списком API. По любой из приведенных там ссылок также можно получить схему, добавив в конце URL `.json`, например так:

https://flussonic.com/doc/api/reference.json

Ниже приведён пример того, как можно в несколько строчек кода с помощью *React Query* и *openapi-client-axios* получить доступ к списку потоков:

```
import React from 'react';
import { useQuery } from 'react-query'
```

```
import OpenAPIClientAxios from 'openapi-client-axios';


const api = new OpenAPIClientAxios({
  definition: 'http://localhost:8080/streamer/api/v3/schema',
  axiosConfigDefaults: {
    headers: {
      'Authorization': 'Basic dXNlcjpwYXNzd29yZA==',
    },
  },
});


function Streams() {
  const { isLoading, error, data } = useQuery({
    queryKey: "streams",
    queryFn : () => {
      return api.init()
        .then(client => client.streams_list({}))
        .then(res => res.data);
    },
    keepPreviousData: true,
    refetchInterval: refetchInterval,
  });

  if(isLoading) return <div>Loading</div>;
  return <div>
    {data.streams.map((stream) => <div key={stream.name}>{stream.name}</div
   >}
  </div>;
}
```

Основная идея этого кода в том, что библиотека прочитает схему API, получит из неё список *конечных точек*, или *эндпойнтов*, и сгенерирует из них набор функций. В данном примере `streams_list` взят именно из схемы и сгенерирован программно.

Приватное и публичное API В ответе на запрос могут быть возвращены поля, которые не описаны в схеме. Такие поля — часть **приватного** API. Игнорируйте их, например, как deprecated поля.

Мы используем приватное API, чтобы вводить экспериментальные поля, которые могут поменяться без предупреждения и обратной совместимости. Когда мы будем уверены, что экспериментальное поле работает как ожидается, вы увидите его описание в **публичном** API.

Тело запроса также может включать приватные параметры. А те поля, которые не относятся ни к приватному, ни к публичному API, Flussonic просто проигнорирует.

Устаревшие поля Некоторые поля во Flussonic API отмечены как `deprecated: true`. Это означает, что мы решили через некоторое время удалить это поле и использовать вместо него

новое поле.

Обычно для устаревших полей мы задаем параметр `x-delete-at`, в котором указываем версию Flussonic, в которой планируется удаление. Например, `x-delete-at: 23.02` означает, что мы планируем удалить это поле в версии Flussonic 23.02. Если вы используете это поле и не хотите, чтобы мы его удаляли, вы можете связаться с нами в течение оставшегося периода времени и обсудить возможность оставить это поле.

Коллекции

Практически все объекты организованы в **коллекции** (аналог SQL-таблиц). Сетевой доступ к системе на чтение данных во многом определяется способом чтения объектов из коллекции.

Для быстро работающих пользовательских интерфейсов и предсказуемо работающих программ необходимо:

- уметь получать минимально необходимый набор данных,
- иметь возможность достаточно надёжно получать все данные из коллекции.

Эти два требования немного конфликтуют друг с другом. Требование на ограничение набора данных подразумевает получение лишь **части** объектов из коллекции, например, 50 потоков из 2000, что есть на сервере.

Проблема заключается в следующем: когда клиент захочет получить **весь** список потоков, ему потребуется сделать **несколько** запросов к серверу. Есть вероятность, что при появлении новых потоков в промежутке между двумя соседними запросами, эти новые потоки **не попадут** в выборку.

> Мы не предлагаем единого подхода к фиксированию снапшотов коллекции и допускаем подобный риск потери ряда записей при постраничной выборке из динамически меняющейся коллекции.

Для предсказуемого доступа к подмножеству коллекции мы описываем и реализуем язык, определяющий следующие действия над коллекцией:

- фильтрация (аналог SQL `WHERE`),
- сортировка (аналог SQL `ORDER BY`),
- ограничение количества элементов (аналог SQL `LIMIT`),
- дополнительная фильтрация по курсору (аналог SQL `OFFSET`),
- ограничение набора полей (аналог SQL `SELECT`)

HTTP-методы доступа к коллекциям Стандартный способ передачи запроса на чтение с клиента на сервер осуществляется с помощью языка строки запроса (query string) и метода `GET`. Традиционно считается, что доступ, не предполагающий каких-либо модификаций, осуществляется с помощью GET-запроса (в т.ч. для кеширования).

> Важно отметить, что фактор кеширования сильно устарел и в оригинальном виде неактуален, однако мы его реализуем для удобного старта.

При использовании простого и даже примитивного языка строки запроса (query string) необходимо или использовать стандартный подход вида `key=value`, или использовать нестандартные разделители. В последнем случае такая строка запроса будет парситься нестандартно, что может вызвать ряд затруднений:

- Слишком сложные запросы при вложенных условиях. Написание такого рода условий в строке запроса (query string) будет выглядеть крайне громоздко и сложно, что противоречит одному из наших главных принципов — удобство и простота создания запросов.
- Неверная обработка запросов при использовании дополнительных разделителей в подмножествах. В таком случае необходимо помнить какие символы можно использовать без последствий, а какие — нет. Например, запятая (",") или дефис ("-") не используются в именах полей и не преобразуются в строке запроса. Таким образом, их можно использовать, в то время как амперсанд ("&") уже нельзя, т.к. это специальный символ для строки запроса и, если его не экранировать, то могут возникнуть проблемы при обработке запроса.

Так, например, некоторые компании используют в запросе сортировки символы - и + для указания направления сортировки. Символ + является спец. символом и при декодировании строки запроса будет расцениваться как пробел. Значит, необходимо либо экранировать его в %2B, либо воспользоваться нестандартным парсером, чтобы избежать возможных проблем с выполнением запроса. Использование нестандартных парсеров затрудняет подготовку запроса стандартными библиотеками.

> Существует ещё одна особенность, связанная с использованием символов Юникода (Unicode). У некоторых компаний аналог SQL запроса `WHERE age > 20` кодируется в строке запроса (query string) как `age>20`. Использование Unicode-символа вместо стандартного ASCII-символа > позволяет избежать экранирования для размещения в HTML, но набрать такой запрос с клавиатуры весьма сложно. Мы, в свою очередь, отказались от использования символов Юникода, так что набрать из консоли их не удастся.

Описанные выше проблемы призваны проилюстрировать сложность упаковки разнообразных условий и аналогов языка SQL в очень ограниченный язык *HTTP query string* (язык строки запроса).

Более сложный вариант — передача языка запроса в JSON-формате в теле POST-запроса. Такой подход выглядит неизбежным решением при использовании сложносоставных вложенных условий.

Формат ответа При **доступе к коллекции** возвращается JSON-объект со следующими полями:

```
{
  "ITEMS": [...],
  "next": ...,
  "prev": ...,
  "estimated_count": ...,
  "timing": ...
}
```

Вместо поля `ITEMS` подставляется имя той коллекции, которую запрашивали. Для потоков это будет `streams`, для сессий — `sessions`.

- `next` и `prev` — это значения курсоров (следующий и предыдущий соответственно).
- `estimated_count` — примерное количество элементов в коллекции. Точность этого значения не уточняется.
- `timing` — служебный объект с различными временами доступа. Не описывается, может измениться или удалиться.

Фильтрация коллекций Чтобы отфильтровать запрос, добавьте параметры в URL строку запроса.

Фильтрация по значению Запрос на **фильтрацию по значению** (например, `provider`) передается следующим образом:

```
curl http://FLUSSONIC-IP:8080/streamer/api/v3/streams?provider=Sky
```

Поддерживается запрос на **вхождение в список**:

`provider=Sky,Canal,CNN` (значения перечисляются через запятую)

Можно также указать **условие на поле вложенного объекта**:

`stats.alive=true`

Фильтрация по условию Для фильтрации можно также использовать условия непрямого сравнения. Для кодирования таких условий мы используем суффиксы типа `_lt` или `_gt`. Так условие `WHERE stats.delay < 5000` будет записано следующим образом: `stats.delay_lt=5000`.

Мы определили фиксированный набор суффиксов, обеспечивая тем самым два непересекающихся множества с имеющимся множеством имён полей во всех наших системах. Это позволяет нам предоставлять достаточный набор возможностей для создания запросов доступа.

Так, например, в одном из вариантов API было принято решение передавать запросы не по прямому сравнению с помощью `.`: `delay.lt=5000`. Однако в этом случае авторы API лишились возможности давать доступ к полям вложенных объектов.

Ниже приведён список поддерживаемых суффиксов:

Запись в строке запроса	SQL	Комментарий
`age=50`	`age = 50`	
`age_lt=50`	`age < 50`	
`age_lte=50`	`age <= 50`	
`age_gt=50`	`age > 50`	
`age_gte=50`	`age >= 50`	
`age_is=null`	`age IS NULL`	Только для сравнения с NULL.
`age_is_not=null`	`age IS NOT NULL`	Только для сравнения с NULL.
`age_like=pattern`	`age LIKE '%pattern%'`	Только для строковых параметров

Несколько указанных в query string фильтров применяются к коллекции *последовательно*, подобно запросам с использованием AND в SQL:

```
curl  http://FLUSSONIC-IP:8080/streamer/api/v3/streams?stats.bitrate_gt
   =4000&stats.clients_count_lt=10
```

Аналога OR для query string мы не предлагаем в силу сложной записи скобок.

Поля фильтрации указаны без префикса. Например, можно было бы указывать их через `filter.age=50`. Это может быть удобнее с точки зрения программирования, валидации и т.п., но неудобно для человека, который будет пользоваться таким API.

Сортировка коллекций Сортировка коллекций нужна для отображения в веб-интерфейсе *Flussonic UI* и получения предсказуемой постраничной выборки коллекции.

Параметры сортировки передаются через запятую к ключу:

`sort=stats.ts_delay,-stats.bitrate`

По умолчанию используется сортировка по возрастанию. Для смены направления укажите - перед названием поля.

Отсутствие префикса перед полями фильтрации означает, что мы не допускаем в нашем API поля `sort` в объектах.

Важно отметить, что в нашем API везде добавляется **имплицитная** сортировка. Если клиент закажет сортировку по полю `provider` (т.е. по полю, которое заведомо неуникально), то получится несколько групп объектов, которые отсортированы непонятным образом. Для этого мы дописываем в хвост запроса на сортировку поля типа `name` и `position`, применяя тем самым **имплицитную** сортировку.

Если они явно указаны в списке полей сортировки, то повторная сортировка по ним уже не производится.

Список конкретных полей для имплицитной сортировки мы не указываем, поскольку он может измениться в любой момент, если мы решим в конкретном случае пользоваться по-другому.

Ограничение количества элементов (курсоры) В этом документе неоднократно упоминалось, что HTTP API должен предоставлять доступ к коллекциям огромного размера, отдавая их по частям.

Получить первые 100 элементов из коллекции очень просто: достаточно добавить поле `limit=100` в строку запроса (query string). Вопрос в том, как получить следующиее 100 элементов.

> Классический ответ из баз данных SQL — передать поле `offset`. Однако мы от него отказались, поскольку он вычислительно дорогой (из-за алгоритма "маляра") и не имеет практической ценности. Мы говорили о том, что ряд коллекций может изменяться между соседними запросами, а значит, `offset` будет гарантированно промахиваться и отдавать все записи не последовательно, а неизвестно как, без шанса вычислить, что было потеряно.

> При постраничном доступе не требуется получать *записи со смещением 100*. Необходимо получить *следующую пачку записей*. Такой подход с курсорами совершенно непривычен для `mysql`, по причине их практического отсутствия в этой БД. Однако в более старых БД курсоры всё ещё используются.

В ответе на запрос первых 100 элементов мы отдаем поле `next` (а в следующих выборках и `prev`). Значение этого поля необходимо передать в качестве параметра `cursor=` в строке запроса и, таким образом, получить следующую выборку:

```
$ curl -sS "http://FLUSSONIC-IP:8080/streamer/api/v3/streams?select=name&
   limit=1&name_like=a&sort=name"  |jq
{
  "estimated_count": 5,
  "next": "JTI0cG9zaXRpb25fZ3Q9MiZuYW1lX2d0PWEx",
  "prev": null,
  "streams": [
    {
      "effective": {
        "name": "a1",
        "position": 2,
        "section": "stream",
        "static": true
      },
      "name": "a1"
    }
  ],
  "timing": {
    "filter": 0,
```

```
    "limit": 0,
    "load": 3,
    "select": 0,
    "sort": 0
  }
}
```

```
$ curl -sS "http://FLUSSONIC-IP:8080/streamer/api/v3/streams?select=name&
   limit=1&name_like=a&sort=name&cursor=
   JTI0cG9zaXRpb25fZ3Q9MiZuYW1lX2d0PWEx"  |jq
{
  "estimated_count": 5,
  "next": "JTI0cG9zaXRpb25fZ3Q9MyZuYW1lX2d0PWEy",
  "prev": "JTI0cG9zaXRpb25fbHQ9MyZuYW1lX2x0PWEyJiUyNHJldmVyc2VkPXRydWU=",
  "streams": [
    {
      "effective": {
        "name": "a2",
        "position": 3,
        "section": "stream",
        "static": true
      },
      "name": "a2"
    }
  ],
  "timing": {
    "filter": 0,
    "limit": 0,
    "load": 1,
    "select": 0,
    "sort": 0
  }
}
```

Последовательными запросами мы получаем **все** элементы выборки. Курсор устроен довольно просто: в нём закодированы значения сортируемых полей для последнего элемента выборки и именно по ним идет дополнительная фильтрация перед отдачей.

Ограничение набора возвращаемых полей Мы предусмотрели возможность частичного отображения полей. Суммарное количество полей в информации о потоке в медиасервере превышает 100 и, если необходимо получить только имена потоков, то запрашивать всю информацию довольно опрометчиво и нецелесообразно.

Для того, чтобы вернуть только часть полей, достаточно указать опцию `select`, например: `select=name,title`. Также можно запросить поля вложенных объектов: `select=name,stats.media_i`

Создание и обновление (upsert)

Мы приводим Flussonic Media Server API к стандарту JSON Merge Patch. Это позволит привести метод частичного обновления списков к единому формату среди всех продуктов экосистемы *Flussonic*. Пока мы рекомендуем начать передавать в API-запросах полные вложенные списки.

Традиционно концепция REST диктует разделение методов для создания и обновления уже созданных объектов.

Мы практически отказались от такого разделения и предпочитаем более устойчивый к сетевым сбоям подход с одновременным созданием или обновлением объектов. Если методы на создание и обновление разделены, то в выполняемом коде необходимо писать условие с повторами и осуществить проверку в цикле. Мы же сделали так, что сначала создаётся объект на сервере и, если пришел ответ о том, что такой объект уже создан, то попытаться его обновить (или же нет), в зависимости от ситуации.

Фактически этот подход соответствует правилам обработки формата документов JSON merge patch:

- Если передаваемый JSON merge patch содержит данные JSON, которые отсутствуют в целевом документе JSON, то эти данные добавляются.
- Если целевой документ уже содержит передаваемые данные, их значения обновляются.
- Значение `null` данных в JSON merge patch означает, что существующие данные в целевом документе удаляются.
- Если данные в передаваемом JSON merge patch представляют собой что-либо, кроме объекта, то весь целевой документ перезаписывается передаваемым JSON merge patch.
- Частичное изменение данных, не являющихся объектом, невозможно. Например, не получится заменить только некоторые значения в массиве.

Не нужно повторно передавать обязательные поля, если они уже есть в целевом документе и не изменились.

Если на сервере и на клиенте не реализовать *Idempotency token*, т.е. уникальный идентификатор (ID) действия, то повторные запросы на создание могут приводить к созданию нескольких новых объектов. Клиент может и не знать об этом, потому что сетевые запросы могут прерываться и без получения ответа. В случае с теми данными, которыми мы оперируем, как правило, получается создать ID объекта на стороне клиента, а значит, не нужно запрашивать генерацию ID на сервере (как это бывает в случае с буквальным следованием REST).

Необходимо сделать PUT-запрос:

```
curl -X PUT -H "Content-Type: application/json" -d '{"title":"ORT"}' "http
    ://FLUSSONIC-IP:8080/streamer/api/v3/streams/ort"
```

Важный момент: для API очень удобно, чтобы ID объекта занимал один сегмент. Это означает, что, если, например, имя потока составлено из нескольких сегментов: `sports/football`, то в доступе через API необходимо экранировать символ `/` с использованием `%2F` следующим образом:

```
curl -X PUT -H "Content-Type: application/json" -d '{"key":"value"}' "http
    ://FLUSSONIC-IP:8080/streamer/api/v3/streams/sports%2Ffootball"
```

Такой подход мы называем UPSERT, потому что это аналог SQL UPSERT.

Idempotency token для POST Idempotency token гарантирует, что одна и та же операция не будет выполнена дважды. Это особенно важно для облачных платформ, чтобы средства не списались со счета несколько раз за одно и то же действие.

Рассмотрим работу Idempotency token на примере:

- Клиенты используют для создания и обновления потоков запросы POST. В заголовке каждого запроса от клиента передается поле `Idempotency-Key` — это уникальное значение, которое создается на стороне клиента и которое сервер использует для распознавания последующих попыток выполнения одного и того же запроса.
- Когда клиент создает в облаке новый поток, имя этого потока генерируется *Flussonic* (а не на стороне клиента) — это гарантирует уникальность имен потоков.
- Если связь прерывается, клиенту приходится перезапустить запрос с тем же самым значением `Idempotency-Key`. *Flussonic* поймет, что это тот же самый запрос, и вернет сгенерированное для него имя. А если `Idempotency-Key` отсутствует, *Flussonic* посчитает повторную попытку отдельным запросом и сгенерирует новое имя потока.

Чтение объекта

Прочитать объект можно с помощью простого HTTP-метода GET:

```
curl -sS "http://FLUSSONIC-IP:8080/streamer/api/v3/streams/ort"  |jq
```

Ответ: JSON с информацией об объекте.

```
{
  "effective": {
    "name": "ort",
    "position": 7,
    "section": "stream",
    "static": true
```

```
  },
  "name": "ort",
  "named_by": "config",
  "position": 7,
  "static": true,
  "stats": {
    "alive": false,
    "bytes_in": 0,
    "bytes_out": 0,
    "client_count": 0,
    "dvr_enabled": false,
    "dvr_only": false,
    "dvr_replication_running": false,
    "id": "61496e6e-a22f-46af-bce5-5479f9067ead",
    "input_error_rate": 0,
    "last_access_at": 1632202350648,
    "lifetime": 0,
    "opened_at": 1632202350648,
    "out_bandwidth": 0,
    "publish_enabled": false,
    "remote": false,
    "retry_count": 19,
    "running": true,
    "running_transcoder": false,
    "start_running_at": 1632202350648,
    "transcoder_overloaded": false
  }
}
```


Удаление объектов

Удалить объект можно HTTP-методом DELETE:

```
curl -sS -X DELETE "http://FLUSSONIC-IP:8080/streamer/api/v3/streams/ort"
```

Ответ: HTTP 204 без тела.

Part II

Solutions

Chapter 4

DVB Cable, Sat & Terrestial

4.1 IP Ingest

4.1.1 Захват мультикаста

Flussonic Media Server умеет захватывать видео, передаваемое мультикастом по протоколу UDP MPEG-TS, включая RTP-инкапсуляцию.

В компьютерных сетях мультикаст, отправленный с источника, останавливается на первом же коммутаторе и не проходит дальше, пока потребитель в сети явно не запросит получение данных, отправляемых в мультикаст-группу. Такой запрос выполняется по протоколу IGMP и инициируется ядром ОС, а не ПО, которое не имеет доступа к запросам. Пока ПО работает, ядро переотправляет запросы IGMP коммутатору, продлевая действие запроса данных из мультикаст-группы. Без запросов IGMP коммутатору потребитель перестанет получать мультикаст уже через несколько секунд.

Содержание:

Требования

- Выделенный сервер с установленным Media Server. Виртуальный сервер может не подойти из-за особенностей работы мультикаста и высокой сложности настройки сети.
- Источник нешифрованного мультикаста. Вам нужно знать его адрес мультикаст-группы и порт. В простом случае начните с SPTS-источника, т.е. один канал — одна группа. Затем можете перейти к MPTS и инкапсуляции T2-MI.

Захват SPTS из источника сервером с одним интерфейсом

Если на сервере один сетевой интерфейс, то сделайте следующее:

- Создайте поток с некоторым именем.
- Укажите источник вида `udp://239.0.0.1:1234`:

```
stream example {
    input udp://239.0.0.1:1234;
}
```

Figure 4.1: SPTS ingest

Проигрывая видео и аудио в браузере, вы можете столкнуться со следующими трудностями:

- Если у вас видео с кодеками H.264 (AVC) или H.265 (HEVC), то видео будет проигрываться, но не на всех смартфонах Apple. Если у вас видео с кодеком MPEG-2, то для проигрывания видео потребуются специализированные программы, например, VLC.
- В мультикасте аудио передаётся в кодеках MPEG-2 audio или AC-3. Оба варианта не проигрываются в браузере.

Поэтому достаточно убедиться в следующем:

- поток активен,
- время жизни потока растёт без разрывов,
- входящий битрейт совпадает с ожидаемым, варьирующимся от 1 до 10 Мбит:

Figure 4.2: SPTS stream bitrate

Выбор интерфейсов

Как правило сервер, захватывающий мультикаст, имеет более одного интерфейса. Один интерфейс подключен к локальной сети, чтобы получать видео, а другой интерфейс подключен к Интернету, чтобы отправлять видео по HLS или HTTP MPEGTS и скачивать обновления.

Маршрут по умолчанию назначен на внешний интерфейс, поэтому ядро ОС будет отправлять запросы IGMP на внешний интерфейс. Так Flussonic Media Server не сможет получать видео.

Чтобы явно указать интерфейс, через который требуется запрашивать и получать мультикаст, добавьте его название в адрес источника перед адресом мультикаст-группы:

```
stream example {
  input udp://eth2@239.0.0.1:1234;
}
```

Если по какой-то причине вам удобнее указать IP-адрес интерфейса, на который отправлять запрос IGMP, то укажите его в адресе источника после адреса мультикаст-группы. Например, если на интерфейсе `eth2` сервер имеет IP-адрес `10.100.200.3`, то конфигурация будет иметь следующий вид:

```
stream example {
  input udp://239.0.0.1:1234/10.100.200.3;
}
```

Захват MPTS

Чтобы захватить мультипрограммный транспортный поток (MPTS), используйте вместо `udp://` специальные опции с указанием протокола. Подробнее об MPTS

Тюнинг ОС

Настройки Linux по умолчанию не позволяют захватывать видео по UDP без потерь, поэтому надо серьезно увеличивать размеры сетевых буферов.

Подробная информация об этом есть в статье про настройку производительности.

Важно также отметить, что для HD каналов рекомендуется размер буферов порядка 16 мегабайт.

Проблемы с захватом

С мультикастом часто бывают разнообразнейшие проблемы. Если у вас есть проблемы с качеством захватываемого мультикаста, можете попробовать проверить, в чём именно проблема.

Во-первых, вам надо полностью убрать все настройки firewall. `iptables -F`. Сначала надо сделать чтобы работало, потом всё остальное. В некоторых дистрибутивах Linux (например, CentOS) по умолчанию идут жесткие правила для iptables.

Также нужно отключить rp_filter, чтобы не было проблем с маршрутизацией. *Flussonic* сам отключает rp_filter только на том сетевом интерфейсе, где вы включаете прием мультикаста, чтобы не повышать уязвимости системы к сетевым атакам. Если по каким-то причинам *Flussonic* не смог отключить rp_filter, сделайте это вручную:

```
sysctl -w 'net.ipv4.conf.eth0.rp_filter=0'
```

и

```
sysctl -w 'net.ipv4.conf.all.rp_filter=0'
```

При необходимости измените `eth0` на нужный интерфейс.

Далее, следует понимать, что когда вы настроите мультикаст во Flussonic Media Server и будете смотреть видео с Flussonic Media Server, на качество видео влияет множество факторов: качество сигнала, качество захвата, работа сервера и качество вашей сети. Проблемы, которые вы увидите в такой конфигурации, будут говорить только о том, что у вас проблемы, но никак не о работе Flussonic Media Server. Особенно это проявляется при отладке работы HD каналов: смотреть без затыков 10 мегабит видео могут считаные проценты пользователей.

Например, если вы запустите:

```
/opt/flussonic/contrib/multicast_capture.erl udp
   ://239.0.0.1:1234/10.100.200.3 output.ts
```

запишете секунд 30 видео, скопируете себе на компьютер и посмотрите видео в VLC, то вы получите неискаженную картину того, как мультикаст приходит на сервер. Этот скрипт не распаковывает MPEG-TS, а пишет сырой мультикаст на диск.

Если на этом этапе вы получили хорошее ровное видео, то можно идти дальше и запускать на самом сервере:

```
curl -o output.ts http://127.0.0.1:80/example/mpegts
```

Таким образом, вы получите видеопоток, который был захвачен Flussonic Media Server, распакован и упакован обратно в MPEG-TS. Этот файл надо скачать себе на компьютер и посмотреть локально, чтобы убедиться в том, что ваше качество канала не влияет на эксперименты.

Если на этом этапе видео тоже хорошее, а при просмотре с Flussonic Media Server дергается, проблема скорее всего в том, что канала не хватает на передачу видео с Flussonic Media Server к вам.

Проблемы с коммутаторами

Иногда проблемы возникают с настройками коммутаторов. Например, у одного клиента возникла проблема с ограничением на количество принимаемых каналов. Оказалось, что стоит лимит на количество подписок на одном порту. Это можно выяснить следующей командой:

```
#debug igmp snooping all
```

При этом появляются сообщения:

```
%Jun 25 15:12:18 2015 SrcIP is 192.168.121.2, DstIP is 226.2.1.16
%Jun 25 15:12:18 2015 Groups joined have reached the limit, failed to add
   more groups
```

В данном случае получилось починить с помощью команды:

```
#ip igmp snooping vlan XX limit group <1-65535>
```

Проблемы с адресами мультикаст-групп

В определенных случаях на головных станциях возникают проблемы с адресами мультикаст-групп в диапазоне от `224.0.0.0` до `239.1.1.1`. Поэтому рекомендуется использовать адреса мультикаст-групп от `239.1.1.1` и выше. Все, что ниже, иногда могут не работать.

4.1.2 Захват MPTS

MPTS — это транспортный поток (MPEG-TS), содержащий сразу несколько телеканалов, иными словами мультипрограмный. MPTS используется в DVB-сетях, где отсутствует протокол IP и нет других способов поделить канал связи между отдельными потоками видео: спутник, кабельные сети, наземное радиовещание. Flussonic Media Server умеет также формировать MTPS.

Как правило, MPTS принимают мультикастом по UDP из головной станции, принимающей DVB-сигнал и отправляющей поток в IP-сеть. Реже MPTS принимают по HTTP.

На этой странице:

- Требования.
- Как настроить захват MPTS в Flussonic Media Server.

Требования

- Соответствие требованиям для захвата SPTS мультикастом.
- Номер требуемой программы (PNR) в MPTS. Обычно его можно узнать у поставщика контента или посмотреть в потоке самостоятельно, используя утилиту FFprobe:

```
ffprobe http://FLUSSONIC-IP/STREAM_NAME/mpegts
```

В выводе найдите `Program`, справа от которого будет указан номер этой программы:

```
Input #0, mpegts, from 'http://FLUSSONIC-IP/STREAM_NAME/mpegts':
  Duration: N/A, start: 86814.877644, bitrate: N/A
  Program 1
    Metadata:
      service_name    : Service01
      service_provider: FFmpeg
    Stream #0:0[0x100]: Video: h264 (Main) ([27][0][0][0] / 0x001B),
   yuv420p(top first), 720x576 [SAR 16:11 DAR 20:11], 25 fps, 25 tbr, 90k
   tbn, 50 tbc
    Stream #0:1[0x101]: Audio: aac (LC) ([15][0][0][0] / 0x000F), 48000 Hz,
     stereo, fltp, 59 kb/s
...
```

Как настроить захват MPTS в Flussonic Media Server

Для каждой программы, которую вы хотите захватить, выполните следующие шаги:

- в Flussonic UI создайте поток, зайдя в **Media** > **Streams** и нажав **+** в левом верхнем углу. Укажите источник через `mpts-udp://`, например, `mpts-udp://239.0.0.1:1234`.

* `mpts-udp://` — захват по UDP. * `mpts-http://` — захват по HTTP. * `mpts-https://` — захват по HTTPS. * `udp://` — не рекомендуется, но будет работать.

!!! warning Если вы настроите захват MPTS с помощью той же конфигурации, что и захват мультикастом SPTS `udp://`, то сервер несколько раз захватит один и тот же поток с высоким битрейтом.

* `mpts-dvb://` — захват с DVB-карты

Сохраните настройки, нажав **Save**.

- Перейдите в настройки потока, нажав на имя потока, и затем на вкладку **Input**. Откройте настройки источника, нажав **Options** справа от URL источника. В разделе **Headers** найдите поле *MPEG-TS program: select only specific program from MPTS MPEG-TS* и укажите номер программы (PNR), которую требуется захватить.
- Для DVB-T2 MI сигнала, инкапсулированного в MPEG-TS, также укажите идентификатор канала PLP (Physical Layer Pipe), используя `?plp=` в URL источника.
- Сохраните настройки.

Повторите эти шаги это столько раз, сколько программ вы хотите захватить.

Параметры захвата MPTS вы можете найти в 7Cinputstag/stream/operation/stream-get/response%7CinputsFluss
API Reference.

Пример настройки для захвата трёх программ из MPTS в конфигурационном файле `/etc/flussonic/fluss`

```
stream one {
  input mpts-udp://239.0.0.1:1234 program=2001;
}

stream two {
  input mpts-udp://239.0.0.1:1234 program=2002;
}

stream three {
  input mpts-udp://239.0.0.1:1234 program=2003;
}
```

, где:

- `mpts-udp://239.0.0.1:1234` — источник MPTS по UDP.
- `program` — номер программы.

При использовании источника `mpts-udp://` Flussonic Media Server захватит MPTS-поток всего один раз. Так каждый поток на сервере принимает требуемую программу. Этим `mpts-udp://` отличается от остальных источников, включая `udp://`.

Используя `mpts-udp://` для захвата MPTS по UDP, вместо `udp://`, сервер захватывает MPTS один раз, не нагружая CPU до предела. Ниже представлен график использования CPU в разделе Pulse при захвате 12 программ из MPTS:

Figure 4.3: График загрузки CPU при захвате UDP MPTS

4.1.3 Захват спутникового видео

Для захвата видео со спутника используется так называемое IRD (Integrated Receiver-Decoder) оборудование и головные станции. Flussonic Media Server умеет принимать видео по IP протоколам с любых IRD устройств и систем. Также можно захватывать видео с плат DVB-S и др. сразу во Flussonic.

Рассмотрим принципы захвата видео со спутников и вопросы выбора оборудования для этой цели.

Содержание:

- Долгота
- Настройка ресивера
- Дескремблинг
- Выбор оборудования
- Захват с плат DVB-S используя Flussonic
- Захват с плат TBS ISDB-T Quad используя Flussonic
- Получение каналов
- Итоги

Долгота

Телевизионные спутники висят на геостационарной орбите Земли над экватором. Их положение над Землей постоянное, поэтому спутники идентифицируются долготой, над которой они висят. Так, например, спутник ABS 2 вращается вокруг Земли, постоянно находясь над 75° восточной долготы.

Геостационарное размещение делает очень простым монтаж спутниковых тарелок: достаточно направить на нужную точку и не нужно поворачивать тарелку.

Поскольку спутник находится над конкретной долготой, то, как правило. разные спутники используются для вещания определенного набора каналов. Например, спутник Appstar-7 76.5°E наполнен индийскими каналами, а спутник Galaxy-17 91.0°W больше используется для вещания каналов для США.

Важно отметить, что у некоторых спутников есть несколько лучей, т.е. зон максимального приёма. Каналы в лучах могут отличаться. Например, один и тот же спутник может на Россию вещать один набор каналов, а на Таиланд другой.

Настройка ресивера

С одного спутника в одном луче уходит множество мультиплексоров. Мультиплексор — это один цифровой канал, для приёма которого надо настроить карту захвата на определенную частоту и поляризацию.

Т.е. с одного спутника одновременно вещается множество частот в нескольких поляризациях.

Поляризация бывает правой/левой или горизонтальной/вертикальной. Бытовые спутниковые антенны (точнее головки этих антенн или конвертеры) умеют принимать сразу обе поляризации на один выходящий кабель, но делают это плохо. Профессиональные конвертеры принимают все поляризации, но на разные выходы.

Такое разделение обусловлено тем, что для переключения приёма поляризации плата захвата подает напряжение 13 или 18 вольт. Ниже таблица соответствия поляризации и напряжения:

Напряжение (В)	**13-14**	**17-18**
Линейная поляризация	вертикальная	горизонтальная
Круговая поляризация	правая	левая

В некоторых ресиверах указывается напряжение, в некоторых указывается поляризация. По сути всё сводится к подаче напряжения.

Если простым сплиттером соединить две платы захвата и на одной подать 18 вольт, а на другой 13, то вторая не получит сигнал.

Некоторые ресиверы умеют отключать напряжение. В этом случае они должны быть сплиттером соединены с другим ресивером/картой захвата, который всё-таки подает напряжение.

Диапазоны частот со спутника разделяются условно на верхний и нижний диапазон. Граница проходит примерно на 11700 МГц. При захвате частот ниже 11700 обычно выставляется частота гетеродина, она же LNB Frequency в 9750 МГц. При захвате частот выше 11700 настраивается частота гетеродина в 10600 МГц

После того, как правильно скоммутировали провода, не напутав с напряжением, настроили нужную частоту на приёмнике и частоту гетеродина и приёмник автоматически выбрал FEC (количество битов для контроля ошибки) и модуляцию (QPSK, 8PSK и т.п.), то ресивер начинает получать битовый поток, т.е. мультиплексор.

Мультиплексор представляет из себя MPEG-TS поток и позволяет в один поток упаковывать много разных телеканалов с разными языковыми дорожками и субтитрами. Бытовой спутниковый приёмник позволяет забрать только один канал из мультиплексора, но профессиональные приёмники и DVB PCI платы захвата позволяют снимать все каналы с мультиплексора.

Ниже будет подробнее описана структура MPEG-TS потока в мультиплексоре.

Дескремблинг

Большинство каналов на спутнике вещаются зашифрованными. Шифрование используется для управления доступа к каналам различных пользователей: кто оплатил следующий месяц, тот и смотрит каналы.

Процедура расшифрования спутникового канала называется *дескремблингом*, а сам шифрованный канал называют скремблированным.

Механизм управления доступом к спутниковым телеканалам называется Conditional Access (CA), в русской терминологии используется термин «условный доступ».

На сегодняшний день используются различные схемы шифрования, но в основном все современные схемы работают примерно следующим образом:

- абонент получает карту доступа (похожа на большую, необрезанную SIM-карту с таким же чипом);
- на карте доступа есть приватный ключ;
- раз в месяц меняется ключ пакета;
- через спутник рассылается для каждого абонента ключ пакета, шифрованный публичным ключом карты;
- карта доступа запоминает ключ пакета;
- раз в минуту на каждом канале меняется ключ потока, который шифрован ключом пакета;
- если пользователю пришел ключ пакета и он смог его расшифровать, то он может получить доступ к каналу.

У этой схемы есть вариации и усложнения, но концептуально схема именно такая. Если не заплатить денег спутниковому оператору, то в конце месяца он не разошлет обновленный ключ и карта не сможет расшифровывать.

С технической точки зрения с дескремблированием сложилась грустная ситуация. Спутниковые операторы и пираты ведут многолетнюю и безуспешную борьбу друг с другом, в которой страдают операторы связи.

Когда обычный абонент покупает спутниковую тарелку для дома, он получает карту доступа и бытовой спутниковый приёмник, который имеет чип для дескремблирования одного канала по той схеме шифрования, которая выбрана у этого оператора. Условно говоря, приёмник для НТВ+ не подойдет к Континент-ТВ.

Оператор связи не может использовать 200 бытовых приёмников чисто технически, поэтому используются профессиональные приёмники, которые снимают не по одному каналу с мультиплексора, а все каналы. Однако официальный метод дескремблирования, который предлагают операторы связи, подразумевает использование специальных CA модулей. Это плата, похожая на PCMCIA модуль, в который втыкается карта доступа.

CA модуль самостоятельно дескремблирует каналы. Для этого он забирает каналы с головной станции, дескремблирует и отдает обратно.

Проблема в том, что даже для профессионального CA модуля дескремблирование 8 каналов — это предельная нагрузка. Учитывая, что на многих мультиплексорах сегодня бывает до 30 каналов, получается, что один и тот же мультиплексор надо захватывать через сплиттер несколько раз, используя дорогостоящие карты захвата или головные станции.

Детальное описание процесса дескремблирования на головной станции или с помощью компьютера находится за рамками этого описания, особенно учитывая, что допустимые схемы дескремблирования должны быть согласованы с поставщиком контента, чтобы не нарушить уголовный кодекс.

Выбор оборудования

Традиционный способ захвата спутникового эфира — использование так называемых головных станций.

Головная станция — это специализированный спутниковый приёмник, умеющий захватывать больше одного канала (до сотен каналов).

Более дорогие головные станции, такие как WISI Compact Headend, предлагают более плотную компоновку по сравнению с недорогими станциями:

Figure 4.4

Вместо 1-2 мультиплексоров в 1U корпусе можно захватывать вплоть до 24. Правда дескремблировать их все не получится, потому что в этом случае количество мультиплексоров сократится до 12 (нужно место под CI модули), а профессиональная головная станция не может дескремблировать больше 8-10 каналов с одного мультиплексора, потому что это по большому счёту дорогущий, но слабый компьютер.

Альтернативный вариант — использование компьютера для захвата со спутника.

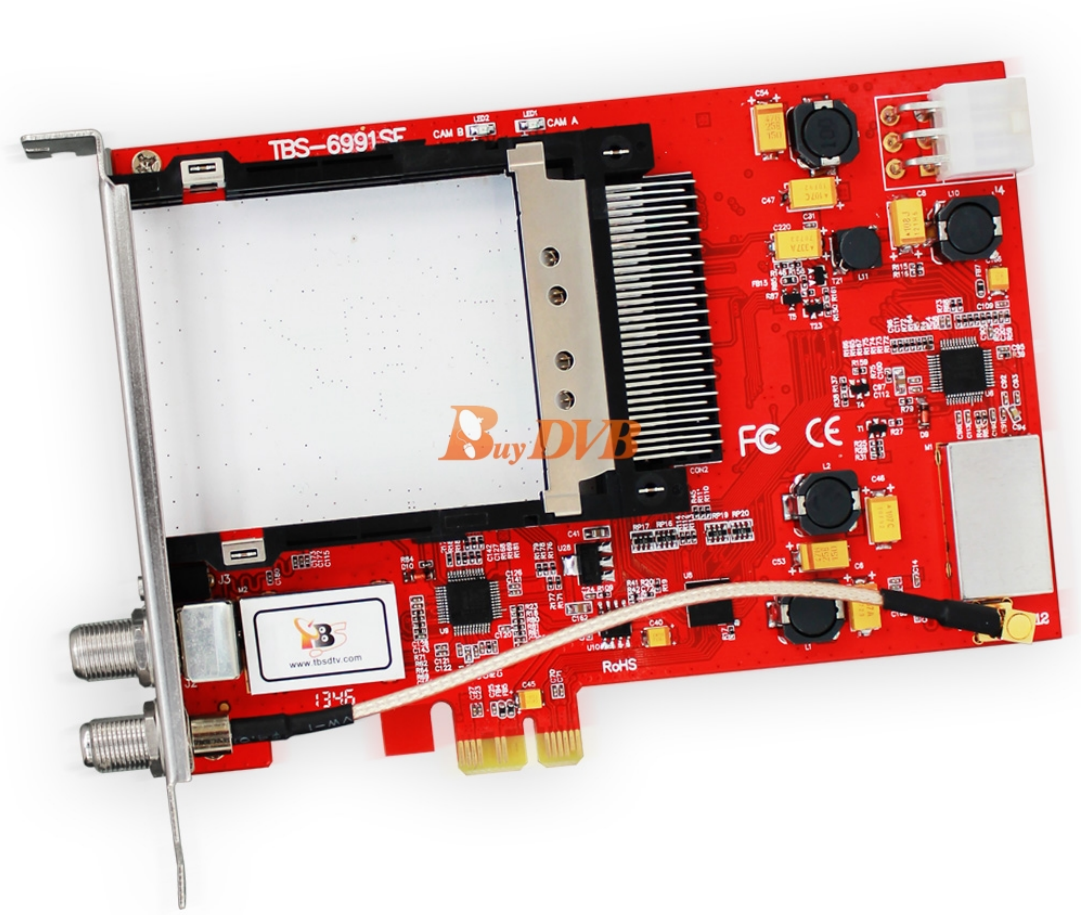


Figure 4.5: Платы захвата

В обычный PC помещается до 7 подобных плат (главное подобрать материнскую плату). Разумнее всего брать либо с 4 выходами, либо с 2 но с CI модулем.

При покупке подобных плат очень важно свериться с поддержкой их в Linux на сайте проекта LinuxTV, кому в конце концов может понадобиться плата захвата, которая поддерживается только Windows?

Flussonic сможет работать с видео независимо от того, на каком оборудовании оно захвачено. Поэтому при выборе оборудования достаточно учесть различия в обслуживании, ориентируясь на выделенный бюджет и технические предпочтения.

При построении головной станции можно выбрать отдельное полностью аппаратное решение либо решение, построенное на x86 платформе.

Аппаратные решения отличаются большей ценой и несколько меньшей гибкостью. Для настройки

таких решений, как правило, используется веб-интерфейс, что удобно, но может замедлить решение разных проблем. Взамен вендоры таких решений гарантируют стабильную емкость, длительную непрерывную работу и превосходную поддержку.

Программные решения, которые вы будете использовать с платами захвата под PC, в свою очередь обладают непревзойденной гибкостью в настройке и работе. Настройки можно задавать через конфигурационные файлы, а Linux предоставляет более широкие возможности для отладки проблем, чем закрытые аппаратные решения.

Захватывать видео с плат DVB, ATSC, ISDB можно и при помощи Flussonic Media Server.

Захват напрямую с плат DVB, ATSC, ISDB

Вы можете захватывать видео с плат DVB, ATSC, ISDB сразу во *Flussonic*. Для этого добавьте параметры платы в директиве **dvb_card** и задайте источник через схему **mpts-dvb://**.

Пример:

```
dvb_card a0 {
    system dvbs2;
    adapter 1;
    frontend 3;
    frequency 195028615;
    symbol_rate 29500;
    polarization v;
    modulation qam256;
    disabled;
    comment "13E high vertical";
}
stream channel5 {
  input mpts-dvb://a0?program=1713;
}
```

Здесь:

- system (atsc|dvbs2|dvbt2|dvbt|isdbt) — тип адаптера. Процесс настройки аналогичен для различных стандартов. См. также пример с платой TBS ISDB-T Quad.
- frontend — номер frontend на плате.
- frequency — несущая частота для данной программы в Гц.
- symbol_rate — символьная скорость мультиплексора.
- polarization — поляризация для данного канала.
- modulation — тип модуляции.
- disabled — устройство не работает.
- program — программа (телеканал).

Типов адаптера поддерживается больше, но не все они еще протестированы и поэтому нет гарантии, что захват будет работать. Полный список возможных значений параметра `system` см. в схеме

Настройка захвата с DVB карты в UI Все карты отображаются в списке в **Media** > **DVB cards**. Для каждой добавленной и включенной карты зеленый индикатор сверху показывает уровень сигнала. По ссылке **Programs** можно посмотреть структуру потока и выбрать программы для вещания в конкретных потоках.

Figure 4.6: DVB options

Чтобы добавить карту DVB, перейдите в **Media** > **DVB cards** и нажмите **Add DVB card**. Заполните поля:

- Name — имя DVB карты. По умолчанию новой карте присваивается имя **newDVBCardN**, где N — порядковый номер (начиная с 1). Вы можете изменить имя карты.
- Adapter — номер адаптера. Номера адаптеров видны в /dev/dvb/adapterN.
- System — тип адаптера. Помимо DVB поддерживаются разные системы, их настройка похожа.
- Frequency — несущая частота для данного канала в мегагерцах.
- Polarization — поляризация для данного канала.
- Symbol rate — символьная скорость мультиплексора.

Чтобы раскрыть расширенные опции, кликните по стрелке справа в строке с основными опциями.

- Frontend — номер фронтенда на плате. У адаптера может быть 1-N фронтендов, но чаще всего он один и по умолчанию его номер 0.
- Enabled — использовать эту DVB карту.

Остальные расширенные опции DVB карты:

- Code rate HP — скорость кода потока с высоким приоритетом.
- Code rate LP — скорость кода потока с низким приоритетом.
- Guard interval — режим вставки защитного интервала, т.е. отступа между передачами данных, который не дает им смешиваться.
- Rolloff — коэффициент избирательности, в %. Используется для оценки полосы пропускания, совместно с символьной скоростью.
- Pilot — включение, отключение или автоматическое определение контрольных сигналов.
- Modulation — способ модуляции DVB-C.
- Hierarchy — констелляция для иерархической передачи.
- Transmission mode — режим передачи DVB.
- Bandwidth — полоса пропускания, в Гц.
- Plp stream id — идентификатор канала PLP; если задан, то перед декодированием MPEG-TS осуществляется распаковка DVB-T2 MI.
- Hw — тип устройства для карты адаптера.
- Device — номер модулятора в адаптере.
- Serial — серийный номер карты.

- Int freq — опорная частота, в МГц.
- Compensate time drift PPM — максимальное компенсирующее смещение внутреннего времени источника, округленное до 6 знаков после запятой.
- Port — номер порта.
- Attenuator — снижение уровня сигнала.
- Interleave — использовать перемежение битов в битовом потоке, чтобы минимизировать искажения при передаче данных.
- Gain — усиление выходного сигнала до указанного значения в дБ.
- Input bitrate — битрейт на входе, в Мбит/с.
- Video device — видео-устройство для захвата видео с карт Stream Labs SDI/ASI. Это путь к файлу устройства, созданному Video4Linux на диске.
- High band — используется ли высокий диапазон частот.
- Comment — любой текстовый комментарий.

Теперь нужно добавить поток с источником вида `mpts-dvb://a0 program=<NUMBER>`. Вы можете сделать это вручную, выбрав **Media** > **Streams** > **Add**, или автоматически на странице **Programs** (смотрите описание ниже).

Просмотр структуры MPTS и добавление потоков Теперь вы можете просмотреть структуру и служебную информацию захваченного с DVB карты MPTS и добавить поток для конкретной программы на вашем *Flussonic*.

Чтобы просмотреть структуру MPTS, перейдите в **Media** > **DVB cards** и нажмите ссылку **Programs** рядом с зеленым индикатором. Вы увидите таблицу, в которой перечислены все программы с информацией о PID (сколько видео-, аудио- и текстовых треков содержит та или иная программа).

Просмотр информации о треках Чтобы посмотреть подробную информацию о треках, содержащихся в программе, нажмите значок информации в столбце **PIDs**. Для каждого трека вы увидите его номер, PID, разрешение, кодек и битрейт.

Добавление потока для программы Чтобы создать поток для конкретной программы, нажмите кнопку **Add** для этой программы. *Flussonic* автоматически создаст соответствующий поток. Источник потока будет выглядеть следующим образом: `mpts-dvb://a0 program=<NUMBER>`.

Имя потока будет соответствовать сервисному имени программы, например, `Euronews_Russian`. Если для программы не указано сервисное имя, имя потока будет сгенерировано из имени DVB карты и идентификатора программы, например, `a0_790`.

Figure 4.7: DVB options

Просмотр информации о сигнале Для каждой программы можно посмотреть статистическую информацию о захваченном сигнале. Для этого нажмите значок информации рядом с зеленым индикатором.

Вы можете посмотреть следующие параметры статистики:

- ber — процент битов с ошибками от общего числа переданных битов (bit error rate).
- has_carrier — в сигнале обнаружена несущая частота.
- has_lock — DVB сигнал успешно зафиксирован.
- has_signal — поверх обычного уровня шума обнаружен сигнал.
- has_sync — обнаружены байты синхронизации.

Figure 4.8: DVB options

- has_viterbi — DVB сигнал зафиксирован на стадии декодера Viterbi.
- snr — соотношение ”сигнал-шум” в процентах.
- snr_raw — соотношение ”сигнал-шум” в исходном 16-битном формате.
- strength — уровень сигнала в процентах.
- strength_raw — уровень сигнала в исходном 16-битном формате.

Значения параметров сигнала могут различаться в зависимости от производителя устройства и драйвера DVB-карты.

Figure 4.9: DVB options

Передача потока с DVB-карты без обработки Flussonic может принимать и передавать поток с DVB-карты «как есть», без перепаковки. Для этого применяется URL вида

`tshttp://ADMIN:PASSWORD@FLUSSONIC_IP/flussonic/api/dvbts/[DEVICE_ID]`

Пример:

```
dvb_card a0 {
    system dvbs2;
    adapter 1;
    frontend 3;
    frequency 195028615;
    symbol_rate 29500;
    polarization v;
    modulation qam256;
```

```
    disabled;
    comment "13E high vertical";
}
stream STREAM_NAME {
  input tshttp://ADMIN:PASSWORD@FLUSSONIC_IP/flussonic/api/dvbts/a0 program
   =123;
}
```

Фиксация номеров устройств для DVB карт После перезагрузки ОС сервера номера устройств в ней могут поменяться. Например, если вы захватываете видео с нескольких DVB карт и перезагрузите ваш сервер, может оказаться, что DVB захват сломан, т.к. номера этих карт в системе изменились.

Чтобы избежать такой ситуации, зафиксируйте номера DVB карт, создав правило udev.

Для этого сначала запустите следующую команду, чтобы посмотреть параметры DVB карт:

```
udevadm info -a -n /dev/dvb/adapter0/frontend0

   looking at device '/class/dvb/dvb0.frontend0':
    KERNEL=="dvb0.frontend0"
    SUBSYSTEM=="dvb"
    DRIVER==""

   looking at parent device '/devices/pci0000:00/0000:00:1e
   .0/0000:02:00.0':
    KERNELS=="0000:02:00.0"
    SUBSYSTEMS=="pci"
    DRIVERS=="b2c2_flexcop_pci"
    ATTRS{vendor}=="0x13d0"
    ATTRS{device}=="0x2103"
    ATTRS{subsystem_vendor}=="0x13d0"
    ATTRS{subsystem_device}=="0x2103"
    ATTRS{class}=="0x028000"
    ...
```

Затем создайте файл `/etc/udev/rules.d/10-local.rules` со следующим содержимым (фиксация по параметру KERNELS):

```
SUBSYSTEM=="dvb", KERNELS=="0000:02:00.0", PROGRAM="/bin/sh -c 'K=%k; K=$${
   K#dvb}; printf dvb/adapter0/%%s $${K#*.}'", NAME="%c", GROUP="video"
SUBSYSTEM=="dvb", KERNELS=="0000:02:02.0", PROGRAM="/bin/sh -c 'K=%k; K=$${
   K#dvb}; printf dvb/adapter1/%%s $${K#*.}'", NAME="%c", GROUP="video"
```

Захват с плат TBS Quad ISDB-T

Для захвата с платы Quad ISDB-T в файл конфигурации добавьте следующие параметры платы в директиве **dvb_card**, заменив значения номера адаптера и частоты на свои.

Затем задайте источник через схему **mpts-dvb://**.

Пример:

```
dvb_card a0 {
  system isdbt;
  adapter 5;
  frequency 546000000;
}
stream channel5 {
  input mpts-dvb://a0?program=1713;
}
```

Здесь:

- system — тип адаптера (isdbt)
- frequency — несущая частота в Герцах
- program — телеканал

Получение каналов

Как было выше сказано, мультиплексор — это MPEG-TS поток. Транспортный контейнер MPEG-TS позволяет в один поток байт упаковывать много одновременно идущих потоков, предоставляя стандартизованный способ для выбора нужного под-потока.

Один телеканал называется программой (program). MPEG-TS, в котором идет только одна программа, называется SPTS — single program transport stream. Со спутника идет MPEG-TS, в котором много программ, это называется MPTS — multiple program transport stream.

MPTS удобен для передачи в носителях типа спутника или кабеля, когда полоса фиксированная и для выравнивания трафика поток даже приходится добивать ненужными байтами. SPTS удобен при передаче по IP, когда весь огромный мультиплексор клиенту не нужен, а нужен только один канал.

Процесс смешивания нескольких SPTS в MPTS называется *мультиплексированием* и как правило производится перед подачей потока на спутник или кабель. Процесс расщепления MPTS на несколько SPTS называется *демультиплексированием* и происходит при приёме со спутника.

По IP MPTS передают очень редко, например для транзита со спутника в кабель.

Сам MPEG-TS представляет из себя последовательность пакетов по 188 байт. Первый байт всегда 0x47 и используется для статистически достоверной синхронизации в потоке.

В следующих трех байтах закодирован 13-битный номер потока внутри MPEG-TS. Этот номер называется Pid и поэтому в профессиональном жаргоне зачастую под-поток так же называют пидом по его номеру.

Есть несколько стандартных номеров Pid, которые зарезервированы под служебные нужды. Условно говоря, это все номера до 32.

В потоке с Pid 0 идет информация о существующих программах в этом MPEG-TS потоке. Эта информация упакована в PAT — program adaptation table. PAT — это один из вариантов PSI информации. PSI или PSI таблицы — это метаинформация, идущая в MPEG-TS потоке и необходимая либо для получения доступа к аудию/видео, либо для получения дополнительной информации, как например, расписание передач или информация о каналах на других мультиплексорах.

Важно понимать, что все PSI таблицы проектировались для случаев, когда ресивер не получает никаких данных, кроме данных со спутника. Поэтому большинство PSI таблиц для IPTV OTT сервиса бессмысленны: расписание там зачастую нерабочее, а информация о других мультиплексорах вообще бессмысленна.

В PAT записано, какие номера программ (pnr, program number, service id) идут на каких пидах. На этих пидах будут идти не аудио/видео потоки, а PMT — program mapping table. В PMT уже будет написано, на каком пиде идет видео поток и на каких пидах идут разные языки аудио.

При настройке демультиплексирования можно настраивать по пидам, а можно по pnr. Второе более предпочтительно, потому что пиды на спутнике могут перенастроиться без предупреждения, а номера программ как правило не меняются.

Итоги

Процесс захвата видео со спутника выглядит следующим образом:

- администратор настраивает захват на головной станции или программе с нужного входа, на нужной частоте и с нужной поляризацией;
- поток демультиплексируется из MPTS в несколько разных SPTS, согласно настройкам (скорее всего по pnr);
- несколько отдельных SPTS вещаются в сеть по мультикасту.

В классическом IPTV на этом всё и заканчивается, клиенты получают свой мультикаст через каскад роутеров, общающихся по PIM протоколу, но в нашем случае всё только начинается, потому что дальше результирующее видео надо транскодировать

4.1.4 Захват потока по SRT

Flussonic поддерживает захват видео по протоколу SRT. Подробнее об SRT см. на странице Использование протокола SRT.

SRT ingest URL Чтобы настроить захват потока по протоколу SRT во *Flussonic*, добавьте поток, указав URL источника одним из следующих способов:

- когда у вас есть только IP и PORT:
- Когда вам дали ещё и имя стрима STREAM_NAME:

Например:

```
stream ingest_srt {
  input srt://SRT-SOURCE:8888 streamid="#!::m=request,r=srt_stream";
}
```

В примере выше мы настроили порт 8888 для захвата потока `srt_stream`.

Параметры для управления захватом потоков по SRT *Flussonic* также позволяет управлять захватом SRT с помощью ряда параметров.

*

```
stream ingest_srt {
  input srt://SRT-SOURCE:9999 passphrase=0987654321 streamid="#!::m=request
    ";
}
```

или:

```
stream ingest_srt {
  input srt://SRT-SOURCE:9999?streamid=#!::m=request&passphrase=0987654321;
}
```

В примере выше мы настроили защищённый порт 9999 с помощью `passphrase`.

Кроме того, Вы можете передать в `streamid` потока:

- `u`=USERNAME — имя пользователя, используемое *Flussonic* в качестве токена при авторизации в сессии захвата.
- `a`=USER_AGENT — агент пользователя, также используемый при авторизации в сессии захвата.

4.1.5 Прием публикации по SRT

Вы можете легко опубликовать поток по SRT на определенный сервер Flussonic Media Server.

> Если у вас много серверов, а количество активных публикаций заранее неизвестно, используйте логику, описанную в статье Прием публикаций по SRT от множества авторов.

Настройка в UI

Чтобы настроить публикацию во *Flussonic Media Server* по протоколу SRT:

- Создайте поток с пустым **Source URL**, а затем в профиле этого потока установите переключатель **Publication** в положение **Enabled**.
- Укажите порт **SRT (dedicated publish port)** и **passphrase** для приема публикации в этот поток.
- Сохраните настройки. После этого появится URL в поле **SRT (DEDICATED PUBLISH PORT)**. Используйте этот URL для публикации.

> Описанный выше способ позволяет задать отдельный порт для публикации по SRT в конкретный поток. Если вы хотите использовать глобальный порт для приема публикации по SRT во все потоки, используйте ссылку **SRT (SHARED)**, а порт задайте на вкладке **Config**. Подробнее о портах для SRT читайте в разделе Порт для SRT.

Проверка публикации

Опубликуйте поток во *Flussonic*, например так:

```
ffmpeg -re -i /opt/flussonic/priv/bunny.mp4 -c copy -y -f mpegts 'srt://
   localhost:9050?passphrase=mytopsecret'
```

Настройка через файл конфигурации

Настроить поток для приема публикации по SRT можно в файле конфигурации. В зависимости от способа указания SRT-порта настройки и URL будут различаться:

- Глобальный порт (в UI **SRT (SHARED)**)

```
srt_publish {
  port 9998;
  passphrase 0123456789;
}
stream pub {
  input publish://;
}
```

URL для публикации:

```
srt://FLUSSONIC-IP:SRT_PORT?passphrase=PASSWORD&streamid=#!::r=STREAM_NAM,m=publ
```

- Отдельный глобальный порт для публикации

```
srt_publish {
  port 9998;
}
stream mysrt {
  input publish://;
}
```

URL для публикации потока:

```
srt://FLUSSONIC-IP:SRT_PORT?streamid=#!::r=STREAM_NAME
```

- Отдельный порт для потока

```
stream mysrt {
  input publish://;
  srt 9998;
}
```

Для публикации используйте URL:

```
srt://FLUSSONIC-IP:SRT_PORT?streamid=#!::m=publish
```

- Отдельный порт для публикации потока

```
stream pub {
  input publish://;
  srt_publish {
    port 9998;
    passphrase 0123456789;
  }
}
```

URL для публикации:

```
srt://FLUSSONIC-IP:SRT_PORT?passphrase=PASSWORD
```

> Если вы определяете **одновременно** глобальные и локальные настройки, последние имеют больший приоритет и применяются первыми.

Дополнительные параметры публикации по SRT

Список параметров для управления публикацией по SRT (указываются внутри `srt_publish`) приведен здесь.

Кроме того инициатору публикации *Flussonic* отправляет в URL информацию об агенте (версии *Flussonic*) и идентификаторе сессии `sessionId` (самостоятельно их указывать не нужно).

4.2 SDI & ASI

4.2.1 Захват SDI на DekTec

Flussonic Media Server позволяет:

- Получать видео и аудио с плат захвата DekTec SDI.
- Передавать видео на DekTec SDI.
- Читать телетекст из полученного с DekTec потока как в стандартном (SD SDI), так и в высоком (HD SDI) разрешении.
- Читать метки врезки рекламы из полученного с DekTec потока. Метки врезки рекламы преобразуются из VBI SCTE-104 в формат SCTE-35, пригодный для отправки в MPEG-TS и HLS. Дополнительные настройки для этого не требуются.
- Передавать телетекст (Teletext B) из MPEG-TS в видео, отправляемое на SDI карты DekTec.

Установка драйвера DekTec в Ubuntu

Вы можете установить драйвер, используя DekTec SDK. Readme драйверов содержит подробную информацию о том, какой драйвер должен быть скомпилирован для конкретной карты. Следуйте этой инструкции. После установки драйвера перезагрузите сервис flussonic командой `service flussonic`. Если вы обновили ядро Linux, то может потребоваться переустановить драйвер DekTec.

Захват видео с платы DekTec SDI

Серийный номер платы захвата Dektec SDI вы можете найти во вкладке DVB `cards` в выпадающем меню `Installed boards` или с помощью консольной утилиты от Dektec `DtInfoCL`.

Для захвата видео с платы DekTec SDI настройте поток с источником `dektec://serial_number:port`.

Чтобы указать источник через веб-интерфейс, перейдите в раздел **Media** > нажмите имя потока > во вкладке **Input** в поле **New URL** введите источник > нажмите **Save**.

Вы также можете настроить поток через файл конфигурации, например:

```
stream example {
  input dektec://2174220025:2;
}
```

Захват звука с платы DekTec SDI

Flussonic Media Server может получать с платы DekTec SDI звук в форматах PCM и AC-3 (Dolby Digital). Первые две звуковые дорожки должны быть в формате PCM. Если следующие две звуковые дорожки содержат аудио в формате AC-3, то оно также будет получено.

Никаких дополнительных настроек для этого не требуется.

Figure 4.10: DekTec SDI

4.2.2 Отправка потока на DekTec SDI

Flussonic Media Server может не только захватывать, но и передавать поток на плату захвата и вывода DekTec SDI.

Серийный номер платы захвата Dektec SDI вы можете найти во вкладке DVB `cards` в выпадающем меню `Installed boards`.

Для вывода на DekTec используйте директиву `push dektec://serial_number:port`:

```
stream test {
  input udp://239.0.0.1:1234;
  push dektec://2174220025:2 video_format=pal;
}
```

Flussonic декодирует и затем передает поток на указанное устройство DekTec. При необходимости можно указать опцию `deinterlace=true` для устранения чересстрочности.

4.2.3 Захват ASI на DekTec

ASI (*Asynchronous Serial Interface*, Асинхронный последовательный интерфейс) — логический интерфейс для передачи потоков MPEG-TS со скоростью до 270 Мбит/с в зависимости от применения.

ASI применяется в спутниковом и кабельном цифровом телевизионном вещании.

Flussonic Media Server поддерживает платы DekTec для приёма MPEG-TS по ASI, как описано ниже, а также может отправлять потоки по ASI.

Захват ASI с плат DekTec во Flussonic

Для того, чтобы настроить приём потока по ASI, настройте параметры карты захвата в **dvb_card** и создайте поток через **mpts-dvb://** URL:

```
dvb_card asi {
  hw dektec_asi;
  serial 2174220026;
  port 1;
}
stream asi_1 {
  input mpts-dvb://asi?program=1020;
}
```

, где:

- **hw dektec_asi** указывает на использование модуля `dektec_asi` для захвата потока;
- **serial** — серийный номер карты захвата;
- **port** — номер порта.

Посмотреть список подключенных DekTec-устройств можно с помощью утилиты *DekTec DtInfoCL*, которую Вы можете скачать с сайта Dektec.

4.2.4 Отправка потока на DekTec ASI

Flussonic Media Server может не только захватывать, но и передавать MPTS-поток на плату захвата и вывода DekTec ASI.

Для вывода ASI на DekTec используйте директиву `push dektec-asi://serial_number:port` в конфигурации мультиплексора:

```
transponder mux1 {
  bitrate 10000k;
  push dektec-asi://2174207373:3;
  program 1 {
    source channel_1;
    title test;
    lcn 0;
    service_type digital_tv_avc_hd;
    pid 5032 pmt  pmt;
    pid 5033 a1;
    pid 5034 v1 bitrate=6000 pcr;
    pid 5035 l1;
  }
}
stream channel_1 {
  input fake://fake;
}
```

4.2.5 Stream Labs SDI

Flussonic Media Server может получать видео и аудио с плат захвата Stream Labs с поддержкой Video4Linux.

Захват с платы захвата Stream Labs

Настройте поток следующим образом:

```
stream example {
  input v4l2:// video_device=/dev/video0 audio_device=plughw:1,0;
  transcoder vb=1000k;
}
```

Если SDI плата работает через v4l2, то установите пакет sdi-v4l-bridge

```
apt install sdi-v4l-bridge;
service flussonic restart
```

4.2.6 Чтение телетекста из VBI

Flussonic позволяет читать EBU Телетекст и субтитры (EBU Teletext subtitle data) из *VBI* (Vertical Blanking Interval) потоков, полученных с SDI-платы, и передавать их в видео, которое вы отправляете в MPTS или SPTS.

VBI — это перерыв в последовательности строк, который используется в аналоговом телевидении. Во время VBI информация об изображениях не передается. Это время используется для возвращения в верхнюю часть экрана луча кинескопа в телевизорах с электронно-лучевыми трубками. Область VBI не видна, но она содержит информацию, необходимую для синхронизации изображений. Она также может содержать такую информацию, как телетекст или скрытые субтитры.

При получении видеопотока с SDI-платы *Flussonic*:

- Декодирует полученные данные.
- Считывает из VBI информацию о телетексте.
- Сжимает данные для последующей передачи по Интернету.
- Упаковывает поток с телетекстом в MPEG-TS.

Включение чтения телетекста

Flussonic позволяет описать информацию о телетексте в таблице PMT (Program Map Table) итогового MPEG-TS потока, двумя способами:

- В Flussonic UI
- В конфигурационном файле

Если вы используете карты захвата Stream Labs для приёма телетекста, то укажите `vbi_device=/dev/vbiN`. Из устройства `/dev/vbiN` *Flussonic* будет получать данные телетекста, например `vbi_device=/dev/vbi0`.

В Flussonic UI

- На вкладке **Media** > **Streams** откройте настройки потока, нажав на имя потока.
- На вкладке **Input** откройте настройки захвата, нажав **Options** справа от URL источника.
- Найдите раздел **Teletext descriptors** и укажите параметры телетекста: номер страницы телетекста (page), язык телетекста (language), тип страницы телетекста (page type) и, если необходимо, дополнительные параметры (extra source params).
- Чтобы сохранить настройки, нажмите **Save**.

Figure 4.11: Teletext descriptors UI

В конфигурационном файле

- Откройте конфигурационный файл `flussonic.conf`.
- Укажите в настройках выбранного потока опцию `ttxt_descriptors` с параметрами телетекста следующим образом:

```
ttxt_descriptors=page:lang:type[,page:lang:type]...
```

Настройка `ttxt_descriptors` определяет, что будет прописано в опциях телетекст-трека. Эти данные передаются в таблице PMT потока MPEG-TS. По умолчанию используется значение `0x100:rus:initial`.

Пример: `ttxt_descriptors=0x100:rus:initial,0x888:rus:subtitle`

Flussonic самостоятельно определяет те страницы телетекста, данные с которых должны быть отмечены в PMT как субтитры.

Параметры телетекста В настройке `ttxt_descriptors` укажите следующие параметры:

- `page` — номер страницы телетекста. Получите информацию о страницах у поставщика потока. Укажите значение `page` в следующем формате: `0x[teletext_magazine_number][telete`
  где `teletext_magazine_number` — номер журнала телетекста от нуля (0) до семи (7), а `teletext_page_number` — номер страницы телетекста в шестнадцатеричной системе счисления от `00` до FF. Например, `0x288` указывает на страницу 88 второго (2) журнала.
- `lang` — язык телетекста. Указывается в соответствии со стандартом ISO 639-2.
- `type` — тип страницы телетекста, определённый в соответствии со спецификацией Specification for Service Information (SI) in DVB systems, 6.2.32 Teletext descriptor в EN 300 468 Digital Video Broadcasting (DVB). В таблице PMT поддерживаются следующие типы страниц телетекста:

Flussonic не поддерживает страницы `additional` и `program_schedule`. Если они вам нужны, сообщите нам на support@flussonic.com.

Примеры конфигураций для чтения телетекста из разных источников

Ниже указаны примеры конфигураций *Flussonic Media Server* для чтения телетекста из разных источников:

- Из потока с карты Stream Labs:

```
stream example_stream {
    input v4l2:// audio_device=plughw:1,0 ttxt_descriptors=0x100:rus:
   initial,0x888:rus:subtitle vbi_debug=true vbi_device=/dev/vbi0
   video_device=/dev/video0;
}
```

- Из потока с карты Decklink:

```
stream example_stream {
    input decklink://0 pixel=10 ttxt_descriptors=0x100:rus:initial,0x888:
   rus:subtitle;
}
```

- Из потока с карты Decklink с NVENC-кодированием:

```
stream example-stream {
  input decklink://2 pixel=10 ttxt_descriptors=0x100:rus:initial,0x888:rus:
    subtitle;
  transcoder deviceid=0 external=false hw=nvenc vb=5000k vcodec=h264
    open_gop=false preset=veryfast size=3840x2160:fit:#000000 ab=128k
    split_channels=false;
}
```

Flussonic по умолчанию считывает телетекст из потока MPEG-TS. Вы можете переписать значение дескриптора телетекста в настройках потока, если это необходимо. *Flussonic* затем отправляет эти данные в выходном потоке MPEG-TS.

Чтобы переписать значение дескриптора телетекста в настройках потока, выполните следующие шаги:

- Настройте поток с дорожкой телетекста.
- Укажите новое значение для параметра `ttxt_descriptors` следующим образом:

```
stream teletext_stream {
  input udp://MULTICAST-IP:PORT ttxt_descriptors=0x888:rus:impaired;
}
```

4.2.7 Передача телетекста из MPEG-TS в аналоговое видео

Flussonic Media Server может передавать телетекст из MPEG-TS в аналоговые потоки в SD качестве, которые он передает на SDI карты Decklink и DekTec. Телетекст добавляется в VBI (интервал вертикального гашения) выходного потока, и от вас потребуется указать номера строк VBI, в которых будет передаваться дорожка с телетекстом.

Предварительные условия:

- Входной поток MPEG-TS, содержащий телетекст (**Teletext B**).
- Выходной поток, содержащий SD-видео, которое Flussonic будет передавать на SDI карту Decklink или DekTec.

Чтобы передать одну дорожку с телетекстом в SDI, укажите номера строк, в которых будет упакован телетекст в выходном потоке. Пример для Decklink:

```
stream out {
  input file://vod/mpegts.ts;
  push decklink://1 format=pal vbi_lines=ttxt:7:8:9:319:320:321;
}
```

В примере в опции `vbi` указаны шесть цифр через двоеточия — это номера строк VBI, в которых будет передаваться дорожка с телетекстом. Первые три — это VBI строки, переданные в первом полукадре, а следующие три цифры — это строки во втором полукадре.

Если телетекст не поместился в указанные строки, он не будет передаваться. В этом случае следует указать больше строк в `vbi`.

4.2.8 Чтение скрытых субтитров CEA-608/708 из SDI

Скрытые субтитры (*англ.* Closed captions) — текстовая репрезентация звуковой части ТВ-программы, фильма и т.п. Это своего рода транскрипция или перевод диалога, звуковых эффектов и некоторой ключевой информации, необходимой для понимания происходящего. Изначально они создавались для людей, имеющих ограничения по слуху. Информация в скрытых субтитрах передаётся в самом видеопотоке, зритель может включать или сделать невидимыми скрытые субтитры по необходимости. Разные транспортные протоколы видео поддерживают разные стандарты скрытых субтитров.

Подробнее см.: Субтитры.

Flussonic способен автоматически обнаруживать скрытые субтитры в SDI-потоке и осуществлять их чтение.

Что же делает *Flussonic*? *Flussonic* читает субтитры CEA-608/708 из SDI-источника, упаковывает их в видеодорожку MPEG-TS в виде *H.264 SEI NALU*

> *Файл H.264 состоит из некоторого числа* NAL Units *(Network Abstraction Layer Units), т.е. единиц так называемого уровня сетевой абстракции, а* SEI* (Supplemental Enhancement Information) — это дополнительная информация о расширении.

CAE-608 — это символьный формат скрытых субтитров, используемый в потоках, который предоставляет до 4-ёх каналов для передачи информации. *Flussonic* добавляет 4 текстовых дорожки с субтитрами для 4-ёх каналов CEA-608 (одна дорожка на один канал). В итоге, на выходе мы имеем видеодорожку со скрытыми субтитрами CEA-608/708 и 4 дорожки с текстовыми субтитрами. *Flussonic* даёт возможность проиграть дорожку с субтитрами в виде WebVTT и TTML, сигнализируя об их наличии в HLS, DASH и MSS манифестах.

Flussonic осуществляет захват SDI-потока с помощью плат Decklink SDI, Stream Labs SDI, AJA SDI и Magewell SDI.

Подробнее о работе с платами и их конфигурации во Flussonic см: Decklink SDI, Stream Labs SDI, AJA SDI.

4.3 Own channel

4.3.1 Как создать свой IPTV-телеканал (серверный плейлист)

На этой странице:

- О серверном плейлисте в Flussonic.
- Как сделать свой IPTV-телеканал из видеофайлов.
- Как сделать свой IPTV-телеканал из потоков.
- Управление расписанием вещания.
- Как настроить вставку рекламы для серверного плейлиста.

Как оператор IPTV/OTT вы можете создать свой **FAST (Free Ad-Supported streaming TV)** канал — бесплатный для зрителя ТВ-канал с рекламой. Например, информационный канал, который распространяет информацию для абонентов и рекламирует новые услуги, или канал для вещания фильмов. Такие каналы создаются из VOD-файлов и распространяются бесплатно, но монетизируются за счёт продажи рекламы.

FAST-канал с видеофайлами из локального VOD-хранилища вместо полученных по сети потоков:

- Сэкономит веб-трафик.
- Привлечёт новых абонентов. Зрители не платят за просмотр такого телеканала.

> Убедитесь, что вы соблюдаете права интеллектуальной собственности.

О серверном плейлисте в Flussonic

Собственный канал в Flussonic представляет собой список воспроизведения, или серверный плейлист, со ссылками на источники вещания — файлы и видеопотоки на сервере Flussonic. Серверный плейлист проигрывается циклично и может запускаться по расписанию.

Серверные плейлисты используются для решения следующих задач:

- Вещание телеканала на десятки одновременных зрителей в локальной сети.
- Переключения между потоками с камер, например, раз в минуту.
- Создания платформы цифровой видеорекламы для показа информационных или рекламных роликов.

Недостатки серверных плейлистов У серверных плейлистов есть ряд недостатков при их использовании в Интернете для вставки видео на сайт:

- Невозможность использовать таргетинг при вставке рекламы.
- Невозможность учитывать рекламу через AdRiver и другие подобные инструменты.
- Сложность мультибитрейтной доставки: разные файлы могут иметь разное количество дорожек с битрейтом.
- Технически сложно реализовать перемотку назад, а это — преимущество интернет-доставки по сравнению с эфирной.
- Сложно реализовать паузу.

Из-за невозможности реализации адекватного инструмента учёта рекламы рекомендуется использовать клиентские плейлисты вместо серверных плейлистов. Так абонент IPTV сам формирует плейлист из доступных каналов.

Однако серверные плейлисты можно использовать для решения других задач помимо онлайн-вещания. Практика показывает, что пользователям приятнее смотреть то, что предлагают, а не искать самим.

Как сделать свой IPTV-телеканал из видеофайлов

Чтобы составить серверный плейлист из видеофайлов, следуйте шагам ниже:

1) Подготовьте видеофайлы для трансляции. 2) Настройте VOD-локацию загрузите видеофайлы. 3) Создайте серверный плейлист. 4) Создайте поток с источником 'playlist://'.

> Видеофайлы и потоки в серверном плейлисте должны быть идентичны друг другу по параметрам: видео- и аудиокодеки, разрешение, битрейт.

Шаг 1. Подготовьте видеофайлы Чтобы обеспечить совместимость между устройствами и браузерами, а также хорошее проигрывание контента зрителями, подготовьте видеофайлы. Видеофайлы в серверном плейлисте должны быть идентичны друг другу по своим настройкам: видео- и аудиокодеки, разрешение, битрейт.

Шаг 2. Настройте VOD-локацию 1) По умолчанию в конфигурационном файле `/etc/flussonic/flusson` существует VOD-локация с именем `vod`, которая указывает на путь `/opt/flussonic/priv` для размещения файлов:

```
vod vod {
storage /opt/flussonic/priv;
}
```

либо

```
vod vod {
storage priv;
}
```

В примере используется директория, указанная в `vod`. Если вы хотите разместить файлы в другом месте, то создайте другую VOD-локацию или добавьте каталог в существующую VOD-локацию. Вы также можете сделать это через веб-интерфейс.

1) Загрузите в каталог на сервере видеофайлы для трансляции.

В примере используются файлы `bunny.mp4` и `beepbop.mp4`, которые уже есть в `/opt/flussonic/priv/`.

Шаг 3. Создание серверного плейлиста Плейлист — это текстовый файл со списком ссылок на источники вещания. Для редактирования плейлиста используется nano — редактор для работы с текстовыми файлами в Linux-системах.

1) Установите nano на сервер Flussonic, выполнив в командной строке следующие команды:

```
apt-get update && apt-get install nano
```

2) В одной директории с видеофайлами создайте плейлист `playlist.txt` с помощью следующей команды:

```
nano /opt/flussonic/priv/playlist.txt
```

Файл сразу откроется в редакторе. Добавьте в него ссылки на видеофайлы, которые будете вещать. Ссылка состоит из названия VOD-локации и имени файла:

```
vod/bunny.mp4
vod/beepbop.mp4
```

3) Сохраните изменения и выйдите из редактора, нажав **CTRL + X** и затем **y**.

Шаг 4. Создание потока 1) Создайте статический поток в веб-интерфейсе, зайдя в раздел **Media** в меню навигации слева и нажав **+** возле вкладки **Streams**. Укажите имя потока *Stream name* и URL источника *Source URL (if available)*. URL источника состоит из настройки `playlist://` и пути до плейлиста. Например, `playlist:///opt/flussonic/priv/playlist.txt`, где `/opt/flussonic/priv/playlist.txt` — путь до плейлиста.

Вы также можете создать поток в конфигурационном файле `/etc/flussonic/flussonic.conf` или с помощью API-запроса `Flussonic-API: PUT /streams/{name}`.

2) Если вы редактировали настройки через конфигурационный файл, то перечитайте конфигурацию сервера, выполнив в командной строке Linux следующую команду:

```
service flussonic reload
```

В списке потоков Flussonic появится новый поток, который по кругу будет проигрывать указанные файлы.

Как сделать свой IPTV-телеканал из потоков

Серверный плейлист из потоков похож на плейлист из видеофайлов за исключением того, что вместо видеофайлов VOD-локации указываются потоки.

Чтобы сделать серверный плейлист из потоков, выполните следующие шаги:

1) В каталоге VOD-локации создайте плейлист в виде текстового файла, указав в нём имена потоков на сервере Flussonic для вещания. Для управления расписанием вещания используйте управляющие команды. Пример плейлиста `playlist.txt`:

```
#EXTINF:60
cam1
#EXTINF:60
cam2
```

, где:

- `cam1` и `cam2` — имена потоков на сервере Flussonic для вещания, - `#EXTINF:60` — управляющая команда, задающая продолжительность проигрывания потока в секундах.

Плейлист выше последовательно воспроизводит потоки `cam1` и `cam2`, переключаясь между потоками каждые 60 секунд.

2) В веб-интерфейсе создайте поток с источником `playlist://`, в котором укажите путь к плейлисту. Нажмите **Save**, чтобы сохранить настройки:

Вы также можете создать поток в конфигурационном файле `/etc/flussonic/flussonic.conf` или с помощью API-запроса `Flussonic-API: PUT /streams/{name}`.

Управление расписанием вещания

Чтобы управлять расписанием в плейлисте, используйте следующие команды:

- `#EXT-X-MEDIA-SEQUENCE` — номер первого элемента плейлиста. Используется для правильной ротации и обновления плейлиста.
- `#EXTINF` — продолжительность в секундах проигрывания элемента плейлиста. Может использоваться для вставки прямого эфира.
- `#EXT-X-UTC` — время в UTC, когда требуется начать проигрывание элемента плейлиста.
- `#EXT-X-PROGRAM-DATE-TIME` — время начала проигрывания элемента плейлиста в формате ISO 8601: `2013-02-12T12:58:38Z` по GMT.
- `#EXT-X-CUE-OUT:ID=SOME_ID` — метка врезки рекламы, указывающая на начало блока рекламы.
- `#EXT-X-CUE-IN` — метка врезки рекламы, указывающая окончание блока рекламы.

Figure 4.12: Flussonic playlist

Примеры использования управляющих команд см. в разделе Примеры.

После завершения проигрывания каждого видеофайла плейлист перечитывается.

Учитывайте следующие правила обработки плейлистов:

1) Если указана команда `#EXT-X-MEDIA-SEQUENCE`, то запоминается последний проигранный номер и после перечитывания плейлиста проигрывание продолжается со следующего номера. То есть содержимое нового плейлиста может быть любым, синхронизация будет осуществляться со следующего номера. Если в новом плейлисте все номера меньше, чем последний проигранный, то плейлист будет каждую секунду перечитывать файл, ожидая появления правильного номера.

2) Если команда `#EXT-X-MEDIA-SEQUENCE` не указана и сам файл плейлиста не менялся, то

проигрывается следующий элемент. Если менялся, то проигрывание начинается сначала.

Примеры

- С помощью управляющей команды #EXTINF можно настроить продолжительность проигрывания элементов плейлиста Например, показывать первые 30 секунд одного файла и первые 60 секунд второго:

```
#EXTINF:30
vod/bunny.mp4
#EXTINF:60
vod/beepbop.mp4
```

- С помощью тега #EXT-X-UTC можно задать время в UTC начала проигрывания элемента плейлиста:

```
#EXT-X-UTC:1522839600
vod/bunny.mp4
#EXT-X-UTC:1522843200
vod/beepbop.mp4
```

- С помощью тега #EXT-X-PROGRAM-DATE-TIME можно задать время начала вещания элемента плейлиста в формате ISO 8601:

```
#EXT-X-PROGRAM-DATE-TIME:2018-04-04T11:00:00Z
vod/bunny.mp4
#EXT-X-PROGRAM-DATE-TIME:2018-02-04T12:00:00Z
vod/beepbop.mp4
```

Как настроить вставку рекламы для серверного плейлиста

Процесс создания серверного плейлиста для вставки рекламы в поток без меток схож с созданием плейлиста из видеофайлов за исключением следующего:

1) Требуется подготовить видеофайлы с контентом и рекламные ролики и загрузить их в VOD-локацию.

> Видеофайлы с контентом и рекламные ролики в серверном плейлисте должны быть идентичны друг другу по своим параметрам: видео- и аудиокодеки, разрешение, битрейт.

2) Название плейлиста должно оканчиваться на `.onlyvod.m3u8`. Например, `ad_playlist.onlyvod.m3u8`
Это включает механизм врезки рекламы.

3) Плейлист должен содержать метки начала (#EXT-X-CUE-OUT:ID=SOME_ID) и окончания (#EXT-X-CUE-IN) врезки рекламы. Пример плейлиста:

```
vod/film_1.mp4
#EXT-X-CUE-OUT:ID=126
vod/ad_1.mp4
vod/ad_2.mp4
#EXT-X-CUE-IN
vod/film_2.mp4
#EXT-X-CUE-OUT:ID=127
vod/ad_3.mp4
#EXT-X-CUE-IN
vod/film_3.mp4
```

В HLS- и DASH-плейлистах вы увидите SCTE-метки времени начала рекламы, например:

```
#EXT-OATCLS-SCTE35:/DAgAAAAAAAAAP/wDwUAAAB+f/8AABdmoAABAAAAAG1gDGA=
#EXT-X-CUE-OUT:DURATION=17.040
```

См. также:

- Как наложить свой логотип на видео.
- Как сделать свой канал доступным в локальной сети по протоколу UDP.

4.3.2 Создание телеканала из IP камеры

В этой статье будет рассказано, как добавить видео с IP камеры в MPTS-поток, вещаемый в кабельную или спутниковую сеть. Эта дополнительная возможность станет приятным бонусом для зрителей и может быть полезна, например, при организации телевещания в коттеджном поселке: вы можете отдельным каналом вывести публичную камеру, которая круглосуточно снимает въезд на территорию, или транслировать строительство какого-либо важного инфраструктурного объекта. При желании можно дать зрителям доступ к архиву камеры.

Для добавление камеры в мультиплексор нужно:

- Узнать RTSP URL камеры.
- Создать поток с этим RTSP URL в *Flussonic*.
- Добавить созданный поток в мультиплексор.

Эти действия описаны далее на этой странице.

RTSP URL

Flussonic позволяет подключать любые камеры по RTSP, поэтому вам необходимо выяснить RTSP URL камеры. Как правило, его можно найти в веб-интерфейсе IP камеры. Обратите внимание на следующие моменты:

- В адресе должен быть логин и пароль
- Вам нужен IP адрес камеры, который доступен *Flussonic*.

Обычно RTSP URL имеет вид: `rtsp://admin:4321@192.168.45.32/cam/realmonitor?channel=1&s`
Важно то, что после IP адреса почти всегда есть ещё части пути, без которых камера не будет показывать.

Некоторые камеры включают логин-пароль в путь и тогда URL имеет вид `rtsp://192.168.0.213/user=ad`

Иногда камера находится в закрытой сети, и к ней приходится пробрасывать порты на роутере. В этом случае заходя в веб-интерфейс вы обращаетесь к одному IP адресу и порту, а у камеры другой адрес и порт. Не все камеры это корректно обрабатывают и могут вам предложить RTSP URL с внутренним IP адресом. В этом случае надо исправить адрес и порт на внешние.

> В некоторых случаях для подключения камер из закрытой сети можно использовать *Flussonic Agent*. Подробнее об Агенте читайте в статье Использование Flussonic Agent.

Подведем итог: просто IP-адреса камеры недостаточно для подключения к *Flussonic*. Нужно указать корректный и доступный с сервера RTSP URL.

Создание потока

Теперь надо создать новый поток в *Flussonic*:

- На вкладке **Media** — **Streams** нажмите **+**.
- В форме создания потока укажите его название, а также RTSP URL, о котором написано в разделе выше.
- Нажмите **Create**.
- Будет открыт профиль созданного потока на вкладке **Overview**. Если URL правильный, то вы увидите информацию о битрейте, дорожках, превью потока в плеере и т.д.
- При необходимости можно задать настройки, специфичные для RTSP потоков. Для этого перейдите на вкладку **Input** и щелкните **Options** рядом с URL потока. Настройки RTSP включают:
- Чтобы включить запись архива, перейдите на вкладку **DVR** в профиле потока.

См. также:

- Примеры конфигурации при добавлении RTSP-потока через файл конфигурации.
- Альтернативные варианты использования потока с камеры — вставка видео на сайт (embed.html), видео-скриншоты.

Добавление потока с камеры в мультиплексор

Добавление потока с камеры в мультиплексор ничем не отличается от добавления любого другого потока. Это можно сделать из интерфейса:

- Перейдите на вкладку **Media** — **Multiplexers**.
- Найдите требуемый мультиплексор или создайте новый.
- В поле **Add program** выберите или введите название созданного ранее потока с камеры.
- Нажмите **+**.

Готово, теперь видео с камеры будет добавлено в мультиплексор. Обязательные настройки, такие как номер программы (PNR), номер трека (PID) и т.д. задаются автоматически. При необходимости вы можете развернуть настройки и задать все параметры программы и мультиплексора, см. Конфигурирование мультиплексора в UI.

4.4 Services

4.4.1 Добавление EPG в MPTS

Flussonic может формировать MPTS с встроенным в него электронным расписанием передач EPG (Electronic Program Guide). При этом вам не потребуется генератор EPG и ремультиплексор, чтобы добавлять расписание к программам.

В MPEG-TS потоках EPG записывается в таблицы служебной информации транспортного потока Event Information Tables (EIT). *Flussonic* умеет формировать EIT в выходном потоке.

Flussonic может добавить EPG в выходной MPTS двумя способами:

- Копировать EIT из источника, если имеющаяся там программа передач вас устраивает.
- Забирать расписание из файлов в формате XMLTV и конвертировать его в таблицы EIT. В транспортный поток передаётся EIT как для текущего мультиплексора (Actual) так и для других (Other). EPG упаковывается в нужный битрейт, и в UDP мультикаст уходит поток с готовым расписанием.

Копирование EPG из источника

Чтобы скопировать EIT из входного MPEG-TS потока, добавьте в конфигурацию мультиплексора опции EIT следующим образом:

```
transponder tp1 {
  eit {
    source copy://STREAMNAME;
  }
}
```

Вместо STREAMNAME укажите имя исходного потока в *Flussonic*.

Импорт EPG из XMLTV

Как настроить импорт EPG в мультиплексор:

- В настройки мультиплексора нужно добавить опции EIT для импорта EPG из XMLTV файлов.
- При каждом обновлении данных в XMLTV файле необходимо перезагрузить файл в мультиплексор API командой.
- Если номер версии EIT хранится в отдельном файле, то при каждом обновлении данных в XMLTV файле может потребоваться обновить номер версии в файле.

Настройка мультиплексора для передачи EPG Рассмотрим пример:

```
transponder tp1 {
  program 100 {
    title "program1";
    eit_title "example_title";
  };
  other @tp2;
  eit {
        xmltv_url xmltv_dir1;
        interval pf actual=1 other=2;
        interval schedule other=20;
  }
}
```

, где:

- `title` — идентификатор канала, в котором указывается значение `channel id` из XMLTV.
- `eit_title` — имя ТВ-канала в XMLTV-файле. Чтобы связать ТВ-программу из XMLTV-файла с конкретным потоком в MPTS, необходимо указать значение `eit_title` равное `display-name` в XMLTV.
- `xmltv_url` — путь к каталогу с XML файлами. Может быть просто именем файла, например, xmltv_url /path/to/xmltv.xml.
- `interval pf|schedule actual=<INTERVAL 1> other=<INTERVAL 2>` — периодичность отправки таблиц. Расписание передаётся в двух таблицах: текущая/следующая передача (pf) в одной таблице и расписание на несколько дней (schedule) — в другой.

Если установить `interval` равный 0, таблица передаваться не будет.

По умолчанию интервал отправки для `actual` (present/following) — 2 с, `other` (present/following) — 4 с, `actual` (schedule) и `other` (schedule) — 60 с.

> Чтобы передавать большой объем данных EPG, можно увеличить `interval` и/или уменьшить количество программ.

Перезагрузка расписания в мультиплексор При обновлении данных расписания в файле их необходимо передать в мультиплексор.

Для того, чтобы Flussonic загрузил обновлённые XMLTV данные из каталога `xmltv_url`, нужно выполнить вызов API: `Flussonic-API: POST /streamer/api/v3/multiplexers/{name}/xmltv_u`

Для нашего примера:

```
/streamer/api/v3/multiplexers/tp1/xmltv_upload
```

Версия EIT таблицы При обновлении расписания у EIT меняется версия — число от 0 до 63.

Замечания Файл XMLTV с сайта teleguide.info может содержать перекрывающиеся по времени передачи. Если такое встречается, то Flussonic более раннюю передачу добавит в EPG, а более позднюю исключит.

4.5 IP Output

4.5.1 Подготовка DVB-совместимого CBR-потока для одного телеканала

Из любого 7Cinputstag/stream/operation/stream-get/response%7Cinputsподдерживаемого Flussonic источника Flussonic Media Server подготовит SPTS для передачи в DVB-среду, требующую жесткой упаковки видеосигнала в полосу с постоянным битрейтом. Для этого поток транскодируется на CPU и упаковывается в MPEG-TS с соблюдением требований стандарта ETSI TR 101 290.

> Транскодирование на NVENC не позволяет добиться стабильности битрейта (CBR), достаточной для требований стандарта DVB. Мы предлагаем только кодирование на центральном процессоре (CPU).

Допустим, вам нужно из HLS-потока подготовить SPTS с разрешением Full HD и общим битрейтом 6 700 Кбит/с, в котором аудио PID занимает 192 Кбит/с, а видео — 6 100 Кбит/с. Здесь килобит — это 1000 бит, а не 1024. Вы можете настроить Flussonic Media Server через веб-интерфейс или конфигурационный файл.

В веб-интерфейсе

Шаг 1. Настройте транскодирование потока в CBR 1) Откройте настройки потока, нажав на имя потока на странице **Streams**.

2) Перейдите на вкладку **Transcoder** и нажмите **Enable Transcoder**.

3) В разделе **Video** выберите **Target** `HD Television`.

4) Примените настройки, нажав **Save**.

Шаг 2. Отправьте поток в мультикаст-группу 1) В настройках потока перейдите на вкладку **Output** в раздел **Push live video to certain URLs** и укажите адрес мультикаст-группы следующим образом: `udp://239.172.0.1:1234`.

2) Примените настройки, нажав **Save**:

В конфигурационном файле

Шаг 1. Настройте транскодирование потока в CBR В настройках потока укажите `transcoder` и настройте его следующим образом:

```
stream spts-cbr {
  input file://vod/bunny.mp4;
  transcoder target=hdtv;
}
```

, где `target=hdtv` — включение списка пресетов, позволяющих создать видеопид, идеально укладывающийся в MPEG-TS DVB. Для hdtv дефолтом будет 1920x1080 и битрейт 3 мегабита.

Figure 4.13: SPTS CBR transcoder settings

Их можно поменять.

Flussonic сам добивает поток (stuffing) до нужного целевого битрейта.

Шаг 2. Отправьте поток в мультикаст-группу В настройках потока добавьте `push` и укажите адрес мультикаст-группы: `push udp://239.172.0.1:1234;`. Все битрейты флюссоник заполнит самостоятельно.

Figure 4.14: SPTS CBR push udp settings

Шаг 3. Проверка качества выходного SPTS в DVB Inspector

Для этого используйте специальную утилиту, например, DVB Inspector (см. Проверка в DVB Inspector) или любую программу, которая проверяет поток на соответствие стандарту ETSI TR 101 290.

1) Запишите отрезок потока длительностью пару минут с помощью следующей команды в терминале: `/opt/flussonic/contrib/multicast_capture.erl udp://239.172.0.1:1234 spts` Закончите запись, использовав сочетание клавиш *Ctrl+C*.

2) Скачайте получившийся отрезок `spts-cbr-output.ts` на локальный компьютер.

3) Проверьте записанный отрезок в DVB Inspector.

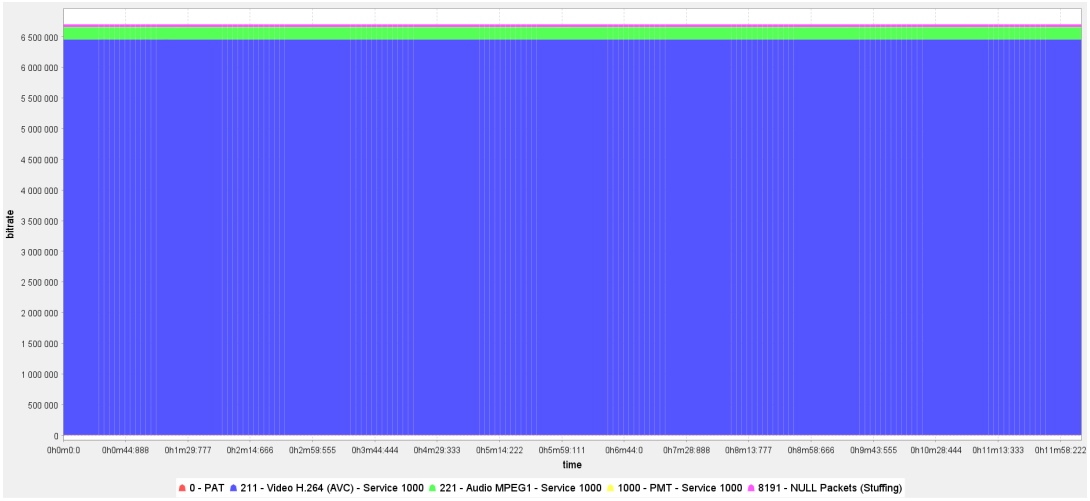


Figure 4.15: DVB Inspector SPTS CBR BitRate View

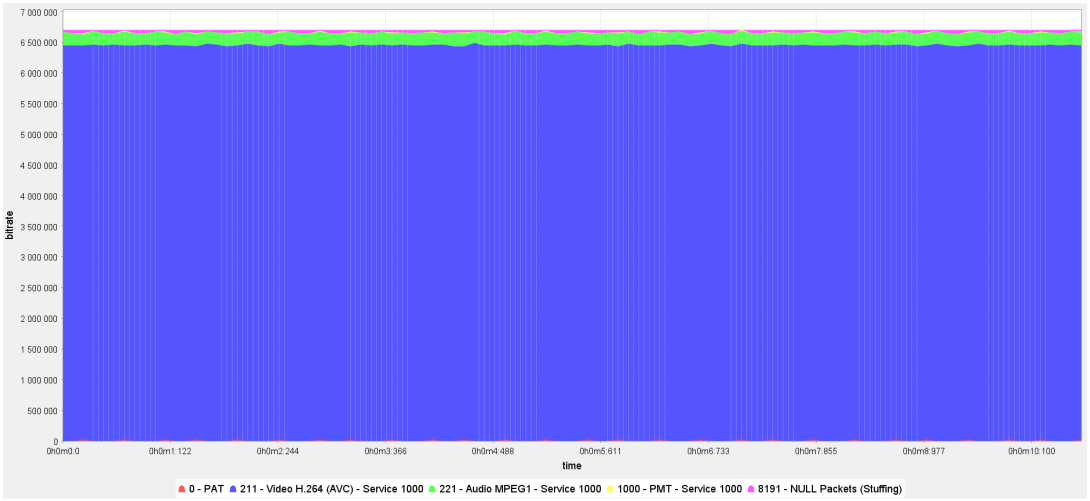


Figure 4.16: DVB Inspector SPTS CBR BitRate View bad ex

В результате вы получите SPTS с постоянным битрейтом (CBR), который можно передать:

- в QAM-модулятор или скремблер для отправки в кабельную сеть,
- в мультиплексор для подготовки MPTS.

Значение по умолчанию шага замера в DVB Inspector (**View** > **Filter** > **Steps**) сглаживает битрейт отдельных PID при большой длительности отрезка. Это значит, что чем больше длительность отрезка записываемого потока, тем ровнее битрейт отдельных PID. Так, если вы запишете отрезок потока длительностью 10 минут и откроете его в DVB Inspector, то увидите идеальный график с ровными битрейтами отдельных PID. Если на этом же отрезке вы установите шаг замера равным 500, то увидите, что в битрейтах отдельных PID есть небольшие колебания. Такие небольшие колебания около одного Кбит/с не влияют на результат и такой поток всё ещё CBR.

4.5.2 Отправка SPTS по мультикасту

При работе с IPTV часто приходится иметь дело с видео, передающимся мультикастом. В подавляющем большинстве случаев, в мультикасте передается MPEG-TS контейнер (по 7 188-байтных пакетов в одном UDP пакете). Реже в сеть передается RTP протокол, внутри которого идет тот же MPEG-TS. RTP нужен для того, чтобы можно было отслеживать потери. В RTP пакете есть 16-битный счетчик, использующийся для отслеживания порядкового номера.

Краткие основы мультикаста

Мультикаст — это UDP пакеты, передающиеся от одного источника группе подписчиков. Адрес, по которому посылаются такие пакеты, обычно находится в диапазоне от 224.0.0.0 до 239.255.255.255, однако 224.0.0.0/8 не рекомендуется из-за большого количества специализированных адресов.

В правильно настроенной сети мультикаст-трафик идет до ближайшего роутера, а роутер уже сам выбирает, какому клиенту какой трафик послать на основании пожеланий клиентов. Пожелания передаются по протоколу IGMP, по которому передаются сообщения о включении в группу рассылки какого-то адреса или исключения из группы.

Таким образом, для того чтобы *Flussonic* рассылал мультикаст клиентам, надо чтобы он слал пакеты в нужный интерфейс (локальная сеть оператора), а роутер был настроен на правильную работу с мультикастом.

Обратная ситуация с захватом мультикаста описана в статье: Прием мультикаста

Предварительные требования

Убедитесь в том, что *Flussonic* синхронизирует время с NTP-сервером. Это важно для стабильного формирования мультикаста: временные метки во входном и выходном потоке будут синхронизированы.

Настройка Flussonic

Настроить мультикаст-рассылку можно в конфигурационном файле либо через веб-интерфейс.

Для настройки мультикаст-рассылки в конфигурационном файле укажите опцию `push` в конфигурации потока и IP-адрес, куда отправлять MPEG-TS поток:

```
stream origin {
  input fake://fake;
}
stream example {
 input hls://localhost:80/origin/index.m3u8;
 push udp://239.0.0.1:1234;
}
```

Для настройки мультикаст-рассылки через веб-интерфейс:

- Создайте поток в веб-интерфейсе и укажите адрес источника на вкладке **Input**.
- Перейдите на вкладку **Output** в настройках потока и укажите URL для мультикаст-рассылки (например, `udp://239.0.0.1:1234`) в разделе **Push live video to certain URLs**.
- (Необязательно) Укажите значения параметров 7Cbody%7Cpushes%7Cvbtag/stream/operation/stream-save%7Cbody%7Cpushes%7Cvbaverage bitrate, 7Cbody%7Cpushes%7Cbitratetag/stream/operation/stream-save%7Cbody%7Cpushes%7Cbitratebitrate, 7Cbody%7Cpushes%7Cpmttag/stream/operation/stream-save%7Cbody%7Cpushes%7CpmtPMT, 7Cbody%7Cpushes%7Cpnrtag/stream/operation/stream-save%7Cbody%7Cpushes%7CpnrPNR, 7Cbody%7Cpushes%7Cstandbytag/stream/operation/stream-save%7Cbody%7Cpushes%7Cstandbystandby, 7Cbody%7Cpushes%7Cmulticast-looptag/stream/operation/s save%7Cbody%7Cpushes%7Cmulticast-loopmulticast loop для выходящего MPEG-TS потока. Для этого кликните на кнопку **Options** и перейдите к редактированию параметров. Укажите необходимые значения для параметров и кликните **Save**, чтобы применить настройки. Это также можно сделать в настройках шаблона конфигурации потоков.

Выбор дорожек Можно выбрать дорожки на отправку:

```
stream origin {
  input fake://fake;
}
stream example {
 input hls://localhost:80/origin/index.m3u8;
 push udp://239.0.0.1:1234?tracks=v1a1;
}
```

Где:

- `v1` — это первая видеодорожка;
- `a1` — это первая аудиодорожка.

Maximum bitrate *Flussonic* может отправлять мультикаст с заполнением значения `maximum bitrate` (максимальный битрейт) в PMT (*англ.* Program Map Table, таблица структуры программ) для каждого ES (*англ.* Elementary Stream, элементарный поток). Для того, чтобы включить эту опцию, необходимо вставить строку `es_max_bitrate=default` в строку запроса (*англ.* query string). Таким образом, конфигурация может иметь следующий вид:

```
stream example {
  input file://vod/STREAM_NAME.ts;
  push udp://239.0.0.1:1234?cbr=6000&tracks=v1a1&es_max_bitrate=default;
  transcoder vb=5000k fps=25 preset=fast hw=cpu ab=192k;
}
```

Указание интерфейса Если IP адрес интерфейса для рассылки мультикаста неудобно указывать по какой-то причине (например, он менялся и вы его не помните), то можно указать его название:

```
push udp://eth0@239.0.0.1:1234
```

вместо

```
push udp://239.0.0.1:1234/10.0.0.5
```

Пример:

```
stream example {
  input hls://provider.iptv/stream/index.m3u8;
  push udp://eth0@239.0.0.1:1234;
}
```

Здесь `eth0` — это название интерфейса, к которому подключена локальная сеть.

Возврат потока, отправленного в мультикаст, на отправляющий хост Flussonic Если вы отправляете поток из *Flussonic* в UDP мультикаст рассылку, можно использовать опцию мультикаст-сокета `multicast_loop`, которая включает прием отправленных UDP данных на хосте-отправителе:

```
stream example_push {
  input hls://provider.iptv/stream/index.m3u8;
  push udp://239.0.0.1:1234 multicast_loop;
}

stream example_ingest {
  input udp://239.0.0.1:1234;
}
```

Опция позволяет захватывать отправленный поток обратно на *Flussonic* средствами *Flussonic* или с помощью какого-либо другого приложения.

Настройка сервера

После того, как вы настроили мультикаст-вещание потока, с большой вероятностью ничего не заработает, потому что часто из-за настроек сервера мультикаст-трафик пойдет в первый интерфейс, который как правило смотрит в интернет. Таким образом, необходимо, чтобы *Flussonic* начал отправлять трафик в тот интерфейс, который смотрит в локальную сеть.

```
route add -net 239.0.0.0/8 dev eth2
```

Где `eth2` — это название интерфейса, к которому подключена локальная сеть. После такого прописывания роутинга мультикаст с *Flussonic* польется в нужный интерфейс и его можно будет увидеть на роутере, а следовательно и на клиенте.

Настройка PID-ов

При отправке MPEG-TS в UDP мультикаст (`push udp://`) для указания PID-ов используйте опцию mpegts_pids.

По-другому можно указать PID-ы так:

```
stream example {
  input hls://provider.iptv/stream/index.m3u8;
  push udp://239.1.2.4:1235 bitrate=7000 pnr=2 vb=6000 pmt=2000 v1=2011 a1
    =2021;
}
```

Сигнализация AC-3 аудиопотока в MPEG-TS MPEG-TS, содержащий элементарные AC-3 аудиопотоки, регулируется моделью STD (System Target Decoder) в System A (ATSC) или System B (DVB). Форматы сигнализации в System A и System B существенно различаются. Поэтому уникальная идентификация (сигнализация) аудиопотоков AC-3 используется не только для однозначного указания на то, что поток AC-3 действительно является AC-3, но и на то, к какой системе (A или B) он принадлежит.

Flussonic может прочитать различные форматы сигнализации AC-3 аудиопотока в PMT MPEG-TS и пробросить исходный формат на выход MPEG-TS в UDP мультикаст или изменить его в соответствии с System A или System B. Это делается с помощью опции `mpegts_ac3`. Она указывается в настройках потока/шаблона и может принимать следующие значения:

- `mpegts_ac3=keep` — сохранить исходный формат сигнализации об AC-3 аудиопотоке и пробросить его на выход MPEG-TS;
- `mpegts_ac3=system_a` — изменить исходный формат сигнализации об AC-3 аудиопотоке на соответствующий System A;
- `mpegts_ac3=system_b` — изменить исходный формат сигнализации об AC-3 аудиопотоке на соответствующий System B;

> Если у вас на входе и на выходе аудиопоток в формате AC-3, то вам не нужно включать транскодирование, чтобы использовать опцию `mpegts_ac3`.

Рассмотрим следующий пример:

```
stream example-stream {
  input udp://MULTICAST-IP-1:PORT-1 programs=2;
  push udp://MULTICAST-IP-2:PORT-2 bitrate=6800 mpegts_ac3=keep;
}
```

Здесь *Flussonic* захватывает UDP-поток с AC-3 аудиопотоком и отправляет UDP-поток с исходным форматом сигнализации аудио на выход.

4.5.3 Отправка потоков в ATSC-C с помощью TBS-карты

Flussonic Media Server позволяет отправлять потоки в кабельные сети ATSC-C без необходимости использовать дополнительные модулирующие устройства. В этом случае TBS карта (сейчас мы поддерживаем TBS6014) работает как генератор сигнала для модуляции потока MPTS (мультиплексора), т.е. конвертации его в сигнал QAMB, который затем можно отправить в кабельную сеть для телевизионного вещания.

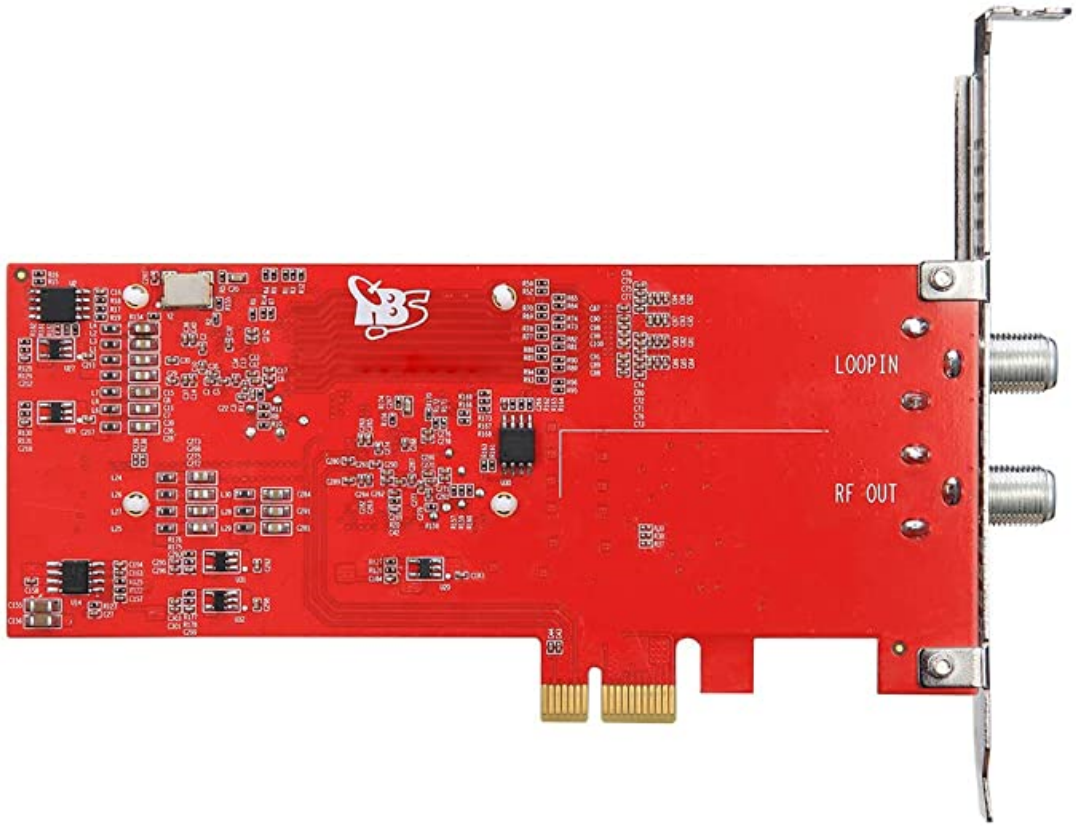


Figure 4.17: TBS card

Для настройки такого пушера сначала добавьте в конфигурацию DVB карту со следующими обязательными параметрами:

- `hw` – модель устройства, должно быть задано значение `tbs6014`
- `adapter` – номер адаптера
- `port` – номер порта

Пример:

```
dvb_card tbsmod01 {
  hw tbs6014;
  adapter 0;
```

```
  frequency 62000000;
  modulation qam256;
  interleave 3;
  gain 5;
  port 0;
  input_bitrate 38;
}
```

Другие необязательные параметры:

- `frequency` — несущая частота мультиплексора для данной программы в Гц.
- `modulation` — тип модуляции TBS.
- `interleave` — использовать перемежение битов в битовом потоке, чтобы минимизировать искажения при передаче данных.
- `gain` — усиление выходного сигнала до указанного значения в дБ.
- `input_bitrate` — битрейт на входе, в Мбит/с.

Вы также можете добавить DVB карту через веб-интерфейс. Для этого перейдите в раздел **Media** > **DVB cards** и нажмите **ADD DVB card**. Затем для параметра Hw выберите значение `tbs6014` и введите остальные настройки.

Теперь настройте мультиплексор с опцией `push dvb://tbsmod01`. Например:

```
stream channel1 {
  input udp://239.0.0.1:1234;
}
stream channel2 {
  input udp://239.0.0.2:1234;
}
transponder newMultiplexer1 {
  bitrate 26970k;
  push dvb://tbsmod01;
  program 100 {
    source channel1;
    title Channel1;
    lcn 0;
    service_type digital_tv_avc_sd;
    pid 101 pmt  pmt;
    pid 102 v1 bitrate=2000 pcr ;
    pid 103 a1 bitrate=500  ;
  }
  program 200 {
    source channel2;
    title Channel2;
    lcn 1;
    service_type digital_tv_avc_sd;
    pid 201 pmt  pmt;
```

Figure 4.18: TBS card configuration

```
    pid 202 v1 bitrate=500 pcr ;
    pid 203 a1 bitrate=100  ;
  }
}
```

При выборе битрейта мультиплексора учитывайте используемый тип модуляции, т.к. он может накладывать ограничение на скорость передачи данных. Например, для модуляции qam64 максимально возможным битрейтом будет 26,90735 Мбит/с, а для модуляции qam265 — 38,81070 Мбит/с.

4.5.4 Отправка MPTS

Flussonic умеет формировать MPTS-потоки (также называемые мультиплексорами), которые можно подавать на модулятор и отправлять в кабельную сеть, эфир или спутник. Подробнее о том, что такое мультиплексоры и зачем они нужны, читайте на странице Мультиплексор.

> *Flussonic* также может принимать MPTS и разделять его на отдельные STPS.

Формирование выходного MPTS

Для настройки отправки MPTS используется директива `transponder`:

```
stream channel1 {
  input udp://239.0.0.1:1234;
}
stream channel2 {
  input udp://239.0.0.2:1234;
}
transponder tp1 {
  bitrate 6400k;
  ts_stream_id 2;
  provider Flussonic;
  push udp://239.172.0.1:1234 multicast_loop;
  push file://dumpts.ts;
  program 1010 {
    source channel1;
    title Channel1;
    pid 1010 pmt;
    pid 1011 v1 pcr;
    pid 1012 a1 bitrate=150;
    pid 1013 l1;
  }
  program 1020 {
    source channel2;
    title Channel2;
    pid 1020 pmt;
    pid 1021 v1 pcr;
    pid 1022 a1 bitrate=128;
    pid 1023 l1;
  }
}
```

Результат работы такого кода хорошо виден в анализаторе:

Этот поток можно смело подать на модулятор и отправить в кабельную сеть, эфир или спутник.

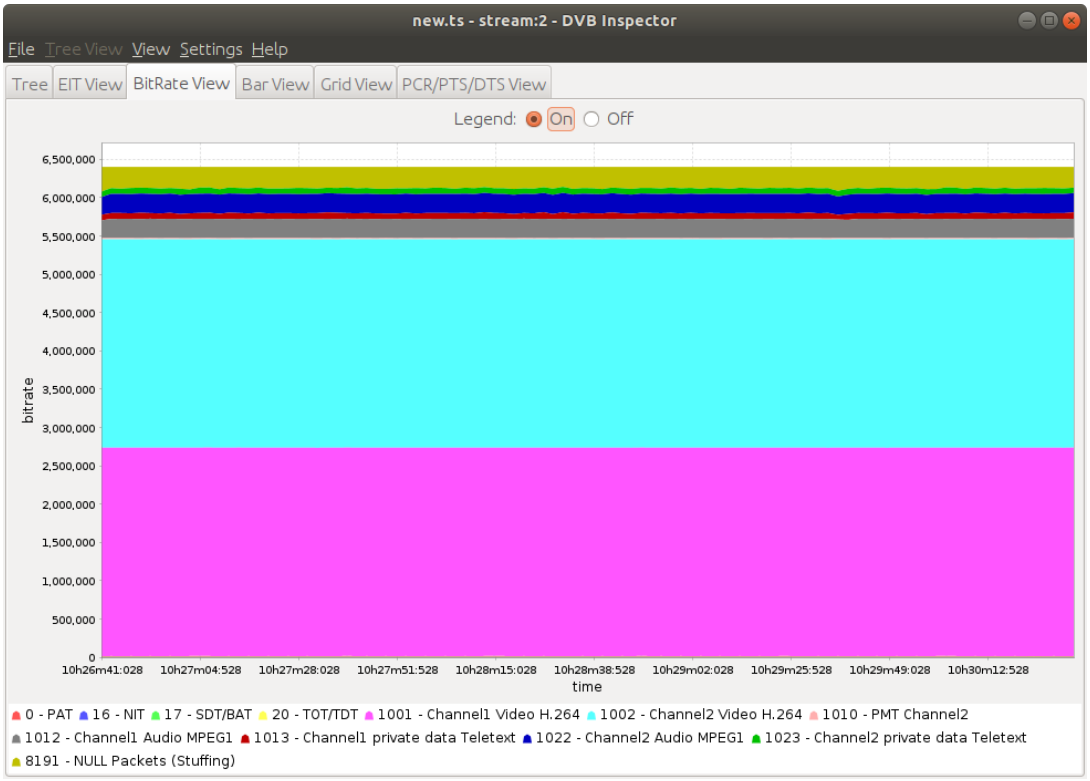


Figure 4.19: CBR MPTS

Выбор выходных треков Входной поток может содержать много дорожек с субтитрами, аудио, видео и т.д. Вероятно, что не все они нужны в выходном MPTS потоке. Вы можете указать, какую дорожку включить в выходной MPTS, назначив ей PID вот так:

```
transponder tp1 {
  bitrate 6400k;
  push udp://239.172.0.1:1234 multicast_loop;
  program 1020 {
    source channel1;
    title Channel1;
    pid 1020 pmt;
    pid 1021 v1 pcr;
    pid 1022 a1;
    pid 1023 l1;
  }
}
```

В MPTS можно использовать видео дорожки (v1, v2, ...), аудио дорожки (a1, a2, ...) и дорожки телетекста (l1, l2, ...). Дорожки с текстовыми WebVTT субтитрами не будут работать в результирующем MPTS.

Только дорожки с назначенным PID будут уходить на мультиплексор или отправляться в UDP,

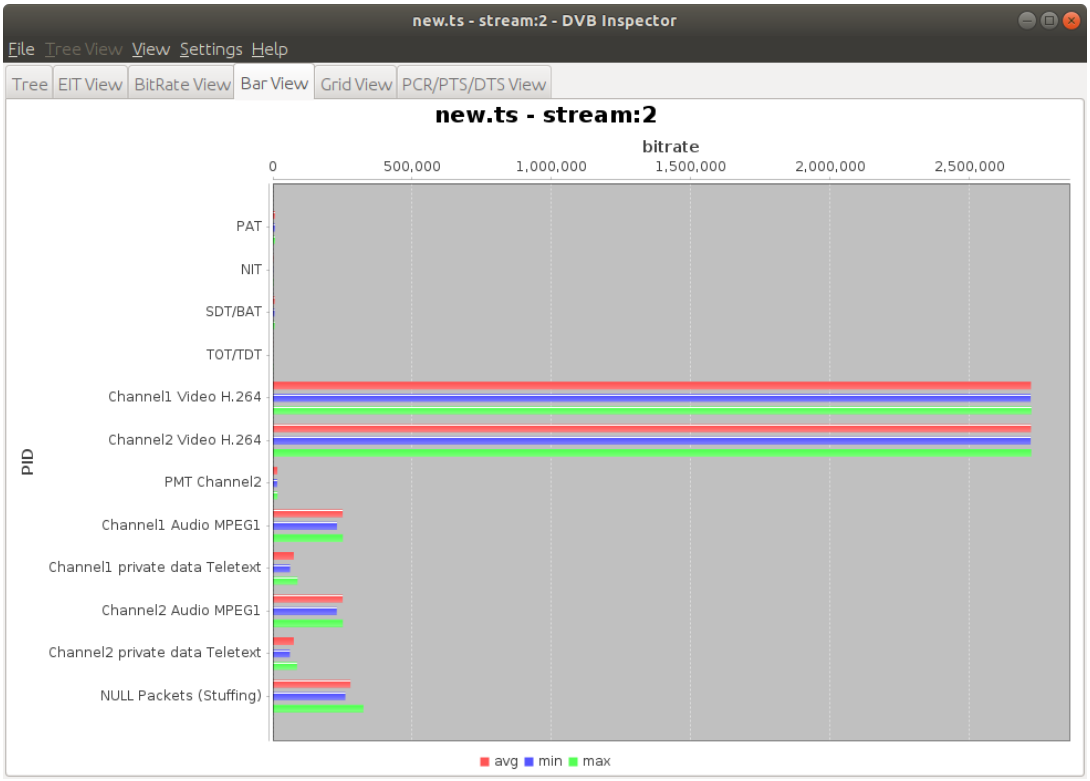


Figure 4.20: CBR MPTS

остальные — не будут. Если дорожки не были указаны, все они будут отправлены в MPTS с автоматически назначенными PID. Но если была указана хотя бы одна дорожка, то другие дорожки не будут включены в вывод.

Это избавляет вас от необходимости создавать вспомогательный поток для приема только тех дорожек, которые вам нужны, а затем отправлять его в MPTS.

Настройки битрейта отдельных дорожек Битрейты дорожек можно указать в настройках мультиплексора следующим образом:

```
transponder tp1 {
  bitrate 6400k;
  push udp://239.172.0.1:1234 multicast_loop;
  program 1020 {
    source channel2;
    title Channel2;
    pid 1010 pmt;
    pid 1021 v1 pcr;
    pid 1022 a1 bitrate=150;
  }
}
```

Такое указание битрейтов позволит вам настроить битрейт для лучшего использования полосы

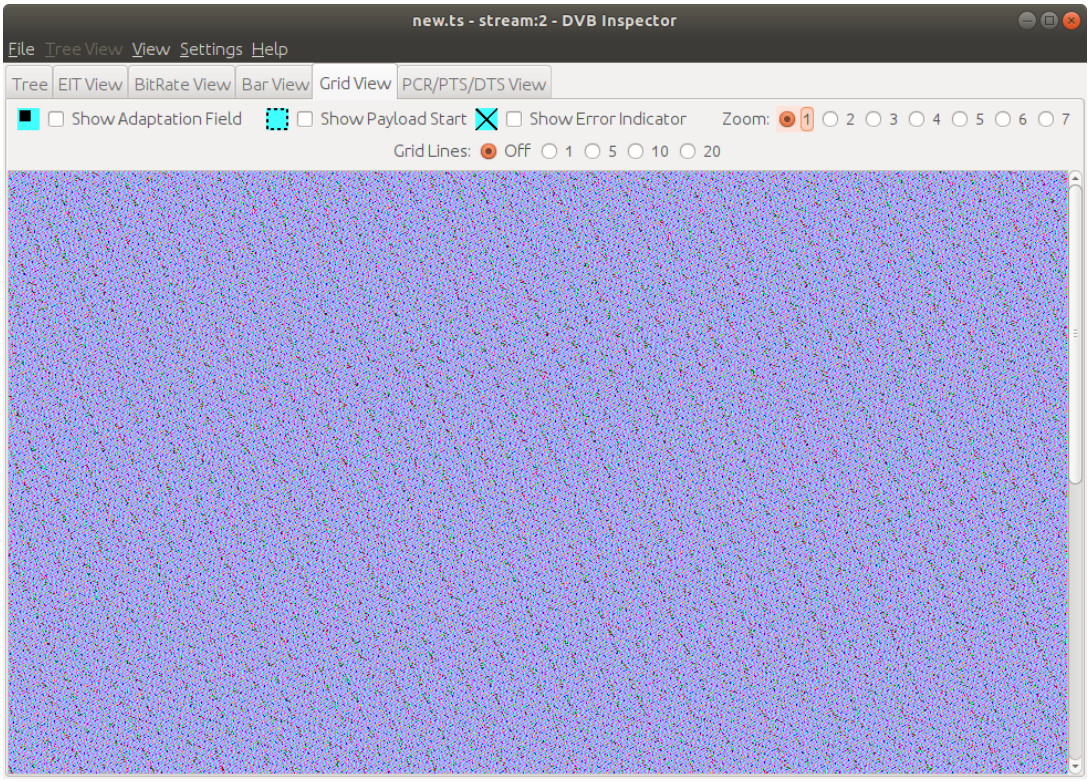


Figure 4.21: CBR MPTS

пропускания, меняя битрейты прямо во время трансляции MPTS.

Буфер для неравномерных источников Если источник, из которого вы формируете MPTS, отправляет данные с большими перерывами между пакетами, например из-за сетевого джиттера, отправка MPTS может прерваться. Чтобы избежать проблем в работе пушера с неравномерным источником, увеличьте параметр `prebuffer` — это значение в миллисекундах, задающее максимальное время ожидания следующего пакета из источника; по умолчанию 300 мс. Обратите внимание, что увеличение этого параметра может привести к увеличению задержки на выходе.

```
transponder tp1 {
  bitrate 7000k;
  prebuffer 800;
  push udp://239.172.0.1:1234 multicast_loop;
  program 1 {
    source channel2;
    title Channel2;
    pid 1010 pmt;
    pid 1021 v1 pcr;
    pid 1022 a1 bitrate=150;
  }
}
```

Расширенные настройки MPTS

Опции SI таблиц Flussonic 20.09 позволяет сформировать более сложные NIT таблицы, содержащие LCN (logical channel number), T2 delivery system descriptor и другие опции. Некоторые из опций передаются также в SDT.

Flussonic реализует вариант настроек в соответствии со спецификацией NorDig Unified Requirements for Integrated Receiver Decoders version 3.1.1, выпущенной NorDig.

В настройки мультиплексора вы можете добавить следующие опции:

- `network 13582 original=8833 name="Example network";` — сеть доставки. Используется один и тот же `original` (original network) для NIT и SDT.
- `ts_descriptor 0x04 04012283;` — тэг и HEX данные любого дескриптора. Это параметр, позволяющий вписать, например, любой ts_descriptor для NIT ts_loop. В примере указаны тег и HEX дескриптора "T2 delivery system descriptor". Добавляется в NIT и SDT.
- `ts_stream_id 2;` — Добавляется в NIT и SDT.
- `service_type` — используется как program P service_type 0x16; Если у программы указан source, но не указан service_type, то Flussonic пытается угадать service_type по пришедшему media_info.
- `timeout service_type 10;` — если за 10 секунд источник не ожил, то вещаем без этой программы в NIT. По умолчанию timeout составляет 15 секунд.
- `program 19 { lcn 7;}` — logical channel number.

Версии PSI-таблиц В PSI-таблицах есть поле `version_number`. При внесении изменения в таблицу текущее значение version_number в ней должно получить инкремент. Это даст сигнал принимающим устройствам (телевизорам) о том, что необходимо перечитать TS содержимое. Flussonic поддерживает указание номера версии в настройке мультиплексора. Таким образом, устройства смогут применить изменения в потоке или подготовиться к ним.

- `version psi VERSION_NUMBER` — глобальная версия для всех PSI таблиц.
- `version sdt VERSION_NUMBER` — версия конкретной SI таблицы, например, SDT.

TOT (time offset table) Flussonic генерирует PSI таблицу TOT (time offset table). Настройки TOT имеет смысл указывать только в основном мультиплексоре, а не в `other`.

Добавьте следующие опции в конфигурацию основного потока:

```
time_offset FRA:1 time_of_change=2018-03-23T03:00:00Z local_time_offset
   =+0100 next_time_offset=+0100;
```

Значения опций приводятся в спецификации

Пример Полный пример конфигурации мультиплексора с расширенными настройками:

```
transponder ts_tp {
  push udp://239.1.2.4:1234 multicast_loop;
  push file://tmp/ts-tp.ts pkt_limit=300000;
  bitrate 27000k;
  provider Flussonic;
  network 123123 original=12345 name="Example network 1";
  ts_stream_id 2;
  ts_descriptor 0x7f 040012340325;
  version psi 4;
  version sdt 9;
  time_offset RUS:7 time_of_change=2018-03-23T03:00:00Z local_time_offset
   =+0600 next_time_offset=+0600;
  timeout service_type 10;
  program 1020 {
    source clock;
    title Channel1;
    lcn 2;
    pid 1120 pmt;
    pid 1121 v1 pcr bitrate=500;
    pid 1122 a1 bitrate=150;
  }
  program 1030 {
    title Channel2;
    lcn 3;
    service_type digital_tv_mpeg2_hd;
  }
  program 1040 {
    title Channel4;
    lcn 4;
  }
  other @02;
}
transponder 02 {
  bitrate 0;
  network 123123 original=12345 name="Another network";
  ts_stream_id 1;
  ts_descriptor 0x7f 040033123325;
  program 1010 {
    lcn 1;
    service_type digital_tv_mpeg2_hd;
  }
  program 1070 {
    lcn 7;
    service_type 0x20;
  }
}
```

Добавление в мультиплексор ссылок на другие мультиплексоры

Для услуги TV необходимо, чтобы были известны все каналы, включенные в услугу, для отображения их на клиентских устройствах. Для этого каждый передаваемый мультиплексор (MPTS поток), а он содержит только часть каналов, должен располагать информацией обо всех остальных передаваемых со спутника каналах на других частотах.

Например, на одной частоте (MPTS потоке, или в мультиплексоре) передается 10 каналов, а всего 40 частот, следовательно, есть 400 каналов, и о них необходимо передать информацию в каждом MPTS потоке. Для этого в настройки каждого потока следует добавить ссылки на другие потоки, составляющие вашу услугу.

Чтобы добавить одну такую ссылку, в опции **other** укажите имя другого MPTS потока.

```
transponder ts_tp {
  push udp://239.1.2.4:1234 multicast_loop;
  push file://tmp/ts-tp.ts pkt_limit=300000;
  bitrate 27000k;
  provider Flussonic;
  network 13582 original=8833 name="Example network 1";
  program 1020 {
    source clock;
    title Channel1;
    lcn 2;
    pid 1120 pmt;
    pid 1121 v1 pcr bitrate=500;
    pid 1122 a1 bitrate=150;
  }
  other @02;
}
transponder 02 {
  bitrate 0;
  network 13582 original=8839 name="Another network";
  ts_stream_id 1;
  ts_descriptor 0x7f 040022830325;
  program 1010 {
    lcn 1;
    service_type digital_tv_mpeg2_hd;
  }
}
```

Мультиплексор, указанный в `other`, отправляется в NIT и SDT.

Резервирование мультиплексора

Вы можете настроить резервирование мультиплексоров таким образом, чтобы при прекращении вещания основного сервера вместо него начинал вещать резервный. Для этого добавьте опцию `standby=true` в директиве `push` резервного мультиплексора. Остальные настройки должны быть идентичны.

Порядок отработки резервирования следующий:

- Оба сервера формируют абсолютно одинаковый набор программ. Основной сервер (без опции `standby`) вещает поток в мультикаст группу. Резервный сервер постоянно проверяет, ведется ли вещание в эту мультикаст группу, но сам не вещает, находясь в ожидании.
- Если вещание прекращается, то резервный сервер начинает вещать сам.
- Как только появится в эфире основной сервер, вещание резервного прекратится.

Пример конфигурации для основного и резервного серверов:

```
transponder main {
  bitrate 6400k;
  push udp://239.172.0.1:1234 multicast_loop;
  program 1020 {
    source channel2;
    title Channel2;
    pid 1010 pmt;
    pid 1021 v1 pcr;
    pid 1022 a1 bitrate=150;
  }
}
```

```
transponder backup {
  bitrate 6400k;
  push udp://239.172.0.1:1234 multicast_loop standby=true;
  program 1020 {
    source channel2;
    title Channel2;
    pid 1010 pmt;
    pid 1021 v1 pcr;
    pid 1022 a1 bitrate=150;
  }
}
```

Конфигурирование мультиплексора в UI

Чтобы добавить мультиплексор, перейдите на вкладку **Media** - **Multiplexers** и нажмите **+**. Задайте имя и битрейт мультиплексера, после этого станут доступны настройки. Щелкайте стрелки, чтобы развернуть или свернуть общие и дополнительные параметры (General settings и Advanced settings).

Затем добавьте программы в мультиплексор:

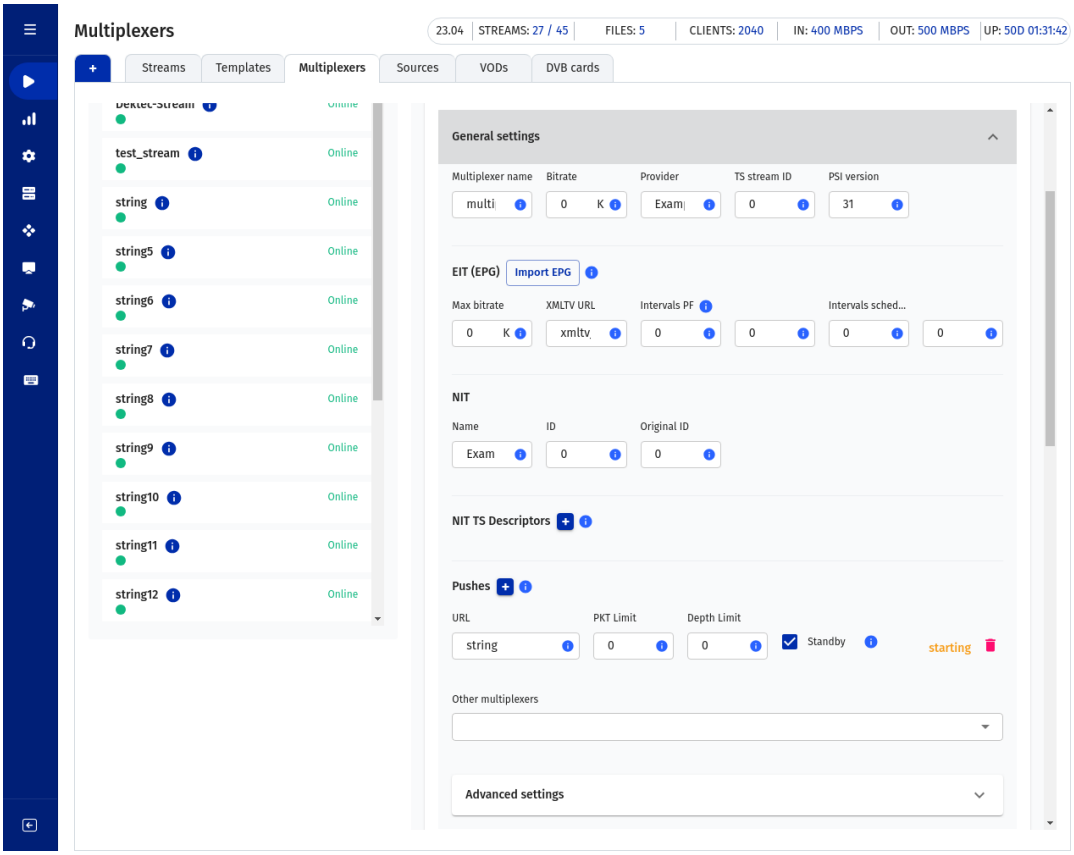


Figure 4.22: multiplexer options

4.5.5 Отправка потока по SRT

Flussonic поддерживает отправку потока по протоколу SRT. Отправка потоков по протоколу SRT широко применяется при доставке видео через Интернет или спутниковую сеть, поскольку SRT гарантирует низкую задержку и при этом предлагает некоторые гарантии доставки контента. Подробнее об SRT см. на странице Использование протокола SRT.

Давайте посмотрим каким образом можно настроить передачу потока по SRT.

Отправка с Flussonic

Вы можете настроить отправку потока по протоколу SRT из *Flussonic Media Server* на сторонний сервер так же, как для любого другого протокола, как описано на странице Отправка потока на другие серверы. В случае SRT необходимо указать URL в одном из следующих форматов:

- Параметры SRT в параметрах URL:
- Параметры SRT в URL query string:

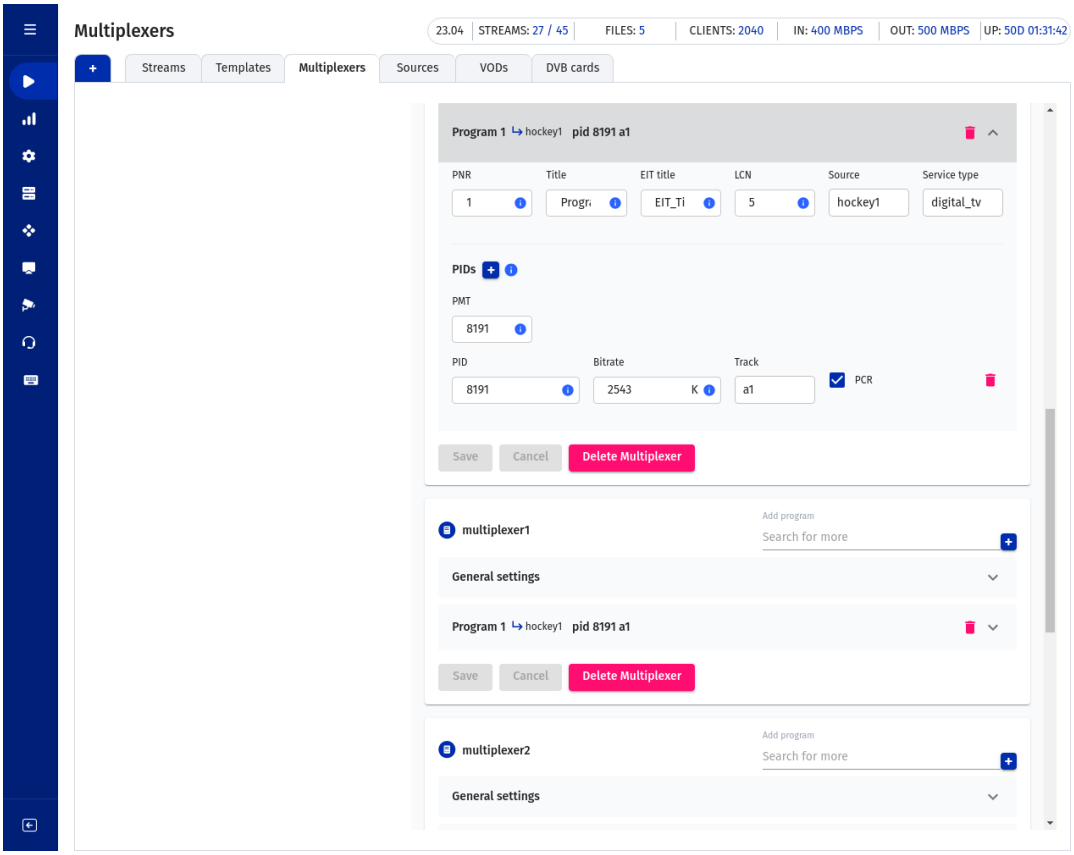


Figure 4.23: multiplexer options

, где:

- `SRT-HOST` — IP-адрес целевого сервера.
- `SRT_PORT` — порт SRT.
- `streamid` — строка, сформированная как описано здесь.
- `"STREAM_NAME"` — имя локации, в которую *Flussonic* будет отправлять SRT-поток.

Рассмотрим пример:

```
stream push_srt {
  input fake://fake;
  push srt://example.com:9998 streamid="#!::r=my-stream-id,m=publish";
}
```

В примере выше мы определили глобальный порт 9998 и настроили отправку потока на сторонний сервер `example.com`.

Параметры для управления отправкой потока по SRT

Вы также можете указать параметры для управления передачей потока по SRT.

Примеры:

```
stream srt_push {
  input fake://fake;
  push srt://example.com:9998?streamid=#!::r=some_random_name&passphrase
   =1234567890;
}
```

4.5.6 Резервирование источника мультикаст-потока

При построении IPTV-сервиса на основе мультикаста возникает задача резервирования источника, когда основной источник становится недоступен и нужно переключиться на работающий источник. В зависимости от того, кто вы: поставщик контента или оператор, зависит то, какую задачу вы решаете:

- Резервирование источника при приёме мультикаст-потока.
- Резервирование источника при отправке мультикаст-потока.

Резервирование источника при приёме мультикаст-потока

Несколько поставщиков передают одинаковый контент мультикастом в точку обмена трафиком, где мультикаст получают десятки операторов. В такой системе возникают ошибки: дублирование данных, избыточный трафик на сервере-приёмнике и другие. Отсюда у владельцев точки обмена трафиком возникает задача — управлять мультикаст-группами так, чтобы оператор подключился только к нужному каналу.

Владельцы точек обмена трафиком могут управлять мультикаст-группами следующими способами:

- Разделить поставщиков по мультикаст-группам: один поставщик — одна мультикаст-группа.
- Разделить каналы по мультикаст-группам: один канал — одна мультикаст-группа.

- Фильтровать трафик по IP-адресу источника на сервере-приёмнике. Так сервер-приемник будет получать избыточный трафик.

- Настроить механизм SSM (Source-Specific Multicast) на основе IGMPv3 на сервере-приёмнике. При подключении к точке обмена трафиком оператор указывает мультикаст-группу и адрес источника, откуда хочет получать трафик. Если запрашиваемый источник недоступен или не работает, то сервер-приёмник автоматически переключится на другой работающий источник (см. Как работает SSM). Для работы SSM (Source-Specific Multicast) оператор должен заранее знать адрес источника. Механизм SSM (Source-Specific Multicast) упрощает управление мультикаст-группами, не создавая дополнительных сложностей в точке обмена трафиком или на приёмнике.

SSM решает следующие задачи:

- Резервирование источника. SSM поддерживает одновременное вещание нескольких источников в одну мультикаст-группу. Так, если два поставщика отправляют контент в одну и ту же мультикаст-группу, то без SSM (Source-Specific Multicast) это приведёт к дублированию данных и путанице на приёме, а с SSM (Source-Specific Multicast) это штатная ситуация.
- Приём мультикаст-потока от поставщика, который требует соответствия стандарту доставки мультикаст-потоков. Некоторые поставщики требуют, чтобы оборудование оператора поддерживало SSM (Source-Specific Multicast) и IGMPv3 для того, чтобы принять мультикаст.

SSM требует поддержки IGMPv3 сетевым коммутатором и приёмником.

Как работает SSM

SSM работает следующим образом:

- В каждом мультикаст-пакете в заголовке передается IP-адрес источника. Основной и резервный источники вещают одновременно в одну и ту же мультикаст-группу.
- Flussonic запрашивает у коммутатора по IGMPv3 пакеты от основного источника с указанием IP-адреса источника.
- Если Flussonic фиксирует потери от основного UDP-источника в течение 7Cinputs%7Csource-timeouttag/stream/operation/stream-get/response%7Cinputs%7Csource-timeoutsource_timeout по умолчанию равному минуте, то он переключается на резервный. Flussonic запрашивает у коммутатора по IGMPv3 пакеты от резервного источника с указанием IP-адреса источника.
- Находясь на резервном источнике, Flussonic каждую минуту опрашивает основной UDP-источник на наличие мультикаст-пакетов.
- Когда основной источник возвращается в эфир и начинает передавать пакеты, то Flussonic переключается на него с резервного источника.

Схема 1. SSM с рабочим основным источником

Схема 2. SSM с выключенным основным источником и рабочим резервным источником

Чтобы настроить SSM (Source Specific Multicast) для мультикаст-потока в Flussonic Admin UI, выполните следующие шаги:

- Перейдите в вкладку **Input** в настройках потока.
- Добавьте основной источник, если он не указан, или измените уже существующий, указав мультикаст-группу, порт и IP-адрес основного источника в следующем виде:

, где:

* `239.1.1.1` — мультикаст-группа, * `1234` — порт, который Flussonic будет слушать, * `10.10.10.1` — IP-адрес источника.

- Добавьте резервный источник, указав данные резервного источника, как в шаге 2.
- Примените настройки, нажав **Save**.

Резервирование источника при отправке мультикаст-потока

Допустим, вы поставщик и у вас есть сервер осуществляющий мультикаст-вещание. Вы хотите доставлять мультикаст-потоки с максимальной надежностью так, чтобы сервер автоматически

переключался на рабочий источник без прерывания вещания. Вы можете зарезервировать этот сервер мультикаст-вещания с помощью потоков Flussonic, вещающих в standby-режиме.

Чтобы принимать мультикаст-поток, оператору нужно подключиться к мультикаст-группе.

> Такой подход работает с любым IGMP на приёмнике, так что вашим клиентам не нужно покупать специальное оборудование.

Как работает standby-режим Standby-режим работает следующим образом:

- Основной и резервный серверы формируют идентичный набор программ. Основной сервер вещает поток в мультикаст-группу.
- Резервный сервер постоянно проверяет, ведется ли вещание в эту мультикаст-группу, но сам не вещает, находясь в standby-режиме (режиме ожидания).
- Если основной сервер прекращает вещание, то резервный сервер выходит из режима ожидания и начинает вещать.
- Как только в эфире появляется основной сервер, резервный прекращает вещание и возвращается в standby-режим.

Схема 3. Standby-режим с рабочим основным сервером

Схема 4. Standby-режим с выключенным основным сервером и рабочим резервным сервером

Чтобы включить standby-режим для мультикаст-потока в Flussonic Admin UI, выполните следующие действия:

- Зайдите в настройки потока и откройте вкладку **Output**.
- В колонке *URL* раздела **Push live video to certain URLs** укажите мультикаст-группу, порт и IP-адрес основного источника следующим образом:

где:

* `239.1.1.1.1` — мультикаст-группа, * `1234` — порт, который Flussonic будет слушать.

- Добавьте резервный источник, указав данные резервного источника, как в шаге 2.
- Откройте **Options** и включите standby-режим для резервного источника, установив флажок *Standby*.
- Примените изменения, нажав кнопку **Save**.

4.5.7 Как передать UDP мультикаст через Интернет с помощью Flussonic?

Описание проблемы

Если вы принимаете спутниковые ТВ-каналы и хотите транслировать их во внешнюю сеть (к примеру, территориально удаленную сеть в другой стране), Flussonic поможет решить эту задачу.

UDP-мультикаст — это лучший способ для организации трансляций в абонентской сети, потому что нагрузка на сервер не меняется с ростом числа подключенных абонентов. Проблема в том, что UDP-мультикастом не получится вещать через открытый Интернет, он работает только в специально сконфигурированной локальной сети.

Организация UDP-мультикаста через Интернет

Решить задачу пересылки UDP-мультикаста через Интернет поможет собственный протокол Flussonic — M4F. Протокол M4F используется для передачи данных между серверами Flussonic. Он имеет преимущества перед другими протоколами.

Figure 4.24: Мультикаст через Интернет

Вам потребуется завести Flussonic Media Server в той сети, где вы собираетесь транслировать UDP-мультикаст. Это будет сервер-приемник.

Таким образом, будем использовать два сервера Flussonic Media Server для передачи мультикаста:

- Сервер-источник получает поток с ТВ-каналом (`channel_01`) со спутника с головной станции по UDP. Здесь достаточно обычной конфигурации.
- На сервере-приемнике настройте поток так, чтобы забирать ТВ-канал с сервера-источника по M4F. Для этого укажите URL такого вида:

```
stream channel_01 {
  input m4f://streamer:8081/channel_01;
  push udp://239.0.0.2:1234;
}
```

Flussonic сам организует получение потока через специальный TCP канал.

Замечание. При передаче на очень большие расстояния неизбежно возникнет задержка. Зато в нашем случае Flussonic обеспечит хорошее качество сигнала благодаря использованию буферов на сервере-источнике и сервере-приемнике для выравнивания скачков.

- Укажите в опции `push` адрес мультикаста. На него ТВ-канал будет вещаться абонентам.
- И в завершение настройте сервер-приемник для трансляции мультикаста.

Отправка зашифрованного видео

Если вам нужно отправить через Интернет зашифрованное видео, мы рекомендуем использовать следующую схему:

- Используйте протокол **M4FS** для передачи видео между двумя серверами Flussonic. M4FS работает так же, как и M4F, с одной разницей: он работает поверх HTTPS (использует шифрование SSL/TLS). Просто следуйте описанной выше процедуре, но в конфигурации потока используйте `m4fs://` сместо `m4f://`.
- Используйте защиту контента с помощью DRM на каждом сервере-приемнике (edge).

Читайте также

- Прием мультикаста
- Рассылка мультикаста

Chapter 5

OTT TV

5.1 Prepare content

5.1.1 Как захватить MPEG-TS поток

MPEG-TS можно захватить следующими способами:

- UDP multicast
- плата захвата DVB
- SRT
- или HTTP MPEG-TS

Последний вариант обычно используется в IPTV OTT, его здесь и опишем.

Вы должны ввести подходящий URL в качестве источника потока когда будете добавлять поток во *Flussonic*. Для добавления потока откройте веб-интерфейс *Flussonic* и нажмите **Add** рядом со **Streams**, а затем введите название потока и URL-адрес.

Если у вас источник `http://upstream/path`, поменяйте его схему с `http` на `tshttp`: `tshttp://upstrea`

5.1.2 HLS источники по запросу (on-demand)

Содержание

- Для чего нужны потоки по запросу (on-demand)?
- Проблемы, связанные с ondemand
- Проблемы, связанные с протоколом HLS
- Как это исправить?

Для чего нужны потоки по запросу (on-demand)?

Если поток настроен на режим **по запросу**, он будет включаться только при получении запроса от пользователя. В случае, если у пользователя пропала необходимость в просмотре, или же источник, из которого транслируется поток, по каким-то причинам остановил свою работу — поток автоматически отключится через определенное время (таймаут). Потоки по запросу нужны, в основном, для экономии ресурсов (в первую очередь трафика). Они хорошо подходят в тех случаях, когда у вас есть большое количество потоков, но вам не нужна их одновременная и постоянная работа.

Подробнее о настройке потоков по запросу здесь.

Проблемы, связанные с on-demand

Особенность работы сегментированных протоколов в режиме по запросу (On-demand) состоит в том, что при их использовании требуется накопление буфера плеера перед началом отдачи видео (первичная загрузка хотя бы 3 сегментов видео), что занимает некоторое время. Несмотря на то, что такие параметры, как используемый протокол передачи медиа, размер GOP (группы кадров), длина сегмента, могут повлиять на время накопления буфера, этот процесс и связанная с ним задержка неизбежны.

Проблемы, связанные с протоколом HLS

- HLS является сегментированным протоколом, соответственно, требуется некоторое время на накопление буфера.
- В Flussonic Media Server реализована проверка, определяющая, жив ли источник. Она заключается в том, что Flussonic (рестример), подключившись к HLS источнику, запоминает последний увиденный сегмент и ждет появление следующего. До тех пор, пока он не появится, Flussonic не начнет воспроизведение. Это приводит к дополнительной задержке воспроизведения.

В случае, если мы принимаем неизвестный (не-Flussonic) HLS источник и ретранслируем его, без этой проверки **нет никакого другого способа узнать**, онлайн источник или нет. Отсутствие данной проверки может привести к зацикленному воспроизведению последних считанных сегментов в случае, если источник в какой-то момент уйдет в оффлайн. Flussonic не может доверять стороннему источнику (даже если он предоставляет нужные данные), потому что невозможно определить, чем является этот источник.

Совокупность всех этих факторов приведет к тому, что вы столкнетесь с проблемой долгого старта начала воспроизведения потока.

Как это исправить?

У нас есть решение для обеих этих проблем — наш собственный протокол M4F. Его использование **убирает необходимость проверки источника** (этот протокол гарантирует, что рестример не загрузит уже имеющиеся сегменты второй раз, а источник сбросит плейлист, если на нем пропали кадры), также он поддерживает **prepush** (быстрое наполнение буфера плеера) и предоставляет информацию об архиве источника (или нескольких источников одновременно), дает возможность проксировать архив. Как итог — нивелируются обе причины этой задержки. M4F является внутренним протоколом Flussonic, следовательно, его использование возможно только при условии, если Flussonic Media Server — и источник, и рестример. Подробнее о протоколе ‘M4F‘.

Если ваша цель — именно потоки по запросу (On-demand) и их вывод по протоколу HLS — мы настоятельно рекомендуем вам обратиться к поставщику с просьбой поставить себе Flussonic Media Server, и использовать протокол M4F для передачи данных между ними. В результате вы получите хорошо синхронизированную конфигурацию и решите проблемы, связанные с долгим запуском потоков.

5.1.3 Как принять телетекст из SDI в HLS/DASH

Телетекст

Телетекст — это дополнительные данные (обычно текст, но могут быть и небольшие изображения), которые передаются в телесигнале со спутника. Спутниковое видео попадает на *Flussonic Media Server* через SDI-карты или сразу в формате MPEG-TS. При приеме потоков с SDI-карты *Flussonic* определяет наличие телетекста в SDI-источнике и автоматически упаковывает его в поток MPEG-TS без изменений, поэтому дальнейшая обработка телетекста (преобразование в WebVTT или TTML) осуществляется одинаково.

См. также статью об упаковке телетекста для передачи в MPTS/SPTS.

Получение телетекста из SDI *Flussonic* позволяет читать телетекст различного формата из потоков, захватываемых с SDI-платы. Поддерживаемые форматы телетекста для SDI-источников:

| Плата | VBI (SD SDI) | OP-47 (HD SDI) | | ———– | ——— | ——— | | DekTec | □ | □ | | Decklink | □ | □ | | Stream Labs | □ | □ | | Magewell | □ | □ | | AJA | □ | □ |

***VBI** — *это данные, расположенные в невидимой области кадра, которая передается во время обратного хода луча по вертикали (т.е. время возвращения в верхнюю часть экрана луча кинескопа в телевизорах с электронно-лучевыми трубками). Практически то же, что и* VANC*, но VANC более современный формат, используемый для HD SDI и позволяющий передать больше данных в одной линии.

OP-47 — это спецификация телетекста для HD SDI, позволяющая повысить стабильность и бесперебойность передачи телетекста с потоками высокого разрешения. В этом формате в одной линии можно передать еще больше данных, чем в VANC.

Передача телетекста в HLS и DASH *Flussonic* позволяет передавать телетекст из MPEG-TS:

- В HLS — телетекст в формате WebVTT
- в DASH — телетекст в формате WebVTT и TTML.

Для настройки преобразования *не требуется дополнительных опций.* Если во входном потоке есть телетекст, то *Flussonic* преобразует его автоматически.

Пример конфигурации (никаких специальных опций):

```
stream example_stream1 {
  input tshttp://EXAMPLE-IP/STREAM_NAME/mpegts;
}
```

Для проверки наличия `dvb_teletext` во входном потоке можно воспользоваться следующей командой:

```
ffprobe http://EXAMPLE-IP/STREAM_NAME/mpegts

Stream #0:0[0x447]: Video: h264 (Main) ([27][0][0][0] / 0x001B), yuv420p(tv
    , bt470bg, top first), 704x576 [SAR 16:11 DAR 16:9], 25 fps, 25 tbr, 90k
     tbn, 50 tbc
Stream #0:1[0xc12](eng): Audio: mp2 ([4][0][0][0] / 0x0004), 48000 Hz,
    stereo, fltp, 192 kb/s
Stream #0:2[0x17e2](swe,nor,dan,fin): Subtitle: dvb_teletext ([6][0][0][0]
    / 0x0006)
```

После преобразования в выходном HLS потоке выводится телетекст в формате WebVTT:

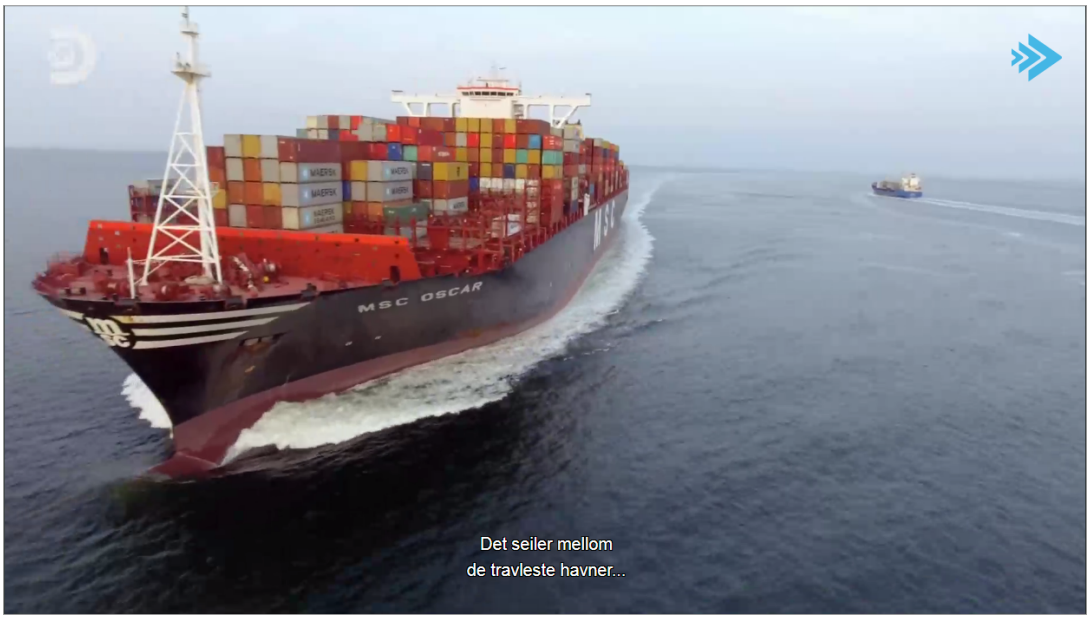


Figure 5.1: Flussonic teletext

Передача скрытых субтитров в HLS и DASH

Наличие скрытых субтитров в HLS и DASH Чтобы HLS или DASH плееры показывали скрытые субтитры, необходимо сигнализировать об этом в плейлист-файлах или так называемых файлах-манифестах. В этих файлах содержится информация о названиях потоков, их URL, а также информация о разрешении, скрытых субтитрах и т.п. *Flussonic* может добавить необходимую сигнализацию в HLS и DASH манифесты.

Для этого добавьте следующий параметр рядом с URL потока в конфигурационном файле (`/etc/flussonic/flussonic.conf`):

```
cc.{608|708}.{INSTREAM-ID}.{lang|name}=VALUE
```

Параметры:

- 608|708 — стандарт скрытых субтитров. Может принимать всего 2 значения: 608 или 708 для стандартов CEA-608 и CEA-708 соответственно.
- INSTREAM-ID — номер канала, в котором содержатся скрытые субтитры. Это целое число от 1 до 4 для CEA-608 и от 1 до 63 для CEA-708.
- lang — параметр языка скрытых субтитров (по стандарту *ISO 639.2/B*)
- name — параметр, указывающий на то, под каким именем будет отображаться название языка в плеере.
- VALUE — конкретное значение языка. В зависимости от того, указан параметр lang или name, форма записи конкретного значения может отличаться.

В качестве примера давайте разберём как будет выглядеть запись языка скрытых субтитров для английского:

- для lang: cc.708.1.lang=eng (по стандарту ***ISO 639.2/B***),
- для name: cc.708.1.name=English (название дорожки, которое будет отображаться в плеере).

Пример использования:

```
stream example_stream2 {
  input tshttp://EXAMPLE-IP/STREAM_NAME/mpegts cc.708.12.lang=fr cc.608.1.
    lang=eng;
}
```

В примере выше содержатся 2 дорожки со скрытыми субтитрами:

- 12-ый канал содержит скрытые субтитры стандарта CEA-708 на французском.
- 1-ый канал содержит скрытые субтитры стандарта CEA-608 на английском.

Вы можете проверить наличие скрытых субтитров в DASH и HLS манифестах.

- для DASH это осуществляется с помощью тега Accessibility в файле с плейлистом. Для этого Вам необходимо воспользоваться командой curl, чтобы сначала загрузить сам плейлист:

```
curl http://FLUSSONIC-IP/example_stream2/index.mpd
```

Возвращаясь к примеру выше, информация о скрытых субтитрах потока example_stream2 будет выглядеть следующим образом:

```
<Accessibility schemeIdUri="urn:scte:dash:cc:cea-608:2015" value="CC1=eng;
   CC1=eng"/>
<Accessibility schemeIdUri="urn:scte:dash:cc:cea-708:2015" value="12=lang:
   fr;12=lang:fr"/>
```

- тот же порядок действий и для HLS. Отличие лишь в формате представления плейлиста:

```
curl http://FLUSSONIC-IP/example_stream2/index.m3u8
```

```
#EXT-X-MEDIA:TYPE=CLOSED-CAPTIONS,GROUP-ID="v1cc",LANGUAGE="fr",NAME="fr12
   ",INSTREAM-ID="SERVICE12"
#EXT-X-MEDIA:TYPE=CLOSED-CAPTIONS,GROUP-ID="v1cc",LANGUAGE="eng",NAME="eng1
   ",INSTREAM-ID="CC1",AUTOSELECT=YES,DEFAULT=YES
```

Извлечение и конвертация скрытых субтитров *Flussonic* позволяет извлекать CEA-608 скрытые субтитры из входного потока MPEG-TS и осуществлять их последующую конвертацию:

- в формате WebVTT — для HLS
- в формате WebVTT и TTML — для DASH.

После операции транскодирования скрытые субтитры сохраняются в выходном потоке.

Чтобы *Flussonic* осуществил извлечение и конвертацию скрытых субтитров, добавьте параметр `cc.extract` к URL-адресу потока.

Для MPEG-TS:

```
stream example_stream3 {
  input tshttp://EXAMPLE-IP/STREAM_NAME/mpegts cc.extract;
}
```

Опция `cc.extract` доступна на источниках MPEG-TS.

Расположение субтитров Для обозначения расположения субтитров на видео настройте параметр `substyle valign=top|middle|bottom align=left|center|right`, например:

```
stream example_stream5 {
  input tshttp://EXAMPLE-IP/STREAM_NAME/mpegts cc.extract;
  substyle valign=top align=left;
}
```

Расположение субтитров можно также задать в UI на вкладке **Output** в настройках потока:

После преобразования в выходном потоке HLS присутствуют скрытые субтитры в формате WebVTT:

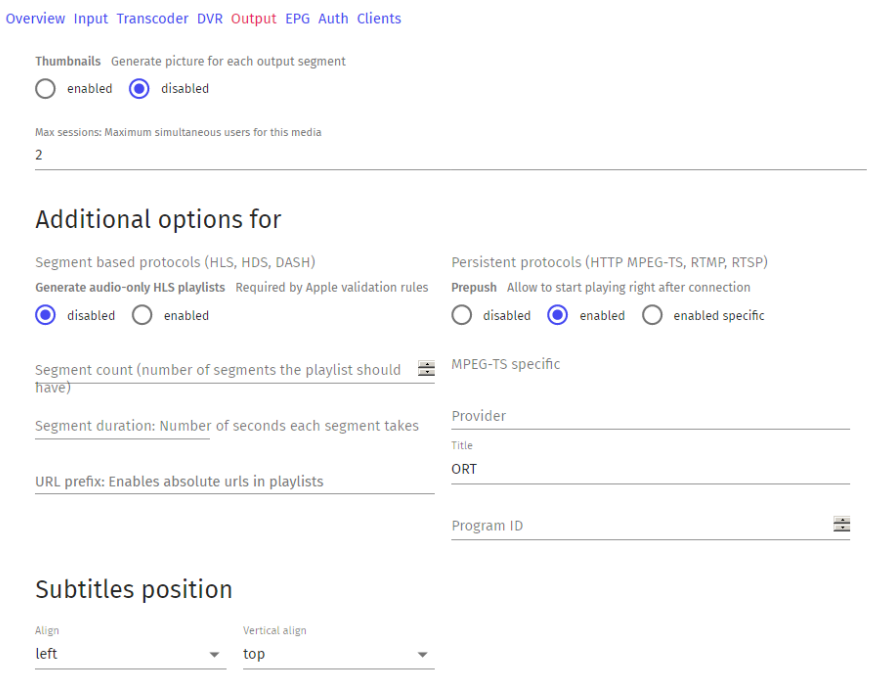


Figure 5.2: Flussonic closed captions

Figure 5.3: Flussonic closed captions

Выбор субтитров для проигрывания по DASH Поскольку в манифест DASH включены два формата субтитров, вы можете выбрать один из них при проигрывании выходного потока:

```
https://FLUSSONIC-IP/STREAM_NAME/index.mpd?text=wvtt
```

или

`https://FLUSSONIC-IP/STREAM_NAME/index.mpd?text=ttml` (TTML используется по умолчанию)

Передача субтитров в MSS

Flussonic передает любой тип субтитров (субтитры, закрытые титры или телетекст) в формате TTML в выходные потоки MSS. Специальной настройки не требуется, единственное требование — входящие потоки должны иметь TTML субтитры.

Для обозначения расположения субтитров на видео используйте опцию `substyle valign=top|middle|b` например:

```
stream example_stream6 {
  input tshttp://EXAMPLE-IP/STREAM_NAME/mpegts cc.extract;
  substyle valign=top align=left;
}
```

О TTML-субтитрах

TTML (Timed Text Markup Language) — это стандарт для субтитров и скрытых субтитров (closed captions), который широко поддерживается плеерами, стриминговыми платформами и другим программным обеспечением, а также используется в телевизионной отрасли. Стандарт TTML имеет богатые возможности для позиционирования, выравнивания, передачи субтитров на нескольких языках и др. Субтитры TTML передаются в виде текстового файла на основе XML с расширением (.ttml) или (.xml).

Flussonic передает TTML-субтитры в MSS и DASH.

5.1.4 Конвертация DVB-субтитров в HLS/DASH WebVTT

Flussonic Media Server может распознавать DVB-субтитры и конвертировать их в текстовый формат WebVTT. Это необходимо для того, чтобы вы могли показать субтитры на мобильных устройствах и плеерах без поддержки DVB.

> Функция OCR-распознавания субтитров доступна в рамках отдельной лицензии, которая приобретается дополнительно.

Распознавание осуществляется с помощью программы . Tesseract распознает текст и передает его Flussonic, который затем создает WebVTT субтитры, доступные по HLS.

Содержание:

- О DVB субтитрах
- О WebVTT субтитрах
- Установка и настройка

См. также:

- Телетекст и Closed Captions

О DVB-субтитрах

Стандарт DVB (Digital Video Broadcast) определяет формат субтитров на основе растрового изображения. В MPEG-TS потоке со спутника DVB субтитры могут идти как текстом, так и изображением. Чаще всего используется передача изображений, так как этот вариант надежнее для просмотра на различных устройствах, которые могут не иметь нужных шрифтов для отрисовки текста.

Например:

```
ffprobe stream_sample.ts
Stream #0:0[0x1a4]: Video: h264 (High) ([27][0][0][0] / 0x001B), yuv420p(tv
    , bt709), 1920x1080 [SAR 1:1 DAR 16:9], 25 fps, 25 tbr, 90k tbn, 50 tbc
    Stream #0:1[0x1ae](fra): Audio: eac3 ([6][0][0][0] / 0x0006), 48000 Hz,
     stereo, fltp, 128 kb/s
    Stream #0:2[0x1af](qad): Audio: eac3 ([6][0][0][0] / 0x0006), 48000 Hz,
     stereo, fltp, 128 kb/s
    Stream #0:3[0x1b8](fra): Subtitle: dvb_subtitle ([6][0][0][0] / 0x0006)
     (hearing impaired)
    Stream #0:4[0x1b0](qaa): Audio: eac3 ([6][0][0][0] / 0x0006), 48000 Hz,
     stereo, fltp, 128 kb/s
    Stream #0:5[0x1b9](fra): Subtitle: dvb_subtitle ([6][0][0][0] / 0x0006)
```

Здесь `dvb_subtitle` — это субтитры, которые приходят изображениями.

DVB субтитры показываются приставками, некоторыми телевизорами, плеером VLC, но не смартфонами iPhone/Android. Flussonic Media Server распознает эти изображения и переведет в текст для показа на таких устройствах.

О WebVTT-субтитрах

WebVTT (Web Video Text Tracks) — это распространённый формат субтитров, который хорошо поддерживается браузерами и предоставляет ряд других возможностей. Конвертируя DVB субтитры в WebVTT, можно уменьшить нагрузку на канал передачи потока.

Файл формата WebVTT является обычным текстовым файлом с расширением `.vtt`, в котором в виде построчного списка прописаны метки с временем старта и временем окончания и текстовые сообщения для вывода к этим меткам.

К видео-контейнеру можно подключить несколько файлов WebVTT с текстом на разных языках. Для каждого языка нужно создавать отдельный файл.

Эти файлы могут использоваться для передачи дополнительных данных в JS плееры. Например, URL-превью картинок для кадров потока. Стандарт WebVTT также поддерживает CSS-стилизацию и опции по размещению в области просмотра видео.

Установка и настройка

Tesseract — качественный консольный OCR движок с открытым исходным кодом. Программа работает с UTF-8, а поддержка языков (включая русский) осуществляется с помощью дополнительных модулей.

Чтобы настроить распознавание и преобразование DVB субтитров:

- Установите Tesseract.

```
apt install flussonic-tesseract
```

2. Для получения дорожки с текстовыми субтитрами добавьте в файле `/etc/flussonic/flussonic.conf` в настройки потока следующую строку:

```
subtitles=ocr_replace;
```

Опция `subtitles=ocr_replace` включает механизм преобразования DVB субтитров в Web-VTT. При этом дорожка с новыми субтитрами в текстовом формате заменяет в выходном потоке дорожку с DVB-субтитрами.

Пример: `stream tvchannel { input tshttp://SOURCE:80/STREAM subtitles=oc`

Если в выходном потоке необходимо получить как прежнюю дорожку с DVB-субтитрами, так и новую дорожку с текстовыми субтитрами, то используется опция `subtitles=ocr_add`:

```
stream tvchannel { input tshttp://SOURCE:80/STREAM subtitles=ocr_add;
```

Замечание. До выхода версии Flussonic 19.10 вместо `subtitles=ocr_replace` использовалась другая опция — `dvbsubs_ocr=true`.

- Для применения настроек выполните команду в консоли:

```
service flussonic reload
```

Если Tesseract запустился для потока, то в логах появятся примерно такие записи:

```
09:44:17.986 <0.966.0> [sow] tesseract_worker:58 start ocr for slv
09:44:18.275 <0.966.0> [sow] tesseract_worker:58 start ocr for srp
09:44:18.759 <0.966.0> [sow] tesseract_worker:58 start ocr for swe
09:44:19.045 <0.966.0> [sow] tesseract_worker:58 start ocr for dan
09:44:19.328 <0.966.0> [sow] tesseract_worker:58 start ocr for nor
```

Пример HLS плейлиста index.m3u8 с субтитрами:

```
#EXTM3U
#EXT-X-MEDIA:TYPE=SUBTITLES,GROUP-ID="subs",NAME="English",DEFAULT=YES,
    AUTOSELECT=YES,FORCED=NO,LANGUAGE="eng",URI="http://flussonic-ip/index.
    m3u8"
#EXT-X-MEDIA:TYPE=SUBTITLES,GROUP-ID="subs",NAME="French",DEFAULT=NO,
    AUTOSELECT=YES,FORCED=NO,LANGUAGE="fra",URI="http://flussonic-ip/index.
    m3u8"
#EXT-X-MEDIA:TYPE=SUBTITLES,GROUP-ID="subs",NAME="German",DEFAULT=NO,
    AUTOSELECT=YES,FORCED=NO,LANGUAGE="deu",URI="http://flussonic-ip/index.
    m3u8"
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=450560,RESOLUTION=480x352,
    SUBTITLES="subs"
http://flussonic-ip/index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=855040,RESOLUTION=480x352,
    SUBTITLES="subs"
http://flussonic-ip/index.m3u8
```

После преобразования в выходном HLS потоке присутствуют субтитры в формате WebVTT:

Передача телетекста и Closed Captions

Flussonic умеет передавать телетекст и скрытые субтитры (closed captions), полученные по MPEG-TS. Для выходного HLS они будут преобразованы в WebVTT, а для DASH — в TTML. См. Телетекст и Closed Captions

Figure 5.4: Flussonic subtitles

5.1.5 Как транскодировать канал в несколько качеств

В этой статье мы рассмотрим типовую конфигурацию транскодера под OTT-сервис. В отдельной статье мы подробно описали как работают такие сервисы, но я кратко напомню ключевые особенности из-за которых и требуется транскодирование: - Передача через Интернет: Нет гарантированной полосы до клиентов - Большое разнообразие устройств: телевизоры разных платформ и поколений, смартфоны, браузеры. Не существует универсального протокола, DRM и кодека, разрешения который бы поддерживался всеми.

Транскодирование в мультибитрейт позволит охватить всех: от 4K телевизоров, подключенные к широкополосному интернету, до пятилетнего смартфона, с которого смотрят футбольный матч через 3G в деревне.

Мультибитрейт - это несколько дорожек различного качества. Качество - субъективный параметр, который складывается из совокупности параметров: кодек, разрешение, битрейт.

Как это выглядит для зрителей:

- В штатном режиме зритель получает самое высокое качество, которое позволяет его интернет и устройство, он вообще не обращает внимание на качество, ему итак хорошо.
- Если интернет-соединение нестабильно, то зритель замечает, что картинка то четкая, то ”мыльная”. Если ситуация не стабилизируется, то он вручную выбирает конкретное качество, чтобы плеер не ”скакал” между разными дорожками.

С теорией закончили, приступим к практике.

Подготовка

Вам нужен сервер, у которого достаточно ресурсов. Мы рекомендуем наш Flussonic Coder или сервер с видеокартой Nvidia. Как подобрать сервер под ваш сервис мы написали в отдельной статье.

Далее по тексту ожидается, что Flussonic Media Server уже установлен, а потоки захвачены. Чаще всего, это будет UDP Multicast или все равно MPEG-TS, но из другого источника.

Настройка

- Откройте вкладку **Transcoder** в настройках потока.
- Нажмите **Enable transcoder**, чтобы показать форму настройки.
- Откройте выпадающий список рядом с кнопкой **Add Video Track** и выберете несколько дорожек. Я выберу все пять: у меня источник 1080p и я хочу чтобы поток можно было посмотреть даже через медленное соединение.
- Нажмите **Save**.

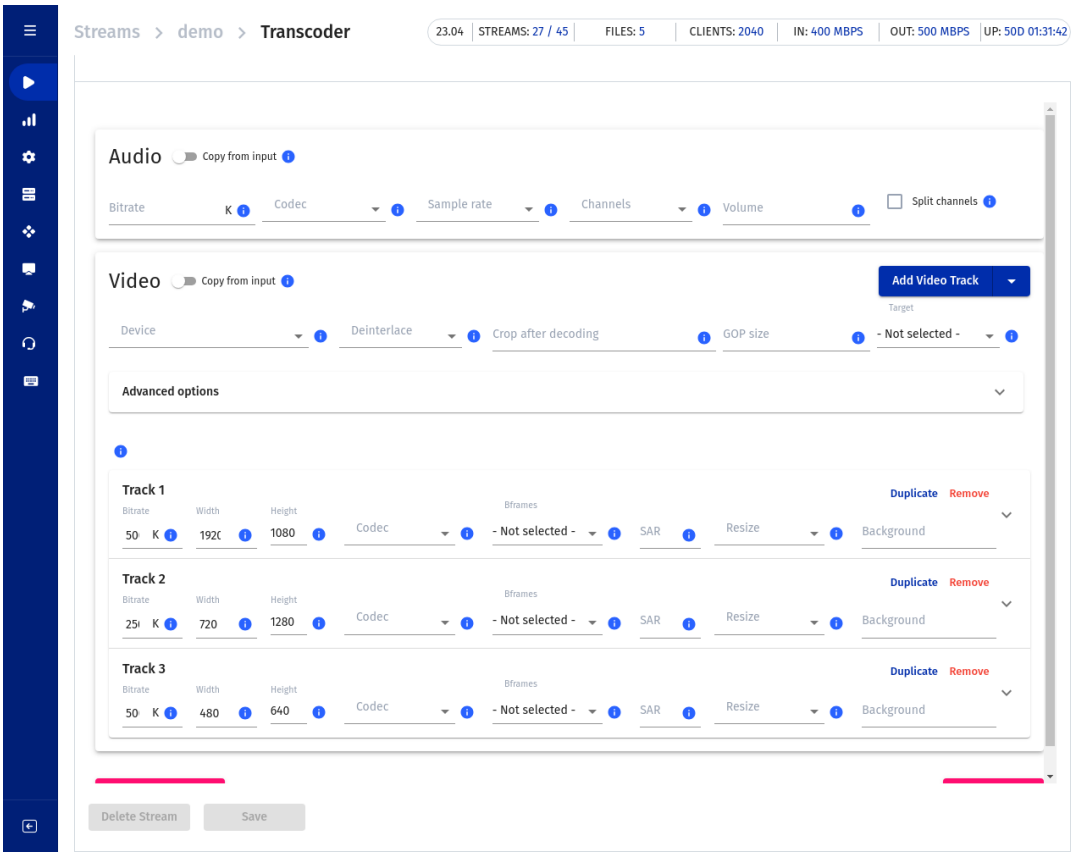


Figure 5.5: Flussonic transcoder

Продвинутая настройка

Если вам не подходят наши настройки по умолчанию, вы можете на свое усмотрение переопределить любой параметр. Подробное описание параметров.

Как проверить, что мультибитрейт работает

Проверить доступные для вывода дорожки можно на вкладке **Overview** в настройках потока. Если вы правильно настроили транскодер, то увидите в разделе **Output media info** будет дорожек больше, чем на входе.

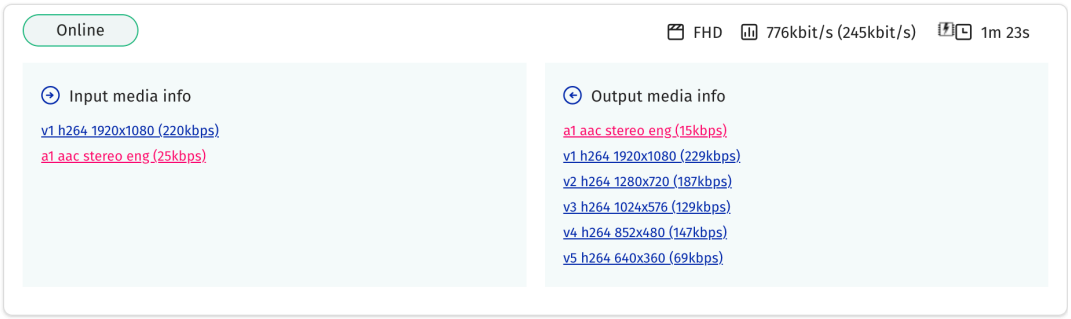


Figure 5.6: Output media info MBR

А в плеере появится меню выбора качества.

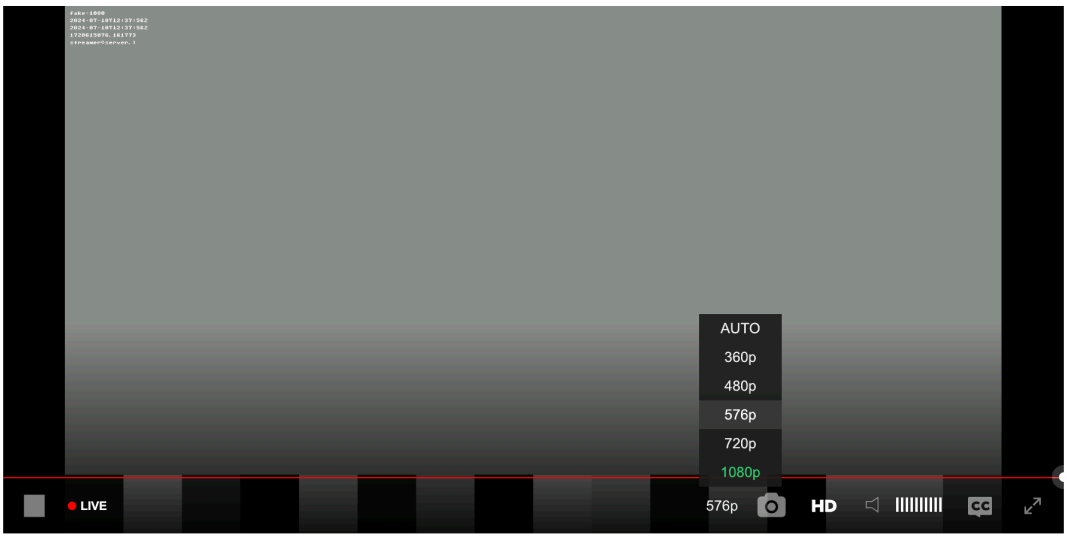


Figure 5.7: Live player MBR

Резюме

Мультибитрейт необходим OTT-сервису, а в Media Server через UI можно настроить типовую конфигурацию за минуту, не разбираясь в настройках транскодирования, просто выбрав заранее подготовленные настройки. Конфигурацию можно адаптировать под ваши требования по качеству, под требования регулятора или доступные ресурсы.

5.1.6 Добавить потоки от стороннего транскодера

Иногда бывает так, что на старте проекта покупается какой-то транскодер, который выдает несколько качеств в UDP потоки и возникает вопрос: а как добавить эти источники в флюссоник так, чтобы на выходе получился хороший DASH/HLS с надежным переключением битрейта в плеере?

Простой ответ: никак, заменить транскодер.

Более подробный будет дан в этой статье вместе с инструментом для отладки.

В чём проблемы с MBR UDP транскодерами?

Самые частые проблемы, по которым нельзя собрать хороший DASH стрим из под таких транскодеров следующие:

- У разных качеств не одинаковые длины GOP-ов. Такая же проблема бывает и при захвате с RTSP камер двух качеств. Это фатально для большинства OTT IPTV плееров.
- Не синхронизированы ключевые кадры (кейфреймы). У разных качеств независимые транскодеры, поэтому один ставит кейфрейм на одном кадре, другой на другом. Если синхронизировать UDP источники по близко расположенным кейфреймам, то на выходе будет рассинхрон как видео друг между другом (будут пропадать или дублироваться кадры на переключении качеств), так и видео с аудио. Ведь аудио будет взято с первого источника, а видео со второго.
- Почти всегда не синхронизированы таймстемпы. С UDP MPEG-TS нормальная практика стартовать время с какого-то произвольного таймстемпа, поэтому чаще всего через пару дней работы с мультибитрейтного UDP транскодера льются потоки, на которых таймстемпы отличаются друг от друга на много часов: один рестартнулся и всё разъехалось. Частично выправляется автокоррекцией, но там см. п.2

Большинство транскодеров на рынке работают с мультибитрейтом так: - Делают несколько независимых потоков из одного входящего - Отправляют в сеть по UDP, протоколу, не имеющего механизмов гарантированной доставки или подтверждения получения

Даже в локальной сети, где между транскодером и упаковщиком минимальное расстоянии, даже при подключении в один коммутатор, UDP периодически теряет часть данных.

Когда такое случается, софт на упаковщике должен принять решение: - Убрать из манифеста один из профилей, вообще не предоставляя клиентам эту дорожку. - Заполнить пустоту ”нулями”, пытаясь сохранить текущую структуру, в надежде, что данные на входе вновь появятся через мгновенние. Это приводит к артефактам и зависаниям.

Напоминаем, что все сегменты должны иметь одинаковые параметры: кодек, разрешение, идентичную GOP структуру, чтобы плеер не подвис при проигрывании. А это невозможно гарантировать, когда часть кадров теряется на входе упаковщика.

Потеря одного профиля в лайве может быть незамечена, особенно, когда нет клиентов, кто смотрел в тот момент этот профиль. Однако, потеря одного профиля испортит записи. Когда Media Server ведет архив для nPVR или Catch-up TV, он инициализирует запись для конкретного набора аудио и видео дорожек. Если один из профилей временно исчезнет, это нарушит целостность всего потока: будут пробелы в разных профилях, продолжительность записи высокого качества будет отличаться от записи низкого качества. По статистике, чаще повреждаются дорожки высокого качества, так как у них больше битрейт.

Поэтому мы не рекомендуем использовать MBR UDP в качестве источников. Лучше использовать протоколы, которые поддерживают мультибитрейт в одном контейнере и гарантируют доставку: M4F, SRT. HLS будет лучше, чем UDP, но у него есть свои особенности.

Как проверить?

В поставке идет утилита `mbr_udp_analyze.erl`, её надо запускать в командной строке и передать ей список адресов, которые надо проверить:

```
/opt/flussonic/contrib/mbr_udp_analyze.erl udp://239.150.12.1:1234 udp
   ://239.150.22.2:1234 udp://239.150.12.3:1234
Synced 3 with shift 0.00
Synced 2 with shift 280.00
SYNC
Started stream analysis
Gop at 70228737.78 ERRORS:
#{error => not_starting_from_keyframe,input => 2,dts_shift => 40.0}

Gop at 70229737.78 ERRORS:
#{error => not_starting_from_keyframe,input => 2,dts_shift => 0.0}

Gop at 70230737.78 OK
Gop at 70231737.78 ERRORS:
#{error => not_starting_from_keyframe,input => 2,dts_shift => 0.0}
```

Из лога в примере видно, что некоторые сегменты получится собрать такими, чтобы на них было возможно переключение, но почти весь поток структурно не годится для работы в OTT среде.

Вот как будет выглядеть хороший анализ на искуственно собранном потоке:

```
/opt/flussonic/contrib/mbr_udp_analyze.erl udp://239.150.12.1:1234 udp
   ://239.150.13.1:1234 udp://239.150.14.1:1234
Synced 2 with shift 0.00
Synced 3 with shift 0.00
SYNC
Started stream analysis
Gop at 69999737.78 ERRORS:
#{error => not_starting_from_keyframe,input => 3,dts_shift => 40.0}

Gop at 70000737.78 OK
Gop at 70001737.78 OK
```

```
Gop at 70002737.78 OK
Gop at 70003737.78 OK
Gop at 70004737.78 OK
Gop at 70005737.78 OK
```

Что же делать?

Вряд ли вы купили какой-то волшебный аппаратный транскодер. На сегодняшний день такого практически не существует и под аппаратными решениями подразумевается максимум наличие какого-то ускорителя от Nvidia, Intel или пары других решений.

Скорее всего вы можете на этот компьютер поставить обычный Linux, на него поставить Flussonic и получить гарантированно работающий MBR транскодер

5.1.7 Аппаратное транскодирование на NVIDIA NVENC

Транскодирование видео на NVIDIA NVENC

Flussonic Media Server умеет кодировать видео, используя GPU на видеокартах NVIDIA. Список поддерживаемых видеокарт можно найти на сайте Nvidia.

Также в системе должен быть установлен драйвер Nvidia версии выше 400.

При использовании NVIDIA Nvenc наш транскодер может обрабатывать 10-битные потоки.

Для транскодирования видео в кодеке AV1 требуется видеокарта NVIDIA Ada Lovelace и выше.

> Некоторые видеокарты Nvidia NVENC имеют ограничение на количество одновременных сессий (транкодируемых потоков). Если с помощью видеокарты Nvidia NVENC транскодируется слишком много потоков, во Flussonic UI будет показано соответствующее предупреждение. Чтобы посмотреть максимально допустимое количество одновременных сессий для вашей карты, см. сайт NVIDIA. Кроме того, если карта «не соответствует требованиям» (unqualified), то ограничение количества сессий может быть «на сервер», а не только «на карту»; для этого ознакомьтесь с политикой NVIDIA.

Установка драйвера Драйвер устанавливается из соответствующего пакета.

Ubuntu 20.04:

```
apt-get install nvidia-driver-515-server --no-install-recommends
```

В `sources.list` должен быть включен компонент `non-free`.

Для других системах можно установить драйвер с официального сайта Nvidia. Инструкция по установке драйвера для Ubuntu

Для работы с большим количеством транскодируемых потоков может потребоваться увеличение лимита на открытые файлы. Сделать это можно командой:

```
ulimit -n 4096
```

Добавьте в файл следующие строки в файл `/etc/security/limits.conf`:

```
* soft nofile 4096
* hard nofile 4096
```

Включение транскодера Настройки транскодера можно задать:

- В конфигурационном файле Flussonic **/etc/flussonic/flussonic.conf** в опциях потока, используя директиву `transcoder` с разными опциями.

- В интерфейсе администратора в **Media** > выбрать поток > **Transcoder**.

Для включения аппаратного кодирования с использованием Nvenc необходимо прописать опцию `hw=nvenc`:

```
transcoder vb=2048k hw=nvenc ab=128k
```

Выбор кодека По умолчанию используется H.264. При кодировании на Nvenc вы также можете использовать:

- H.265 (HEVC):

```
transcoder vb=2048k hw=nvenc vcodec=hevc ab=128k
```

- AV1:

```
transcoder vb=2048k hw=nvenc vcodec=av1 ab=128k
```

Поддержка потоков с 10-битной глубиной цвета Транскодер Flussonic может работать с 10-битными потоками при использовании NVIDIA Nvenc. Эта функция поддерживается для всех вариантов кодеков на входе и выходе.

Чтобы транскодировать видео из не-10-битного в 10-битное, используйте значение опции `pix_fmt`, оканчивающееся на `p10`. Список всех доступных значений см. в схеме API.

Например, для транскодирования 8-bit HEVC/H.264/AV1 в 10-bit HEVC задайте опции транскодера так:

```
transcoder vb=3000k vcodec=hevc pix_fmt=yuv420p10 ab=128k
```

Необходимо использовать драйвера NVIDIA версии выше 400, а версию Ubuntu — 18.04 или выше.

Выбор видеокарты

Вручную Если в системе установлено несколько видеокарт, то можно выбрать какую из них использовать для транскодирования. Для этого используется опция `deviceid`:

```
transcoder vb=2048k hw=nvenc deviceid=1 ab=128k
```

Номер видеокарты можно узнать при помощи команды `nvidia-smi`. По умолчанию используется первая видеокарта `deviceid=0`.

Автоматически Если у вас много потоков, то Flussonic поможет автоматически распределить их между видеокартами для транскодирования. Автоматическое распределение потоков происходит на основании анализа загрузки карты и потребления видеопамяти.

Включить распределение потоков между GPU можно в конфигурационном файле, указав в `transcoder` опцию `deviceid=auto` для каждого потока:

```
transcoder vb=2048k hw=nvenc deviceid=auto ab=128k
```

Обрезка видео Опция `crop` позволяет обрезать видео. Указывается отдельно для каждого видеопотока.

Использование: `crop=x:y:width:height`, где:

- `x:y` — координаты левого верхнего угла выходного видео в пределах размеров входного видео,
- `width` — ширина выходного видео
- `height` — высота выходного видео.

Пример:

```
transcoder vb=2048k hw=nvenc crop=0:0:100:100 ab=128
```

Декодирование на CPU По умолчанию, декодирование также происходит на GPU. Чтобы использовать для декодирования центральный процессор, вместо `hw=nvenc` укажите `hw=nvenc2`:

```
transcoder vb=2048k hw=nvenc2 ab=128k
```

Деинтерлейсинг Деинтерлейсинг (устранение чересстрочности) при использовании `nvenc` происходит по умолчанию. Но можно указать определенный метод при помощи опции `deinterlace`. Например, добавьте `deinterlace=yadif`, чтобы применить метод **CUDA yadif**:

```
stream test {
   input file://vod/test.ts;
   transcoder vb=4000k ab=128k deinterlace=yadif hw=nvenc;
}
```

Все доступные для NVIDIA Nvenc методы можно увидеть в UI в настройках транскодера для потока, в поле **Deinterlace mode**.

В случае `nvenc2` (использование CPU) деинтерлейсинг следует включить с помощью опции `deinterlace=yes`.

Для отключения дорогостоящего деинтерлейса укажите `deinterlace=0`.

Прочие параметры, такие как `size`, `preset`, `bframes`, `level`, используются аналогично CPU транскодеру.

Возможные значения параметра `preset`: `veryfast`, `medium`, `slow`.

Чтение телетекста из VBI при транскодировании SDI-потока на NVIDIA NVENC *Flussonic* может читать телетекст из VBI при транскодировании SDI-источника с помощью NVIDIA NVENC. Примеры конфигурации смотрите в статье: Примеры конфигураций для чтения телетекста из разных источников.

Статистика производительности NVIDIA

Вы можете собирать статистику о работе GPU Nvidia, если включите сохранение статистики в базе данных Pulse. Чтобы начать сохранять данные, добавьте в файл конфигурации Flussonic следующую директиву:

```
nvidia_monitor true;
```

Чтобы прекратить сохранять статистику по Nvidia, обновите конфиг:

```
nvidia_monitor false;
```

Мониторин доступен в сервисе Retroview в личном кабинете.

5.1.8 Intel Quick Sync Video

Информацию о поддерживаемых платформах вы можете посмотреть в официальном GitHub аккаунте Intel: https://github.com/intel/media-driver#supported-platforms

После установки QSV транскодер доступен через опцию **hw=qsv**:

```
stream example {
  input udp://239.1.1.10:5500;
  transcoder vb=3000k hw=qsv ab=64k;
}
```

Подробнее про настройку транскодера

Установка в Ubuntu

Мы подготовили .deb пакет, позволяющий легко устновить поддержку QSV в вашу систему. Совместимость проверена только с **Ubuntu 18.04**.

```
apt install linux-base flussonic-qsv intel-media-va-driver libdrm-intel1
   vainfo i965-va-driver libpciaccess0
reboot
```

Установка в CentOS

Воспользуйтесь официальной инструкцией Intel: https://github.com/Intel-Media-SDK/MediaSDK#system-requirements

5.1.9 Наложение логотипа с помощью транскодера

Flussonic Media Server может накладывать логотип на видео в процессе транскодирования:

Figure 5.8: Транскодер

Необходимо загрузить логотип в формате .png.

Такой логотип будет «вшит» в видеодорожку и будет отображаться на всех устройствах и в архивных записях.

Пример:

```
vb=2048k preset=veryfast logo=/storage/logo.png@10:10 ab=128k
```

Здесь `10:10` — это координаты с отступом 10 от левого верхнего угла экрана. Для размещения в других частях экрана нужно использовать немного более сложную формулу.

Для размещения в центре:

```
vb=2048k logo=/storage/logo.png@(main_w-overlay_w-10)/2:(main_h-overlay_h
    -10)/2 ab=128k
```

Для размещения в левом нижнем углу:

```
vb=2048k logo=/storage/logo.png@10:(main_h-overlay_h-10) ab=128k
```

Для размещения в правом верхнем углу:

```
vb=2048k logo=/storage/logo.png@(main_w-overlay_w-10):10 ab=128k
```

Для размещения в правом нижнем углу:

```
vb=2048k logo=/storage/logo.png@(main_w-overlay_w-10):(main_h-overlay_h-10)
     ab=128k
```

Flussonic Media Server может накладывать логотип только при кодировании на CPU и NVENC.

5.1.10 Динамическое наложение текста

Чтобы добавить на видео динамический неудаляемый текст, который будет доступен на любых устройствах поверх живого видео и в архиве, используйте транскодер Flussonic Media Server. Например, текст может быть таким:

- Название фильма или телепередачи.
- Прогноз погоды.
- Счет футбольного матча.
- Новостная лента, оповещение о чрезвычайных ситуациях.

Альтернативой транскодеру может быть добавление текста на стороне плеера. Этот способ не нагружает сервер, но добавленный таким способом текст будет отображаться по-разному или вовсе не отображаться на телевизорах, мобильных устройствах, приставках и т.п., его легко удалить с видео, и в архиве он будет недоступен.

Как добавить текст

Рассмотрим тестовый пример:

- Добавьте в конфигурацию Media Server тестовый поток с источником `fake://fake`.
- Включите транскодер на вкладке **Transcoder** в профиле созданного потока.
- Отправьте в Media Server API-запрос с опцией 7Ctranscoder%7Cglobal%7Cburntag/stream/operation/strea get/response%7Ctranscoder%7Cglobal%7Cburntranscoder.global.burn:

```
curl -u LOGIN:PASSWORD -X PUT "http://FLUSSONIC-IP/streamer/api/v3/streams/
    demo" \
-H "Content-Type: application/json" \
--data '{"transcoder": {"global":{"burn":{"text":{"position": "tr","text":
    "Your\nText","y": 10,"x": 10,"box": {"color": "green"}}}}}}'
```

Вот так выглядит результат.

Логика смены текста должна быть реализована во внешнем приложении или скрипте.

Подробнее о доступных настройках прожига текста см. в разделе Прожиг текста, времени и субтитров.

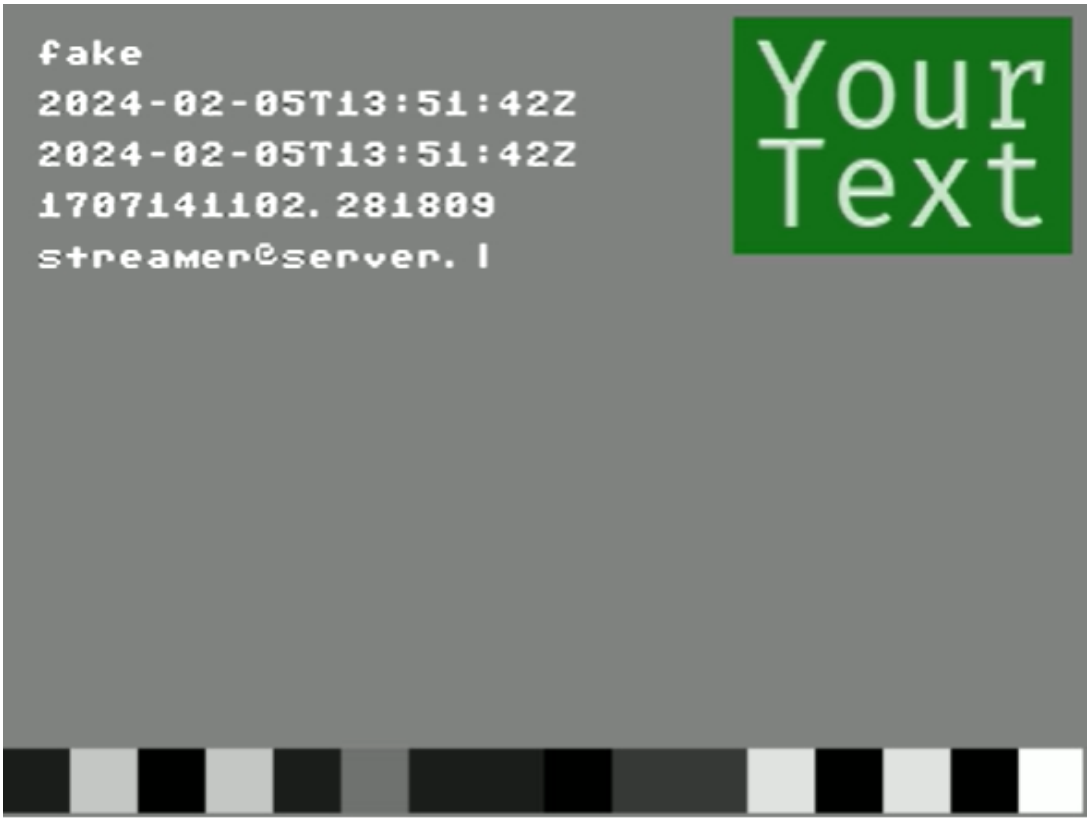


Figure 5.9: Burn example

Связанные задачи

- Наложение логотипа с помощью транскодера — о наложении статического изображения на видео.

5.1.11 Как изменить уровень громкости звука?

Что, если у Вашего источника слишком громкий звук, выше, чем у остальных? Или, наоборот, ниже? В таком случае Вам необходимо правильно настроить уровень громкости звука для таких потоков. Для этого есть два способа: через конфигурационный файл *Flussonic* или через веб-интерфейс *Flussonic UI*. Ниже Вы сможете ознакомиться с обоими способами и выбрать наиболее удобный для себя.

Значение параметра указывается либо в децибелах (dB), либо в целых числах или десятичных дробях (3, 0.5 и т.д.).

- Если указано в целых числах или десятичных дробях, то значение этого параметра указывает на то, **в какое количество раз** Вы хотите изменить уровень громкости звука. Конечное значение высчитывается по следующей формуле:

новое_значение = *avol* текущее_значение*

- Если Вы указываете значение в децибелах (дБ), то конечное значение уровня громкости звука будет высчитываться по другой формуле:

новое_значение = *текущее_значение +/- avol*

в зависимости от того, какое значение будет Вами указано: положительное (+9dB) или отрицательное (-6dB).

Важно: Не забудьте указать плюс (”+”) или минус (”-”) при указании значения!

Изменение уровня уровня громкости звука в конфигурационном файле *Flussonic*

Для изменения громкости звука добавьте параметр `avol` в описание потока в файле конфигурации (`/etc/flussonic/flussonic.conf`).

```
stream example1 {
  input udp://239.0.0.1:1234;
  transcoder vb=copy ab=128k acodec=aac avol=2;
}
```

По умолчанию `avol=1`. В примере выше `avol=2` мы увеличили громкость звука в 2 раза. Если Вы укажете `avol=0.5`, то он уменьшится в 2 раза:

```
stream example2 {
  input udp://239.0.0.1:1234;
  transcoder vb=copy ab=128k acodec=aac avol=0.5;
}
```

В примере ниже Вы можете посмотреть как указывается значение в децибелах (dB). В данном случае мы уменьшаем исходное значение громкости звука на 6 децибел:

Важно: Не забудьте указать плюс (”+”) или минус (”-”) при указании значения!

```
stream example3 {
  input udp://239.0.0.1:1234;
  transcoder vb=copy ab=128k acodec=aac avol=-6dB;
}
```

Изменение уровня громкости звука во Flussonic UI

Для того, чтобы изменить уровень громкости звука через веб-интерфейс *Flussonic UI*:

- Откройте веб-интерфейс *Flussonic UI*.
- Откройте вкладку **Media** -> **Streams** и нажмите на название потока, уровень громкости которого вы хотите изменить.
- Перейдите на вкладку **Transcoder**. В настройках **Audio** укажите значение для изменения параметра уровня громкости звука в строке *Volume*. По умолчанию он равен 1:
- Сохраните изменения, нажав **Save**.

Теперь Вы знаете каким образом можно изменить уровень громкости звука транскодированного потока в *Flussonic Media Server*.

Figure 5.10: Строка Volume

5.1.12 Как зарезервировать захват источника

Кластерный захват потоков необходим для решения следующей проблемы: допустим, у вас есть некоторое количество серверов (до 20), объединённых в группу, а также множество потоков, которые необходимо принять, причём каждый поток может быть принят только на каком-либо одном сервере из группы.

Если один из серверов выходит из строя, потоки необходимо автоматически захватывать на другом сервере.

Механизм кластерного захвата работает следующим образом: в конфигурационном файле описываются все серверы, участвующие в захвате, и прописывается директива `cluster_key`:

```
cluster_key abcd;
```

```
peer flussonic:8080;
peer streamer:8081;
peer s03.myhosting.com;
```

После чего описываются нужные потоки с указанием опции `cluster_ingest`:

```
stream cam01 {
  input file://vod/bunny.mp4;
  cluster_ingest;
}
stream cam02 {
  input file://vod/bunny.mp4;
  cluster_ingest;
}
stream cam03 {
  input fake://fake;
  cluster_ingest capture_at=streamer;
}
```

В качестве параметра опции `cluster_ingest` можно указать явную привязку к одному серверу (`capture_at`). Это нежесткая привязка, потому что если этот сервер выключится, поток всё равно запустится на другом.

Статус серверов в вашем кластере отображается на странице **Cluster** -> **Overview**.

При достаточно большом количестве потоков они будут равномерно распределены между серверами. Настройте балансировщик нагрузки, чтобы задать правила распределения.

Если сервер отключится, потоки автоматически запустятся на других серверах. Если он снова включится, потоки автоматически отключатся на других серверах и вернутся на него. Вы можете посмотреть, на каких серверах сейчас запущены потоки, на странице **Streams**:

Если вы обратитесь за потоком к любому серверу из кластера, на котором этого потока нет, то будете перенаправлены на другой сервер с помощью специального кода. Т.е. фактически вы можете обратиться к любому серверу из кластера и будете перенаправлены на текущий активный сервер.

Сейчас этой возможностью можно пользоваться для захвата с камер или в ситуации, когда необходимо пробросить по узкому каналу много телеканалов и распределить их между ретрансляторами в датацентре.

Таймауты

Вы можете попробовать различные значения таймаутов в конфигурации, но будьте очень осторожны. Установка слишком малых таймаутов может сделать систему нерабочей.

Запомните очень важный факт: в сети невозможно отличить потерю связи от очень долгой задержки.

Figure 5.11: Flussonic cluster ingest

Таймауты для кластерного захвата задаются в миллисекундах.

Пример конфигурации кластера с использованием таймаутов:

```
cluster_key abcd;
peer flussonic1:8080;
peer streamer:8081 {
  fetch_timeout 1000;
  stale_timeout 3000;
}
```

Такая конфигурация сообщит *Flussonic Media Server*, что получать потоки от пиров нужно раз в 1 секунду (1000 миллисекунд). Это ОЧЕНЬ часто, и такое значение не следует использовать в

Figure 5.12: Отображение текущего сервера

рабочей среде. Но вы можете использовать его для тестирования. За это отвечает параметр `fetch_timeout`.

`stale_timeout 3000;` — сообщит *Flussonic Media Server*, что потоки от этого пира недоступны, после 3 секунд (3000 миллисекунд) отсутствия ответа от этого пира.

Таким образом, если этот пир перегружен и не может ответить через 3 секунды, он считается недействующим, и механизм кластеризации запустит свои потоки на локальном хосте.

5.1.13 Резервирование транскодеров с cluster ingest

Механизм cluster ingest можно использовать не только для захвата видео из источника, но и для других сценариев, таких как транскодирование, запись архива (DVR) или отправка видео на другие серверы (например, Youtube). В каждом из этих сценариев каждый поток будет работать на одном из пиров, и если какой-либо пир выйдет из строя, потоки на нем будут полностью отключены, а другие пиры перезахватят эти потоки и будут выполнять с ними необходимые действия. В частности, механизм cluster ingest может быть очень полезен для транскодирования.

Пример сценария

Предположим, что у нас есть два сервера *Flussonic* (*граббера*), которые захватывают спутниковое видео с 200 каналами с головной станции IPTV DVB. Далее чересстрочное видео конвертируется в прогрессивное с помощью четырех других серверов *Flussonic* (*транскодеров*), объединенных в кластер, и наконец рассылается по сети. Мы будем использовать опцию `cluster_ingest` для отказоустойчивого транскодирования всех каналов и определения того, какой пир будет выполнять транскодирование того или иного канала.

Настройка грабберов Давайте настроем грабберы так, чтобы каждый из них захватывал половину телевизионных каналов. В конфигурации каждого граббера будет настроено 100 потоков для захвата соответствующих телевизионных каналов:

- grabber1: потоки 1-100
- grabber2: потоки 101-200

Конфигурация каждого потока будет выглядеть следующим образом:

```
stream dvb01 {
    input mpts-dvb://a1?program=1000;
}
```

Настройка транскодеров Транскодеры будут перезахватывать потоки с грабберов, так что на вход каждого из них придет по 50 потоков.

- transcoder1: потоки 1-50
- transcoder2: потоки 51-100
- transcoder3: потоки 101-150
- transcoder4: потоки 151-200

Чтобы объединить транскодеры кластер, добавим в конфигурацию каждого из них директиву `cluster_key`:

```
cluster_key abcd;
peer transcoder1;
peer transcoder2;
peer transcoder3;
peer transcoder4;
```

Здесь и далее мы предполагаем, что все серверы имеют действительные и разрешимые имена хоста.

Затем в конфигурацию каждого транскодера мы добавим по 50 потоков. Настройки всех этих потоков будут одинаковыми (за исключением имени перезахватываемого потока с граббера):

```
stream dvb01 {
  input m4f://grabber1/dvb01;
  transcoder vb=1k deinterlace=true ab=128k;
  cluster_ingest;
}
```

Если не использовать опцию `cluster_ingest` в конфигурации потоков, все транскодеры будут перезахватывать потоки и обрабатывать их одновременно. Но если какой-либо транскодер выйдет из строя, его потоки вообще не будут транскодироваться и зрители не смогут смотреть некоторые программы.

Использование опции `cluster_ingest` гарантирует, что в случае выхода из строя одного транскодера, его потоки будут перезахвачены другими транскодерами, которые продолжат их обработку. Каждый поток будет обрабатываться только одним транскодером. Если из строя выйдет другой транскодер, потоки снова будут перераспределены между пирами.

Имейте в виду, что при перераспределении потоков от нерабочего транскодера нагрузка на другие транскодеры может возрасти. Например, если все четыре транскодера работают одновременно, каждый из них может быть загружен примерно на 60-70 %, но если один из них выйдет из строя, нагрузка на остальные транскодеры вырастет до 90 %. Это следует принимать во внимане при выборе емкостей серверов для транскодирования.

5.2 Record to DVR

5.2.1 Работа с архивом DVR в Middleware

Эта статья должна помочь разработчику Middleware реализовать работу с архивом видео для реализации следующих задач пользователя:

- поставить прямой эфир на паузу, сходить покурить
- посмотреть текущую передачу с начала, потому что фильм понравился и надо сначала
- посмотреть вчерашнюю передачу
- отложить понравившуюся передачу в закладки, пересмотреть через пару месяцев

Проблема по которой существует эта статья заключается в том, что единственный протокол, который технически может решать эти проблемы - это RTSP, но от него давно отказались и перешли на сегментные HTTP: HLS, DASH (и немного MSS). Оба основных протокола для IPTV OTT не поддерживают эту функциональность из коробки и поэтому необходимы технологические исхищрения для решения этих задач.

Как же правильнее всего организовать доступ к архиву?

С помощью epg-vod проигрывания архива и event проигрывания лайва.

Проигрывание архива по EPG

Для работы этого метода, Middleware должна хранить у себя достаточно точное расписание EPG. Здесь и далее будут отсылки к понятию таймстемпа, времени. Мы рассматриваем только секунды в UTC (он же Epoch). Ни при каких обстоятельствах локальное время не рассматривается, потому что ничего кроме хаоса его использование тут не вносит.

При просмотре передачи рекомендуется сохранять текущее время просмотра в базу данных middleware, чтобы при повторном открытии предлагать продолжить или начать сначала.

Когда пользователь выбирает передачу из архива, которая шла с 1717677139 UTC до 1717679255, надо сформировать для плеера урл: `http://FLUSSONIC-IP/STREAM_NAME/archive-1717677139-211`

В плеере будет работать:

- **перемотка внутри передачи**
- **пауза**
- **ускоренный просмотр**

При завершении проигрывания рекомендуется переключить на следующую передачу в EPG, чтобы сохранять непрерывность поступления контента в потребителя.

Просмотр текущей передачи с помощью event-плейлистов

Просмотр текущей передачи технологически будет сложнее. Мы предлагаем использовать event плейлисты, но они в нативном Сафари не позволяют отмотку назад, поэтому для ТВ не годятся и тогда вам нужно писать на яваскрипте и реализовывать собственный таймлайн. Если Сафари для вас не критичен, то смело используйте точно те же урлы, что и в epg-vod:

```
http://FLUSSONIC-IP/STREAM_NAME/archive-1717677139-2116.m3u8?event=true
```

Хитрость в том, что если момент закрытия плейлиста в будущем, то он будет отдаваться не как VOD, а как EVENT, что позволит плееру его перезапрашивать и двигаться дальше.

После закрытия этого плейлиста надо так же переключиться на следующий по EPG и продолжить просмотр.

Использование такого урла позволит поставить лайв на паузу и продолжить с того же места. Особенно важно отметить, что кнопку «live» необходимо программировать самостоятельно, потому что за время удержания на паузе плейлист может автоматически превратиться из EVENT в VOD и закрыться. В этом случае надо по EPG выяснить, какая передача следующая и перепрыгнуть на неё.

5.2.2 Как сохранить записи телепередач для nPVR

Одно из важнейших отличий IPTV OTT сервиса от линейного ТВ — доступ к архиву передач. В этой статье описывается, как обеспечить сохранение телепередач, которые клиент отложил на длительное хранение. При этом оставшаяся ненужная часть архива должна удалиться за ненадобностью и очистить место на диске.

До 2023 года мы предлагали для решения этой задачи механизм DVR locks, но от него отказались из-за того, что он очень плохо масштабируется и ещё хуже подлежит резервированию.

В этом документе не рассматривается задача обеспечения прозрачного доступа к телепередачам при переносе архива с горячего хранилища в холодное в виде отдельных видеофайлов. Этой расширенной задачей занимается Flussonic Central, рекомендуется пользоваться им. Здесь описывается базовый способ с одним лишь Media Server.

Организационно-правовой вопрос

В некоторых странах запись архива оператором может быть или запрещена, или требовать отдельного согласования/регулирования.

При этом запись передачи на приставку для личных абонента, как правило, допускается. Запись на приставку является ощутимым удорожанием сервиса, ведь нужно каждому абоненту продать более дорогое устройство, в котором появляется дополнительная изнашивающаяся деталь.

nPVR (network Personal Video Recorder) является неплохим выходом из этой ситуации: абонент заказывает запись передачи, а она ему сохраняется на сервере и хранится по его запросу.

Схема решения

- Сервер клиента должен получать EPG и сохранять телепередачи в базу в виде отдельных записей.
- На основании пожеланий подписчиков, сервер клиента должен помечать востребованные нужные телепередачи, как защищенные от стирания.
- Media Server будет для каждого потока регулярно запрашивать с сервера клиента список эпизодов, которые надо оставить в архиве.
- Если в качестве этого списка эпизодов отдавать телепередачи, то они будут сохраняться в архиве пока хватит места на дисках.

Взаимодействие Media Server и сервера клиента

Для того, чтобы включился механизм опроса эпизодов, сервер клиента должен:

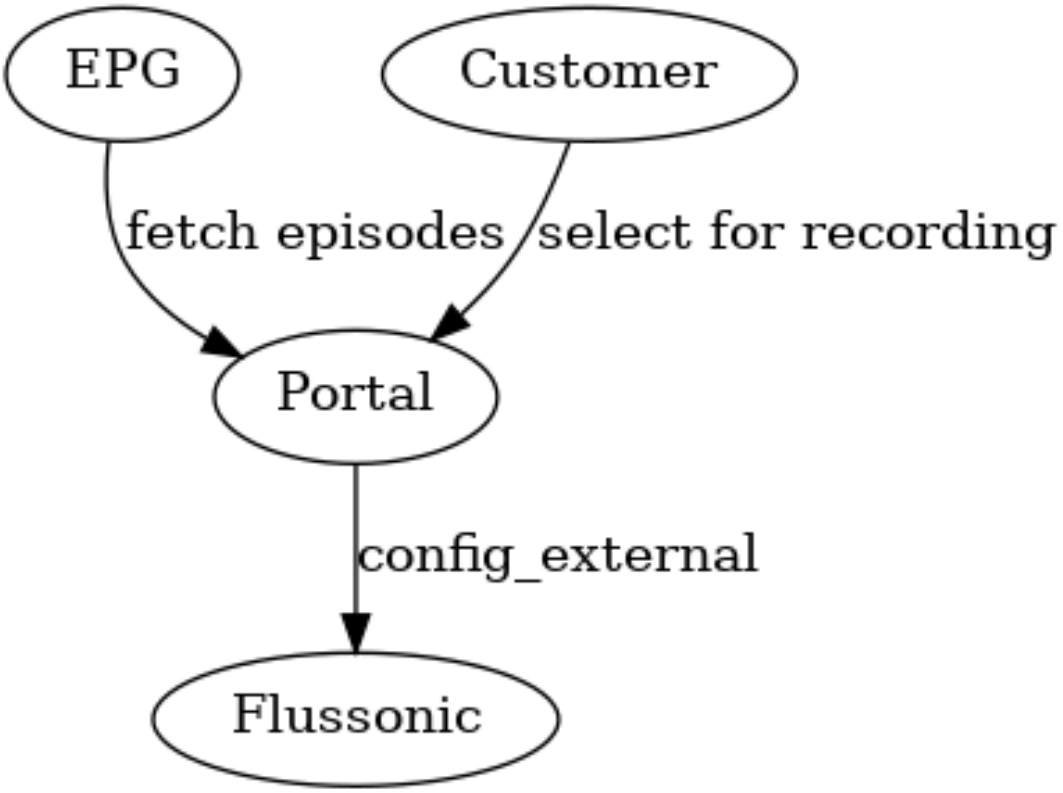


Figure 5.13

- Для работы эпизодов, Media Server должен использовать систему управления потоками config_external.
- В ответе со списком потоков надо отдать 7CX-Config-Server-Episodesundefined%7CX-Config-Server-Episodesзаголовок X-Config-Server-Episodes. Увидев этот заголовок, Media Server включит опрос эпизодов.
- Реализовать метод отдачи списка эпизодов.

В методе `external_episodes_list` очень важно обратить внимание на параметр `media` со списком запрашиваемых потоков. Отсутствие данных по потоку в ответе означает стирание всего архива для этого потока. Т.е. возврат пустого списка означает стирание архива со всех потоков, которые были перечислены в параметре `media`

В этом ответе допустимо вернуть код 400 или 500, если по какой-то причине готового ответа со списком эпизодов нет.

Как это будет работать?

- На Media Server настроен архив потока с глубиной, например, в 3 дня непрерывного архива.

- Все настройки пришли через `config_external` с сервера, отдающего сигнальный заголовок `X-Config-Server-Episodes: true`.
- На основании этого раз в какое-то время Media Server делает запрос к `external_episodes_list`, опрашивая диапазоны, которые планируются к очистке.
- Те фрагменты архива, которые покрыты эпизодами, остаются на месте, остальные удаляются.

Выгрузка записей в виде файлов

В случае, когда требуется долгосрочное хранение с редким обращением, имеет смысл выгрузить фрагменты архива в виде файлов в отдельное внешнее хранилище.

Этой задачей занимается Flussonic Central.

5.2.3 Как сделать отложенный просмотр в другом часовом поясе

Многие телеканалы вещаются с рассчётом только на один часовой пояс и если мы говорим про Россию, то зачастую это только московский часовой пояс.

Если хочется этот же канал отдавать пользователям в Германии или в США, то возникает неудобство: на часах у людей ещё раннее утро, а в телевизоре уже вечерние передачи.

Flussonic Media Server может отложить проигрывание потока на несколько часов, чтобы у людей в другом часовом поясе передача «Доброе утро» шла добрым утром, а не глубокой ночью.

Есть несколько технических способов организовать это в Flussonic Media Server исходя из частоты обращения к различным каналам в различных часовых поясах. Разница между способами заключается в том, сколько раз будет читать архив для отложенного показа канала. Можно запустить отложенный поток и тогда архив будет читаться один раз вне зависимости от количества желающих посмотреть, а можно отдать пользователям персональные URL-адраса и тогда архив будет читаться на каждого пользователя.

Если каналов пишется порядка 250 и хочется сделать вещание для 3-х локаций, то суммарно получается 250 каналов на запись и 750 на чтение. Некоторые каналы имеет смысл сделать постоянно запущенными, а некоторые только по запросу пользователей.

Отложенный поток

Пусть у нас есть настроенный канал:

```
stream channel {
  input fake://fake;
  dvr /storage 1d;
}
```

У канала должен быть настроен архив (в примере это `dvr /storage 1d`). Теперь можно сделать второй поток:

```
stream channel-1hour {
  input timeshift://channel/3600;
}
```

Этот поток будет вычитывать из архива и показывать то, что было один час назад (3600 секунд).

Таких потоков можно создавать столько, сколько нужно.

Персональный доступ к архиву

Если есть настроенный поток:

```
stream example_stream {
  input udp://239.1.2.3:1234;
```

Figure 5.14: Flussonic DVR Timeshift

```
  dvr /storage 1d;
}
```

то к нему можно выдать URL:

- для проигрывания по HTTP MPEG-TS:

```
http://FLUSSONIC-IP/example_stream/timeshift_rel/3600
```

для проигрывания по HLS:

```
http://FLUSSONIC-IP/example_stream/timeshift_rel-3600.m3u8
```

Для мультиязыковых каналов можно отдать следующий URL на приставки:

В этом случае каждый клиент будет отдельно читать архив. Такой метод стоит использовать для редко используемых сочетаний канала и часового пояса.

Пропуск «дырок» в архиве

В случае если в архиве есть незаписанные участки (например источник был недоступен несколько минут), то при проигрывании таймшифта по HLS Flussonic Media Server будет отдавать пустой плейлист при достижении незаписанного участка.

Если же допустимо нарушить временной сдвиг (таймшифт) и перепрыгнуть через эту «дырку», то следует запрашивать плейлист с параметром `?ignore_gaps=true`:

```
https://FLUSSONIC-IP/STREAM_NAME/timeshift_abs-123123123.m3u8?ignore_gaps=
    true
```

С timeshift_abs HLS URL-адресами есть большая сложность, связанная с природой HLS протокола. Дело в том, что Flussonic Media Server может лишь вероятностно связывать отдельные HTTP запросы в одну сессию. Flussonic Media Server считает, что сессия та же, если у двух запросов совпадает IP адрес клиента, имя канала, протокол запроса и токен. В случае с несколькими, идущими подряд timeshift abs запросами, Flussonic Media Server решит, что это одна и та же сессии. В итоге может получиться искажение просмотра. Чтобы избежать этого, следует передавать в timeshift_abs запрос новый токен.

Вариант попроще — запросить HTTP MPEGTS `http://FLUSSONIC-IP/STREAM_NAME/timeshift_abs-1`

Однако HTTP MPEGTS вариант лишает доступа к мультибитрейту.

5.2.4 Резервирование архива

При построении IPTV-сервиса возникают следующие задачи, связанные с архивом:

- сохранение видеоархива,
- постоянная доступность архива.

Для решения этих задач используйте резервирование. Для резервирования архива в Flussonic используется *кросс-репликация*.

На этой странице:

- Что такое кросс-репликация и зачем она нужна.
- Как работает кросс-репликация.
- Как настроить кросс-репликацию.

Что такое кросс-репликация и зачем она нужна

Кросс-репликация — это механизм резервирования архива, при котором два сервера Flussonic записывают и хранят архив потока, а также могут обращаться к источнику и восстанавливать недостающие части архива друг друга (подробнее см. Как работает кросс-репликация). Архивы на серверах синхронизированы и являются полными копиями друг друга. Если один из серверов станет недоступным, то второй продолжит вести запись архива, обращаясь к источнику напрямую. Идентичность копий архива будет восстановлена сервером Flussonic автоматически при возвращении неработающего сервера онлайн.

Кросс-репликация — это репликация с первого сервера на второй и со второго на первый.

Так с помощью кросс-репликации ваш сервис:

- Сохранит возможность просмотра записей архива для зрителей при аварии или временной недоступности одного из серверов с архивом.
- Восстановит недостающие сегменты архива по возвращении работоспособности севера путём передачи данных с работающего сервера.
- Обеспечит идентичность архивов на серверах.

Как работает кросс-репликация

Для передачи данных между серверами Flussonic рекомендуется использовать внутренний протокол Flussonic — M4F. О преимуществах протокола M4F читайте в разделе Обзор протокола M4F.

Механизм кросс-репликации в Flussonic применяется к архиву отдельного потока, а не сервера. Если вы хотите настроить кросс-репликацию для нескольких потоков на сервере, то настройте кросс-репликацию отдельно для каждого потока.

У механизма кросс-репликации есть три режима работы:

- штатный режим:

- Основной сервер захватывает прямой эфир из источника по UDP и записывает архив основного потока условного `example_stream`. - Резервный сервер захватывает прямой эфир и архив основного потока `example_stream` из основного сервера по протоколу M4F в резервный поток `replica_example_stream`.

- аварийный режим:

- Основной сервер выходит из строя, перестаёт принимать прямой эфир от источника и писать архив. - Резервный поток `replica_example_stream` переключается напрямую на источник. Резервный сервер принимает прямой эфир по UDP от источника и продолжает записывать архив.

- режим восстановления после аварии:

- Основной сервер восстановил работу и основной поток `example_stream` переключается на источник. Сервер захватывает прямой эфир из источника по UDP и пишет архив потока.
- Основной сервер забирает недостающую часть архива, которую записал резервный сервер во время простоя основного. - Резервный поток переключается обратно на основной сервер. Резервный сервер снова захватывает прямой эфир из основного сервера и продолжает записывать архив.

Как настроить кросс-репликацию

Чтобы включить кросс-репликацию для потока `example_stream` на основном и резервном серверах Flussonic, настройте на обоих серверах следующее:

- захват из источника (`input udp://`),
- захват из другого сервера Flussonic для репликации (`input m4f://`),
- запись архива (`dvr /storage 3d`) и репликацию, добавив настройку `replicate`.

Пусть primary_flussonic.example.com — основной сервер, а secondary_flussonic.example.com — резервный сервер.

Конфигурация основного потока на основном сервере primary_flussonic.example.com:

```
stream example_stream {
  input udp://224.1.2.3:1234;
  input m4f://secondary_flussonic.example.com/replica_example_stream;
  dvr /storage 3d replicate;
}
```

Конфигурация резервного потока на резервном сервере secondary_flussonic.example.com:

```
stream replica_example_stream {
  input m4f://primary_flussonic.example.com/example_stream;
  input udp://224.1.2.3:1234;
  dvr /storage 3d replicate;
}
```

5.2.5 Как быстро скопировать архив DVR на второй сервер и увеличить одновременное количество зрителей

На этой странице:

- Зачем нужна репликация архива?
- Как работает репликация.
- Как настроить репликацию.

Репликация — механизм копирования архива DVR (Digital Video Recorder) с основного сервера на резервные. Этот механизм позволяет автоматически восстанавливать недостающие части архива после аварии и в короткие сроки ввести сервер в работу.

Зачем нужна репликация архива?

Репликация во *Flussonic* предназначена для решения следующих задач:

- Расширить DVR-сервис и увеличить одновременное количество зрителей.
- Быстро ввести в работу новый DVR-сервер или DVR-сервер после аварии.
- Автоматически восстановить недостающие сегменты архива после аварии.

90% времени для обслуживания зрителей хватает основного сервера с архивом. Сервер записывает архив на диск и доставляет зрителям сегменты из любой временной точки архива. Так одни зрители смотрят запись прямого эфира через сутки после его завершения, а другие — 72 часа после. При этом каждый зритель получает уникальный контент. Вечером, когда люди приходят с работы и учёбы, количество зрителей резко возрастает. Из-за этого увеличивается нагрузка на чтение с диска основного сервера. Чтобы обслужить возрастающее количество зрителей, не перегрузив при этом основной сервер:

- Подключите резервный сервер.
- Скопируйте на резервный сервер архив основного сервера.
- Направляйте новых зрителей на резервный сервер.

По собранным нами данным статистики просмотров контента DVR, время между запросом одного и того же сегмента архива первым и вторым зрителем составляет 18 часов. Получается, что вам дешевле и проще сделать копию архива на резервном сервере, чем настроить локальный кеш. Так зрители смогут смотреть контент непрерывно и без задержек в вещании.

С подключением резервного сервера с архивом пропускная способность сервиса и скорость чтения с диска увеличиваются в два раза, что позволяет обслуживать в два раза больше зрителей.

Главная задача основного сервера — не перегрузить канал репликации на резервный сервер, одновременно раздавая контент зрителям.

Когда вам нужно в короткие сроки ввести резервый DVR-сервер в работу, чтобы доставлять зрителям сегменты архива, используйте репликацию. Вместо того, чтобы записывать архив, глубиной в неделю, с нуля на резервном сервере и ждать семь дней, вы можете за несколько часов скопировать архив с основного сервера и подготовить резервный сервер к работе. Сегменты архива синхронизированы по времени, так что зрители не увидят разницы в получаемом контенте.

Как работает репликация

Схема 1. Схема работы репликации

Основной сервер:

- получает прямой эфир из источника,
- записывает его в архив,
- вещает зрителям прямой эфир и сегменты архива.

Основной сервер не знает о существовании резервного. Как только резервный сервер включается, он делает следующее:

- Подключается к основному серверу.
- Захватывает прямой эфир и копирует архив из основного сервера. Из-за этого увеличивается скорость чтения с диска основного сервера и скорость записи на диск резервного сервера.
- Как только архив резервного сервера заполнится, резервный сервер начинает доставлять сегменты архива новым зрителям.

Как настроить репликацию

Вы можете настроить репликацию для всех потоков или для отдельных из них.

Репликацию можно также настроить через API с помощью настроек 7Cdvr-replicatetag/dvr/operation/dvr-get/response%7Cdvr-replicate‘dvr_replication‘ и 7Creplication-porttag/dvr/operation/dvr-get/response%7Creplica port‘replication_port‘.

Репликация всех потоков Репликацию всех потоков можно настроить как в пользовательском интерфейсе, так и в конфигурационном файле `/flussonic/flussonic.conf`.

В пользовательском интерфейсе Чтобы включить репликацию всех потоков с основного сервера на резервный в пользовательском интерфейсе *Flussonic*, следуйте шагам ниже:

- На резервном сервере в выпадающем меню слева откройте раздел **Config** и перейдите на вкладку **DVR**.
- В разделе **Additional** поставьте галочку напротив *Dvr replicate*. При необходимости также укажите отдельный порт для репликации в поле *Replication port* (подробнее см. Как указать отдельный порт для репликации).
- Сохраните настройки, нажав на значок сохранения в правом верхнем углу.

После этого начнётся репликация архива на резервный сервер.

В конфигурационном файле Чтобы включить репликацию всех потоков с основного сервера на резервный, укажите на резервном сервере директиву `source` с опцией `replicate` в настройках DVR:

```
cluster_key abcd;
source primary-server {
  dvr /storage 20d replicate;
}
```

Репликация одного потока Репликацию потока можно настроить как в пользовательском интерфейсе, так и в конфигурационном файле `/flussonic/flussonic.conf`.

В пользовательском интерфейсе Чтобы включить репликацию отдельного потока с основного сервера на резервное в пользовательском интерфейсе *Flussonic*, выполните следующие шаги:

- На резервном сервере создайте новый поток и в качестве источника укажите поток из основного сервера, архив которого требуется скопировать, через протокол M4F или M4S. Сохраните настройки потока, нажав **Save**.
- Откройте вкладку **DVR** в настройках потока и укажите необходимую глубину хранения архива, на которую резервный сервер будет копировать архив с основного сервера.
- Поставьте галочку напротив *Dvr replicate*. При необходимости также укажите отдельный порт для репликации в соседнем поле *Replication port* (подробнее см. Как указать отдельный порт для репликации).
- Сохраните настройки, нажав **Save**.

После этого начнётся репликация архива на резервный сервер.

Figure 5.15: DVR replicate M4F-источник

В конфигурационном файле Чтобы включить репликацию отдельного потока с основного сервера на резервный, выполните следующие шаги в конфигурационном файле `/flussonic/flussonic.c` резервного сервера:

- Создайте новый поток и в качестве источника укажите поток из основного сервера, архив которого требуется скопировать, через протокол M4F или M4S: `input m4f://PRIMARY-SERVER-IP/S`
- В настройках потока укажите необходимую глубину хранения архива, на которую резервный сервер будет копировать архив с основного сервера: `dvr /STORAGE_NAME 7d`.
- В настройках потока в DVR `dvr` укажите опцию `replicate`. При необходимости также укажите отдельный порт для репликации в параметре `replication_port=` (подробнее см. Как указать отдельный порт для репликации).

Figure 5.16: DVR replicate в интерфейсе

Пример конфигураций на основном и резервном серверах:

- **Конфигурация потока на основном сервере**:

```
stream fake {
  input fake://;
  dvr /storage 7d;
}
```

- **Конфигурация потока на резервном сервере**:

```
stream repl_example1 {
  input m4f://primary-server-ip/fake;
  dvr /storage 7d replicate;
}
```

Так к источнику потока подключается только основной сервер, а резервный сервер захватывает поток с основного сервера в отличие от кросс-репликации.

При наличии ключевого слова `replicate` запись будет включена постоянно, поэтому не используйте опцию 'dvr_offline', которая отключает запись архива, вместо `dvr`.

Репликация работает только по внутреннему протоко
рекомендуем использовать этот протокол для передачи ви
sonic. О преимуществах протокола M4F можно прочитать
M4F.

Как указать отдельный порт для репликации

По умолчанию репликация включается на порту, указанном при настройке M4F-источника. Чтобы избежать перегрузки канала и остановки работы вашего сервиса, вы можете указать отдельный порт для репликации архива.

В пользовательском интерфейсе

- Если вы настраиваете репликацию для конкретного потока, то на резервном сервере в настройках потока перейдите на вкладку **DVR**. Поставьте галочку напротив *DVR replicate* и укажите порт для репликации в поле *Replication port*.

Нажмите **Save**, чтобы сохранить настройки.

- На основном сервере в выпадающем меню слева откройте раздел **Config**. На вкладке **Settings** в разделе **Listeners** укажите такое же HTTP-порт, как в предыдущем шаге, нажав на иконку добавления. Нажмите **Save**, чтобы сохранить настройки.

В конфигурационном файле

- В конфигурационном файле резервного сервера укажите номер порта в параметре `replication_port` рядом с `replicate` в настройках потока, если вы настраиваете репликацию для конкретного потока, или директивы `source`, если для всех потоков:

```
stream repl_example2 {
  input m4f://primary-server-ip/fake;
  dvr /storage 7d replicate replication_port=8002;
}
```

- В конфигурационном файле основного сервера укажите такой же порт через опцию `http`:

```
http 8002
```

Читайте также об Использовании кросс-репликации для восстановления архива.

5.2.6 DVR в облаке

Хранение архива в облаке

Flussonic Media Server может писать видеоархив в HTTP хранилище, например, Amazon S3 или OpenStack Storage (Swift).

Поддерживается два способа записи в облачное хранилище:

- Запись потока посегментно напрямую в облачное хранилище (по умолчанию). В таком случае архив в облаке всегда будет актуален, но это может стоить дорого, если провайдер тарифицирует каждую операцию записи. Кроме того, простые S3-совместимые сервисы (например, MinIO) могут не справиться с таким большим количеством файлов, а для сервиса с несколькими десятками каналов и несколькими днями записи уже счет идет на миллионы файлов.
- Запись более длительными фрагментами длительностью один час (если используется параметр **copy**, см. ниже). В этом случае потребуется локальное хранилище, где будет накапливаться запись за этот час. Этот способ экономит операции записи в облачное хранилище и обеспечивает устойчивость к кратковременным проблемам с сетью, поскольку часовой фрагмент передается асинхронно.

Примеры конфигурации

Для записи на **Amazon S3** необходимо сконфигурировать поток следующим образом:

```
stream chan1 {
  input fake://fake;
  dvr s3://minioadmin:minioadmin@minio:9001/test 10G;
}
```

Для записи на **Amazon S3** и доступа по **HTTPS**, необходимо сконфигурировать поток так:

```
stream chan5 {
  input fake://fake;
  dvr s3s://minioadmin:minioadmin@minio:9001/test 10G;
}
```

Для записи на **OpenStack Storage (Swift)** сконфигурируйте поток следующим образом:

```
stream chan2 {
  input copy://chan1;
  dvr swift://user=test:tester&password=testing@swift:8080/test 10G;
}
```

Для записи на **Akamai storage** сконфигурируйте поток следующим образом:

```
stream chan3 {
  input copy://chan1;
  dvr akamai://keyName:keyValue@akamaihd.net/cpCode/dvr 10G;
}
```

Копирование архива в облако

Параметр **copy** позволяет значительно снизить количество обращений к диску при записи в облачное хранилище.

При использовании копирования *Flussonic* сначала записывает поток на локальный диск (в указанную директорию). Затем, каждый час, он переносит записанные данные в хранилище.

Указывать параметр **copy** нужно так:

```
stream chan4 {
  input copy://chan1;
  dvr /storage copy=s3://minioadmin:minioadmin@minio:9001/test 10G;
}
```

Запись в сетевое хранилище при миграции потока

Группа серверов *Flussonic* может работать с одним сетевым хранилищем, при этом запись ведется в один каталог. При переносе потока с одного сервера на другой новый сервер будет подхватывать запись, сделанную старым.

Flussonic полностью перенесет конфигурацию потока на новый сервер, а архив продолжит работу автоматически.

> Несколько серверов не должны писать один и тот же поток одновременно.

5.2.7 Запись архива DVR на NAS NFS

Развертывание виртуализированных корпоративных окружений часто подразумевает использование сетевого хранилища, как надежного способа сохранить образ виртуальной машины и запустить её на другом сервере. При этом к одному сетевому хранилищу могут подключаться десятки или даже сотни виртуальных машин.

Т.е. сетевое хранилище используется для конфигураций, когда вычислительных мощностей сильно больше, чем данных и трафик с каждой вычислительной ноды относительно небольшой.

Этот опыт иногда пытаются перенести на хранение видео архива и возникает вопрос: как использовать сетевое хранилище в задачах видеонаблюдения или телевидения.

Простой ответ: никак. Это экономически и инженерно неоправдано. Огромный перерасход средств и гарантированные проблемы со стабильностью записи по NFS.

Сетевое хранилище представляет из себя сервер со специализированным для хранения и произвольной записи ПО, причем зачастую размер сетевого хранилища зачастую меньше, чем места на сервере с 36 дисками, собранном под видеонаблюдение.

В итоге вы просто покупаете два сервера, соединяете их по сети и с одного копируете на другой. Никакого эффекта консолидации хранения, подключив 10 видеосерверов к одному хранилищу вы не получите.

Резервирование хранения

Как правило на сетевом хранилище работает какая-то форма RAID с защитой от поломки одного-двух дисков. Сетевое хранилище, устойчивое к поломке целого сервера - большая редкость.

Если вам нужно надежное хранение архива, то лучше не рассчитывать на одну точку отказа, а построить резервируемый кластер.

Как подключить хранилище, если очень хочется?

Обычно сетевые хранилища монтируются по протоколу NFS. В Flussonic нет встроенной поддержки NFS клиента, вместо этого предполагается использование штатного клиента в Linux.

Необходимо знать, что если какие-то пакеты в NFS трафике теряются, сам стример может быть заморожен и будет не отвечать бесконечно, потребуется перезагружать сервер, это особенности реализации NFS в ядре.

Вам могут помочь опции `soft`, `intr`, `timeo`, `retrans`. Для более детальных объяснений стоит обратиться к мануалам операционной системы: `man nfs`

5.3 Playback

5.3.1 IPTV плагин

Во *Flussonic Media Server* есть IPTV плагин, который представлет собой встроенную простейшую IPTV панель. Это по сути Middleware с упралением пользователями и разрешениями на просмотр каналов. IPTV плагин подходит как для организации сервиса для многих сотен клиентов, так и раздачи потоков друзьям и партнерам. Его также можно использовать в качестве авторизационного бекенда.

Flussonic Media Server хранит базу данных в статическом файле JSON на диске, который перезаписывается при каждом обновлении.

Управлять настройками IPTV плагина можно как через пользовательский интерфейс *Flussonic Media Server*, так и через Flussonic API.

На этой странице:

- Включение IPTV плагина
- Управление пакетами
- Управление пользователями
- Генерация плейлиста
- Мультиавторизация

Включение IPTV плагина

Чтобы включить IPTV плагин, перейдите во вкладку **IPTV** и нажмите **Enable IPTV**. Откроется страница **IPTV** с двумя вкладками:

- **Users** – управление пользователями. Подробнее см. Управление пользователями.
- **Packages** – управление пакетами, т.е. списками каналов, доступных пользователям. Подробнее см. Управление пакетами.

Управление пакетами

Пакет (Package) – это набор потоков (каналов), которые предоставляются пользователю вместе, как одна единица биллинга. Пакетами можно управлять на вкладке **Packages** страницы **IPTV**.

Чтобы создать пакет, нажмите на знак плюса слева от вкладки **Users**, введите имя пакета, выберите каналы, которые будут доступны в этом пакете, и нажмите **Save**.

На странице отображается список всех пакетов. Здесь вы можете:

- изменить имя пакета

Figure 5.17: IPTV create package

- добавить каналы в пакет или удалить их из него
- удалить пакет
- отфильтровать пакеты по имени

Все эти операции также можно выполнить с помощью Flussonic API. См. справочник API.

Управление пользователями

Пользователь (User) – это подписчик IPTV с доступом к одному или нескольким пакетам. Управлять пользователями можно на вкладке **Users** страницы **IPTV**.

Figure 5.18: IPTV packages

Чтобы создать пользователя, введите его имя, укажите максимальное количество сессий (одновременных подключений) и выберите доступные пакеты из списка предварительно настроенных пакетов. Затем нажмите **Save**. *Flussonic Media Server* создаст пользователя и автоматически сгенерирует токен для его авторизации. Позже вы можете изменить этот токен.

Внизу страницы отображается список всех пользователей. Здесь вы можете:

- изменить имя, токен или максимальное количество сессий пользователя
- добавить или удалить пакеты, доступные пользователю
- удалить пользователя
- отфильтровать пользователей по имени

Figure 5.19: IPTV create user

- сгенерировать плейлист MPEG-TS или HLS для пользователя. См. Генерация плейлиста.

Все эти операции также можно выполнить с помощью Flussonic API. См. справочник API.

Генерация плейлиста

Вы можете получить m3u плейлист для существующего пользователя для проигрывания по HLS или HTTP MPEG-TS.

Для получения плейлиста нажмите **HLS** или **MPEG-TS** в строке соответствующего пользователя.

Скачанный плейлист будет содержать следующие теги:

Figure 5.20: IPTV users

- `tvg-id` – EPG ID канала. Здесь подставляется имя канала.
- `group-title` – группа, к которой относится канал. Чтобы присвоить канал группе, добавьте директиву `meta group` в конфигурацию соответствующего потока:

```
stream channel01 {
  input fake://fake;
  meta group "Sports";
}
```

Пример плейлиста Допустим, пользователь `User1` имеет доступ к пакету `Mypackage1`. Этот пакет содержит каналы `channel01`, `channel02` и `channel05`. Канал `channel01` относится

к группе ”Sports”, а канал channel02 – к группе ”Nature”. В этом случае скачанный плейлист HLS будет выглядеть следующим образом:

```
#EXTM3U
#EXTINF:-1 tvg-name="channel01" tvg-id="channel01" group-title="Sports",
   channel01
https://demo.flussonic.com/channel01/video.m3u8?token=92zSzw5ve94p01
#EXTINF:-1 tvg-name="channel02" tvg-id="channel02" group-title="Nature",
   channel02
https://demo.flussonic.com/channel02/video.m3u8?token=92zSzw5ve94p01
#EXTINF:-1 tvg-name="channel05" tvg-id="channel05",channel05
https://demo.flussonic.com/channel05/video.m3u8?token=92zSzw5ve94p01
```

Генерация плейлиста также доступна с помощью Flussonic Streaming API. См. Streaming API reference.

Мультиавторизация

Плагин является частным случаем авторизационного бэкенда. Про механизм авторизации сессий во *Flussonic Media Server* можно прочитать здесь. Это означает, что он совместим с другими http-бэкендами, например, Stalker.

Подробнее про мультиавторизацию можно прочитать здесь.

Пример конфигурации с мультиавторизацией с использованием IPTV плагина смотрите здесь.

5.3.2 Middleware Stalker и Flussonic Media Server

Stalker middleware

Stalker — популярная бесплатная middleware от компании Инфомир. Stalker поддерживает работу с нашим архивом и системой авторизации.

Эта статья поможет вам настроить работу Flussonic Media Server совместно со Stalker.

Авторизация

Со стороны Flussonic Media Server Со стороны Flussonic Media Server необходимо добавить одну строку в flussonic.conf:

```
on_play http://<stalker_host>/stalker_portal/server/api/
    chk_flussonic_tmp_link.php;
```

После этого не забудьте перезагрузить конфигурацию:

```
service flussonic reload
```

Со стороны Stalker При создании/редактировании канала в модальном окне **Ссылки для вещания** необходимо из выпадающего списка **Temporary URL** выбрать Flussonic.

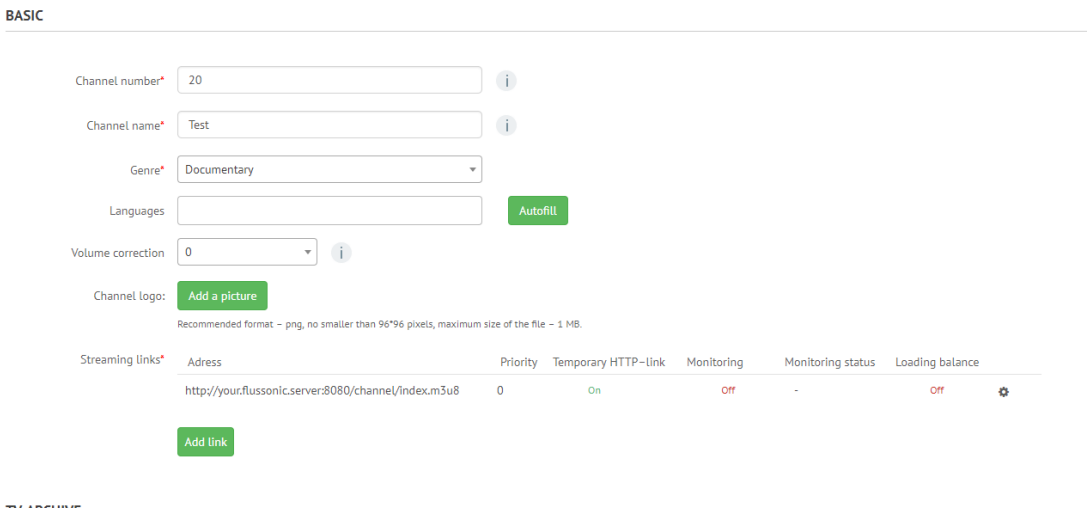


Figure 5.21: Stalker Middleware

На этом настройка закончена. Теперь в административном интерфейсе Flussonic Media Server можно будет увидеть, что пользователи используют токены для авторизации.

Архив

Со стороны Flussonic Media Server Дополнительная настройка не требуется. Просто убедитесь, что у вас включен архив на нужных каналах.

Со стороны Stalker

- Добавьте хранилище:

В панели администратора Stalker перейдите в меню **Хранилище** > **Список хранилищ**.

Нажмите кнопку **Добавить хранилище**.

Заполните поля **Название**, **IP**, **Порт** и во вкладке **Дополнительная информация** выберите **Flussonic DVR**.

- Включите архив у канала:

В панели администратора Stalker перейдите в меню **IPTV каналы** > **Каналы**.

Нажмите **редактировать** напротив канала, где хотите включить архив.

Перейдите в раздел **ТВ Архив**, выберите тип архива **Flussonic DVR**. Выберите сервер архива, который вы создали ранее. Заполните поле **Адрес для записи ТВ архива** (например, для HLS: `http://flussonic:80/streamname/index.m3u8`).

- Настройте EPG.

Документация от Инфомира

На сайте Инфомира тоже есть документация по настройке Stalker+Flussonic. Она может быть новее или старее. Если что-то не получается сделать по этой инструкции, попробуйте воспользоваться документацией от Инфомира: http://wiki.infomir.eu/doku.php/en:stalker:flussonic.

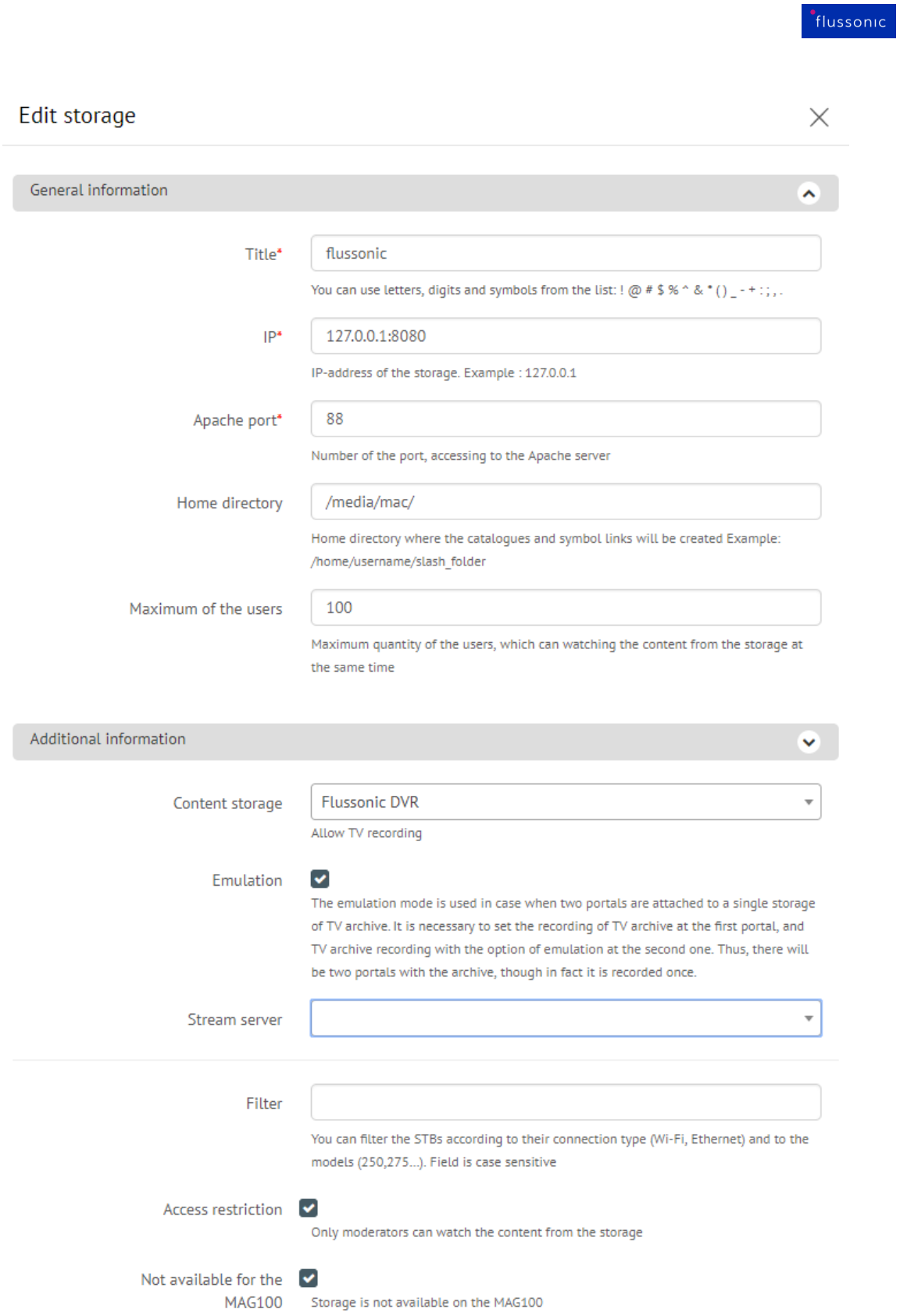


Figure 5.22: Stalker

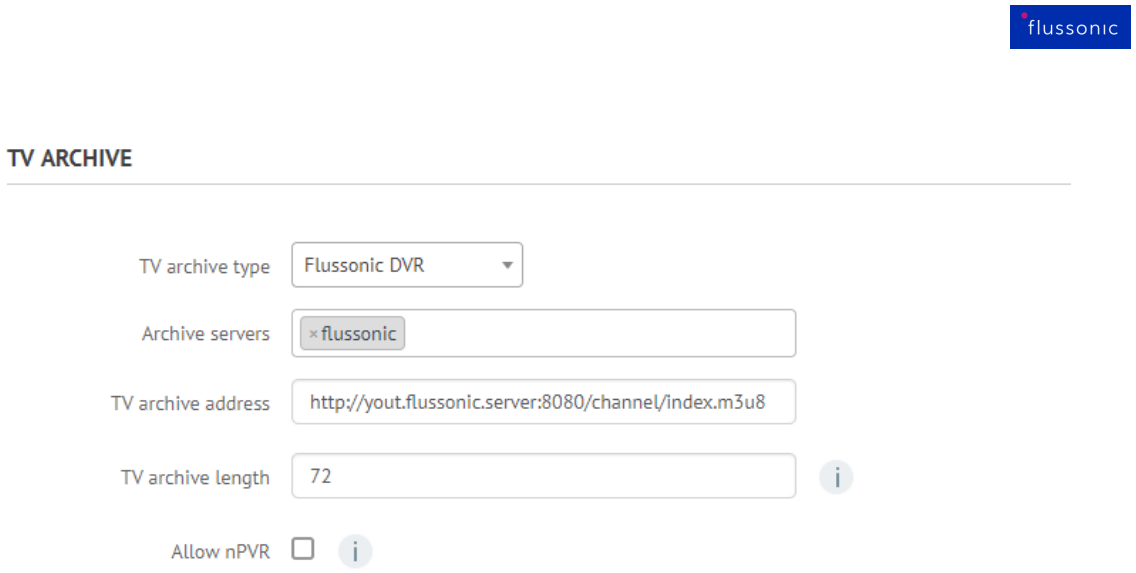


Figure 5.23: Middleware

5.3.3 Как перенаправить клиента на сервер с контентом

Flussonic Media Server может соединиться с другим *Flussonic Media Server*, получить список работающих и доступных по запросу потоков и перенаправлять клиентов на нужный пир, используя свои кластерные возможности.

Настройка

Создание пира Чтобы добавить пир во *Flussonic Media Server*, перейдите на страницу **Cluster** > **Settings**. Введите ключ кластера и имя нового пира, а затем нажмите **Save**.

Параметр `cluster_key` должен быть одинаковым на всех серверах в кластере.

> При добавлении пира через Flussonic UI в поле **New peer hostname** нужно указать адрес без порта. Вы можете указать порт позже, в поле **API URL** в настройках созданного пира. См. Настройки пира.

Вы также можете добавить пир в конфигурационном файле *Flussonic Media Server*:

```
cluster_key abcd;
peer streamer:8081;
```

Можно задать несколько пиров:

```
cluster_key abcd;
peer peer1.example.com;
peer peer2.example.com;
peer peer3.example.com;
```

Figure 5.24: Flussonic cluster peering

Все пиры могут принимать различные потоки, *Flussonic Media Server* будет направлять клиентов на нужный пир.

Статистика пиров Вы можете посмотреть статистику по потокам, захваченным всеми пирами в кластере, на странице **Cluster** > **Overview**.

Для каждого пира можно посмотреть:

- Использование CPU
- Использование памяти
- Количество подключенных клиентов

Figure 5.25: Cluster overview

- Количество захваченных потоков
- Выходной битрейт
- Процент нагрузки на сетевой интерфейс
- Время работы сервера (uptime)

Настройки пира Чтобы посмотреть или изменить настройки отдельного пира, перейдите в раздел **Cluster** > **Settings** > нажмите значок редактирования напротив нужного пира.

Откроется страница с настройками пира:

Здесь можно отредактировать следующие настройки пира:

Figure 5.26: Cluster peer settings

- **API URL** – внутренний адрес для общения по локальной сети (может содержать номер порта).
- **Public payload URL** – публичный адрес, который показывается клиентам.
- **Private payload URL** – внутренний адрес для общения по локальной сети (по умолчанию совпадает с API URL).
- **Fetch timeout** – с какой частотой пир будет пытаться получить данные с удаленного сервера по внутреннему API (в миллисекундах).
- **Stale timeout** – время, спустя которое удаленные потоки на этом сервере считаются неактивными и не могут использоваться в механизме `cluster_ingest`.
- **Channel limit** – максимальное количество потоков.

- **CPU limit** – ограничение на загрузку CPU в процентах.
- **Cluster key** – ключ для авторизации при общении между серверами Flussonic. Ключ должен быть одинаковым на всех серверах в кластере.

Внутренние и внешние адреса пиров Когда серверы *Flussonic* находятся в локальной сети и в настройках вы используете внутренние адреса (имена хостов), которые необходимы для общения пиров между собой внутри локальной сети, эти внутренние адреса могут быть видны на клиентах при проигрывании потока.

Можно показывать вовне внешние адреса пиров, но при этом разрешить им использовать внутренние адреса для взаимодействия внутри кластера. Для этого укажите публичный адрес пира в поле **Public payload URL** во вкладке **Settings** (см. Настройки пира).

Вы также можете добавить в конфигурационном файле в настройках каждого пира опцию `public`, в которой будет указан внешний адрес этого пира:

```
cluster_key abcd;
peer streamer:8081 {
    public streamer.public;
}
```

При перенаправлении на `streamer1.example.com` будет использоваться тот же протокол, HTTP или HTTPS, который указан в настройках.

Перенаправление

Flussonic Media Server будет перенаправлять клиентов на другой сервер, когда они запрашивают поток.

Между пиром (peer) и источником (source) существует очень важное различие, потому что source спроектирован для копирования видео по выделенному каналу, от источника (origin) на вещающий сервер (edge).

Пиринг спроектирован для случаев, когда клиент может получать видео от любого сервера в группе, а сервера (пиры) в группе общаются друг другом, сообщая какие потоки они имеют.

Когда клиент подключается по HLS, HTTP MPEG-TS, RTSP, RTMP или открывает `embed.html` на любой сервер в группе, он может быть перенаправлен на другой сервер, где этот поток действительно находится.

5.3.4 Балансировка нагрузки во Flussonic

Балансировка нагрузки — это процесс распределения запросов клиентов между кластером серверов в соответствии с некоторым алгоритмом.

Балансировка преследует следующие цели:

- предотвращение перегрузки одного из серверов кластера;
- оптимизация использования ресурсов группы серверов;
- максимизация пропускной способности.

Кроме того, балансировщик обладает свойствами масштабируемости и отказоустойчивости. Таким образом, по мере необходимости вы можете добавлять серверы в кластер и, если один из серверов выйдет из строя, то балансировщик сможет обеспечить непрерывную работу с клиентскими запросами.

Flussonic может распределять пользователей между несколькими серверами *Flussonic Media Server*. Балансировка нагрузки достигается за счёт перенаправления запросов клиентов. Поддерживается балансировка как запросов на проигрывание, так и запросов на публикацию.

Существуют разные алгоритмы балансировки нагрузки. Выбор алгоритма зависит от вашей цели и задач. Во *Flussonic* реализованы следующие методы (режимы) работы балансировщика:

- **Наименьшее количество подключений** (режим `clients`):

Направляет клиентский запрос на сервер с наименьшим количеством активных подключений. Эффективен в случаях, когда в кластере присутствует большое количество активных подключений, неравномерно распределённых между серверами. Никаких параметров для настройки указывать не нужно.

- **Наименьший выходной битрейт** (режим `bitrate`):

Направляет клиентский запрос на наименее загруженный сервер по значению выходного битрейта. Можно указать значение параметра `max_bitrate` (в бит/с, можно также указать в Мбит/с `40M` или Кбит/с `40K`) — максимальное значение битрейта для каждого пира. Балансировщик будет направлять трафик на сервер с наименьшим текущим выходным битрейтом.

- **Наименьшая загрузка** (режим `usage`):

Распределяет клиентские запросы, исходя из значения границы пропускной способности сервера. В этом случае необходимо указать `max_bitrate` (в бит/с, можно также указать в Мбит/с `40M` или Кбит/с `40K`) — максимальное значение битрейта для каждого пира. Балансировщик будет

вычислять загрузку канала как процент от максимального битрейта (current bitrate / max_bitrate * 100) и направлять трафик на сервер с наименьшей загрузкой.

!!! warning `max_bitrate` необходимо указывать для **каждого** сервера в кластере, иначе балансировщик будет работать некорректно.

- **Наименьшее количество активных потоков** (режим `streams`):

Распределяет клиентские запросы, исходя из количества активных потоков. Это режим подходит для распределения публикуемых потоков, полученных через `m4s://` от других серверов *Flussonic* (серверов захвата). Например, если вы публикуете потоки по WebRTC, RTMP или SRT на пуле серверов захвата, а затем отправляете их на пул серверов-транскодеров, каждый публикуемый поток будет перенаправлен на транскодер с наименьшим количеством активных потоков.

В каком случае необходимо использовать балансировку нагрузки?

Если у вашей стриминговой платформы или вашего стримингового сервиса более 10 000 зрителей.

Для того, чтобы использовать балансировщик во *Flussonic*, добавьте его в файл конфигурации (`/etc/flussonic/flussonic.conf`):

```
cluster_key SOME_CLUSTER_KEY;
balancer lb0 {
  mode bitrate;
  server p1 max_bitrate=60M;
  server p2 max_bitrate=40M;
  server p3 max_bitrate=30M;
}
```

Укажите следующие параметры:

- **lb** — имя балансировщика;
- **server** — пир (например, `peer1.example.com`);
- **mode** — режим балансировки (`bitrate`, `usage`, `clients`, `streams`). По умолчанию используется `mode bitrate`.

*

`http://FLUSSONIC-IP/lb/channel1/index.m3u8`

Вы также можете определить несколько балансировщиков, если в этом есть необходимость.

Балансировщик нагрузки работает со стриминговыми протоколами на основе HTTP, такими как HLS, MPEG-DASH, а также с WebRTC WHIP/WHEP.

Как настроить балансировщик

- Определите группу серверов для распределения нагрузки и укажите один и тот же `cluster_key` в конфигурационном файле каждого из них, чтобы связать их. На серверах настройте необходимые вам потоки.
- Выберите один или несколько серверов для балансировки. Запускайте столько балансировщиков в вашем кластере, сколько необходимо для достижения требуемого уровня отказоустойчивости. Это могут быть и выделенные сервера для балансировки, и Edge-сервера.
- Настройте балансировщик с помощью `balancer` в файле конфигурации сервера (также не забудьте указать `cluster_key`):

```
cluster_key SOME_CLUSTER_KEY;
balancer lb0 {
  mode bitrate;
  server stream.example.com max_bitrate=60M;
  server stream.example.tv max_bitrate=40M;
  server stream.exmpl.com max_bitrate=30M;
}
```

В примере выше мы определили балансировщик `lb0` с 3-мя серверами и режимом балансировки по значению выходного битрейта (`mode bitrate`).

Пример настройки балансировщика по количеству активных зрителей (`mode clients`):

```
cluster_key SOME_CLUSTER_KEY;
balancer lb0 {
  mode clients;
  server stream.example.com;
  server stream.example.tv;
  server stream.exmpl.com;
}
```

Пример настройки балансировщика по значению загрузки канала (`mode usage`):

```
cluster_key SOME_CLUSTER_KEY;
balancer lb0 {
  mode usage;
  server stream.example.com max_bitrate=60M;
  server stream.example.tv max_bitrate=40M;
  server stream.exmpl.com max_bitrate=30M;
}
```

Пример настройки балансировщика по количеству активных потоков (`mode streams`):

```
cluster_key SOME_CLUSTER_KEY;
balancer lb0 {
  mode streams;
  server trancoder1;
  server trancoder2;
```

```
  server trancoder3;
}
```

Балансировка с учетом GeoIP

Балансировщик может распределять клиентские запросы по серверам в кластере с учетом региона клиента. Вот как это работает:

- Для каждого сервера в кластере вы можете указать один или несколько кодов страны в параметре `countries` и/или опцию `country_default=true`, что означает "все остальные страны".
- *Flussonic* определяет страны, используя свободную базу данных геолокации GeoLite2 от MaxMind.
- Для каждого запроса клиента балансировщик нагрузки ищет сервер с кодом страны, соответствующим региону клиента. Если такой сервер найден, запрос перенаправляется на него. Если найдено несколько серверов с соответствующим кодом страны, балансировщик определяет среди них нужный согласно режиму балансировки (например, `bitrate`).
- Если сервер с нужным кодом страны не найден, запрос перенаправляется на сервер, для которого указано `country_default=true`. Если таких серверов несколько, балансировщик определяет среди них нужный согласно режиму балансировки (например, `bitrate`). Если такого сервера нет, запрос отклоняется.

Пример:

```
cluster_key SOME_CLUSTER_KEY;
balancer lb0 {
  mode bitrate;
  server p1 max_bitrate=60M countries=RU;
  server p2 max_bitrate=40M countries=RU,CN;
  server p3 max_bitrate=30M countries=CN countries_default=true;
  server p4 max_bitrate=30M countries_default=true;
}
```

С такой конфигурацией:

- Если запрос придет из Китая, он будет перенаправлен на P2 или P3 (в зависимости от битрейта).
- Если запрос придет из России, он будет перенаправлен на P1 или P2.
- Если запрос придет из любой другой страны, он будет перенаправлен на P3 или P4.

5.4 Protection

5.4.1 Авторизовать проигрывание с помощью токена

На сервере доступном в интернете почти сразу необходимо организовывать защиту просмотра, давать просмотр тем пользователям, которым можно и отзывать возможность просмотра у тех, кому больше нельзя. Flussonic, в дополнение к традиционным дорогостоящим системам типа DRM, предлагает недорогую и очень эффективную схему при которой портал, на который заходят зрители, выдает ссылку на проигрывание с включенным уникальным токеном, а сам медиа сервер уже проверяет этот токен.

Эта схема работает в тех случаях, когда контент раздается с медиасервера. Если вы используете CDN, то вам нужно или обращаться к поставщику CDN, или использовать DRM.

Настройка тестового примера

Установите триальную версию Flussonic и за несколько минут вы сможете запустить защищенный стрим для тестирования.

```
http 8080;
rtmp 1935;
rtsp 1554;

auth_backend play-auth {
    allow token test1;
}

stream auth-check {
    input fake://fake;
    on_play auth://play-auth;
}
```

Как срабатывает защита?

Различные проигрыватели и пользовательские устройства по-разному будут показывать, что защищенный видео поток более не доступен для проигрывания. Например ffmpeg покажет следующее:

```
$ ffmpeg  -i rtmp://localhost/rtmp/auth-check
ffmpeg version 6.1.1 Copyright (c) 2000-2023 the FFmpeg developers
...
  libswresample   4. 12.100 /  4. 12.100
  libpostproc    57.  3.100 / 57.  3.100
[in#0 @ 0x600001c9c700] Error opening input: Input/output error
Error opening input file rtmp://localhost/rtmp/auth-check.
Error opening input files: Input/output error
```

ffmpeg не раскрывает детали отказа, а по http отдается более стандартизованный код ошибки:

```
$ curl -v http://localhost:8080/auth-check/index.m3u8 -o /dev/null
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
    Current
                                 Dload  Upload   Total   Spent    Left
    Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--
      0*   Trying 127.0.0.1:8080...
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /auth-check/index.m3u8 HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/8.1.2
> Accept: */*
>
< HTTP/1.1 403 Forbidden
< access-control-allow-headers: x-vsaas-session, x-no-redirect, origin,
    authorization, accept, range, content-type, x-add-effective, session, x-
    originator, x-sid
< access-control-allow-methods: GET, PUT, DELETE, OPTIONS
< access-control-allow-origin: *
< access-control-expose-headers: Server, range, X-Run-Time, X-Sid, Content-
    Length, Location
< content-length: 1147
< date: Sat, 03 Feb 2024 19:16:16 GMT
< server: Streamer 24.02
< x-deny-reason: backend_denied
< x-route-time: 172
< x-run-time: 210
<
```

403 отдается, когда авторизация не пустила клиента

Как проиграть с токеном?

HTTP По HTTP протоколам: HLS, DASH, MSS, LL-HLS, WebRTC всё довольно просто, токен надо добавить в query string запроса.

```
$ curl http://localhost:8080/auth-check/index.m3u8?token=test1
#EXTM3U
#EXT-X-STREAM-INF:AVERAGE-BANDWIDTH=260000,BANDWIDTH=320000,RESOLUTION=320
    x240,FRAME-RATE=25.000,CODECS="avc1.42c015,mp4a.40.2",CLOSED-CAPTIONS=
    NONE
tracks-v1a1/mono.m3u8?token=test1
```

```
$ ffmpeg -i http://localhost:8080/auth-check/index.m3u8?token=test1
...
[hls @ 0x11f604b70] Opening 'http://localhost:8080/auth-check/tracks-v1a1/
    mono.m3u8?token=test1' for reading
[hls @ 0x11f604b70] Skip ('#EXT-X-VERSION:3')
[hls @ 0x11f604b70] Skip ('#EXT-X-PROGRAM-DATE-TIME:2024-02-03T19:31:31.839
    Z')
```

```
[hls @ 0x11f604b70] Opening 'http://localhost:8080/auth-check/tracks-v1a1
   /2024/02/03/19/31/36-05000.ts?token=test1' for reading
[hls @ 0x11f604b70] Opening 'http://localhost:8080/auth-check/tracks-v1a1
   /2024/02/03/19/31/41-05000.ts?token=test1' for reading
Input #0, hls, from 'http://localhost:8080/auth-check/index.m3u8?token=
   test1':
  Duration: N/A, start: 73249.691911, bitrate: N/A
  Program 0
    Metadata:
      variant_bitrate : 320000
  Stream #0:0: Video: h264 (Constrained Baseline) ([27][0][0][0] / 0x001B),
    yuv420p, 320x240 [SAR 1:1 DAR 4:3], 25 fps, 25 tbr, 90k tbn
    Metadata:
      variant_bitrate : 320000
  Stream #0:1(eng): Audio: aac (LC) ([15][0][0][0] / 0x000F), 48000 Hz,
   stereo, fltp
    Metadata:
      variant_bitrate : 320000
```

RTSP Протокол RTSP не сильно отличается, в нём тоже есть стандартный урл, стандартный способ передачи query string:

```
$ ffmpeg -i rtsp://localhost:1554/auth-check?token=test1
...
Input #0, rtsp, from 'rtsp://localhost:1554/auth-check?token=test1':
  Metadata:
    title           : Streamer 24.02
  Duration: N/A, start: 0.003000, bitrate: N/A
  Stream #0:0: Video: h264 (Constrained Baseline), yuv420p(progressive),
   320x240 [SAR 1:1 DAR 4:3], 25 fps, 25 tbr, 90k tbn
  Stream #0:1(eng): Audio: aac (LC), 48000 Hz, stereo, fltp
```

RTMP С RTMP немного сложнее, в этом протоколе нет единого понимания урла и поэтому есть куча разных вариантов, как передать авторизационную информацию. Однако существует сложившаяся практика отправлять query string в параметры функции play, склеивая с именем стрима и ffmpeg именно это и делает:

```
$ ffmpeg -i rtmp://localhost/static/auth-check?token=test1
...
Input #0, flv, from 'rtmp://localhost/static/auth-check?token=test1':
  Metadata:
    |RtmpSampleAccess: true
    audiochannels   : 2
  Duration: 00:00:00.00, start: 0.000000, bitrate: N/A
  Stream #0:0: Data: none
  Stream #0:1: Video: h264 (Constrained Baseline), yuv420p(progressive),
   320x240 [SAR 1:1 DAR 4:3], 25 fps, 25 tbr, 1k tbn
  Stream #0:2: Audio: aac (LC), 48000 Hz, stereo, fltp, 28 kb/s
```

SRT Есть отдельная статья про авторизацию проигрывания SRT.

UDP MPEG-TS Протокол UDP MPEG-TS (передача по мультикасту) не включает в себя взаимодействие клиента и сервера, сервер вообще не знает о том, что с него что-то проигрывают и поэтому такой метод авторизации не работает. Облегчает ситуацию то, что мультикаст по интернету не ходит и задачи ограничения доступа немного другие. Для этого используется шифрование и CAS системы.

Как управлять авторизационными токенами

Перечисление токенов в конфиге ни в коем случае не может являться основным механизмом. Мы неоднократно видели, как подобные забытые «секретные» ключи, сделанные для удобства отладки сервиса, потом расходятся по интернету и активно используются.

Основной механизм генерации и проверки — реализация на стороне вашего портала (миддлвари, веб-сайта) системы, которая сама генерирует токены и сама их проверяет.

Если у вас еще нет подписки на Flussonic или триальной лицензии, перейдите по ссылке и запросите бесплатный тестовый ключ. Установка сервера займет совсем немного времени, а тестовая конфигурация с которой вы сможете начать, приведена в начале этой статьи.

Так же, технические специалисты Flussonic с удовольствием помогут вам разобраться и найти недорогое и эффективное решение для вашей задачи по защите видео контента от несанкционированного просмотра.

5.4.2 Авторизация проигрывания по SRT

SRT часто применяется для раздачи телевизионного сигнала через интернет партнерам. Выставленный в интернет поток требует защиты проигрывания, включая задачу отзыва разрешения лишь одному клиенту.

В самом простом случае можно воспользоваться встроенным в SRT механизмом `passphrase`, но он не позволяет выдать разным клиентам разные данные, чтобы можно было одного отключить, остальных оставить. Поэтому многие альтернативные решения предлагают для каждого партнера выдать свой порт и на нём свой `passphrase`, а Flussonic Media Server позволяет использовать стандартный авторизационный токен и валидировать клиентов через авторизационный бекенд

Передача авторизационного токена в SRT

В отличие от других протоколов, SRT имеет сильно отличающийся механизм передачи метаданных на сервер, поэтому между клиентам реализация может разниться. Мы покажем пример для ffmpeg

Настройка тестового примера

Начнем с тестовой конфигурации флюссоника с искуственным источником и одним авторизационным бекендом, разрешающим заранее известный токен.

```
auth_backend srt_play {
    allow token test1;
}

stream srt-check {
    input fake://fake;
    on_play auth://srt_play;
    srt_play {
        port 10300;
    }
}
```

Как срабатывает защита?

```
ffmpeg -v debug -i "srt://127.0.0.1:10300"
...
[AVFormatContext @ 0x15a606eb0] Opening 'srt://127.0.0.1:10300' for reading
[srt @ 0x6000031ef300] No default whitelist set
16:44:12.996228/!W:SRT.cn: @512971494: processConnectResponse: rejecting
   per reception of a rejection HS response: ERROR:PEER
16:44:12.996505/!W:SRT.cn: @512971494: processAsyncConnectRequest: REJECT
   reported from HS processing: Peer rejected connection - not processing
   further
```

```
[srt @ 0x6000031ef300] Connection to srt://127.0.0.1:10300 failed: Input/
    output error
[in#0 @ 0x6000023ecb00] Error opening input: Input/output error
Error opening input file srt://127.0.0.1:10300.
Error opening input files: Input/output error
```

К сожалению, в SRT не предусмотрены коды и статусы ошибок, нет механизма сообщить клиенту, что поток в порядке, просто его не пускают. Т.е. демонстрируемое выше поведение клиента в случае с включенной защитой является не проблемой, а единственно возможным поведением.

Как передать токен?

```
ffmpeg -i 'srt://127.0.0.1:10300?streamid=#!::u=test1'
...
Input #0, mpegts, from 'srt://127.0.0.1:10300?streamid=#!::u=test1':
  Duration: N/A, start: 55414.651911, bitrate: N/A
  Program 1
  Stream #0:0[0xd3]: Video: h264 (Constrained Baseline) ([27][0][0][0] / 0
    x001B), yuv420p(progressive), 320x240 [SAR 1:1 DAR 4:3], 25 fps, 25 tbr,
     90k tbn
  Stream #0:1[0xdd](eng): Audio: aac (LC) ([15][0][0][0] / 0x000F), 48000
    Hz, stereo, fltp, 30 kb/s
```

В примере выше есть две тонкости: правильно сформировать `streamid` и правильно экранировать его, чтобы никакая система по пути не начала интерпретировать спец-символы типа # и !

Формат `#!::u=test1` является стандартом от Haivision, мы ему просто следуем и в поле `u` флюссоник как раз ищет стандартный авторизационный токен.

В этом примере `test1` — это как раз тот токен, который указан в тестовой конфигурации, как разрешенный. В вашем случае может быть что угодно другое.

5.4.3 Как ограничить доступ по IP-адресам

Иногда возникает необходимость ограничить доступ к каналам определенным IP адресам. Или наоборот, разрешать доступ к каналу только определенному списку IP. Эту задачу можно решить с помощью бэкенда, собранного с помощью конструктора бэкендов.

С помощью конфигуратора можно настроить очень гибкие схемы авторизации. На этой странице приведем примеры как заблокировать определенные IP-адреса, или наоборот, пустить мимо системы авторизации. Это может быть полезно для систем мониторинга.

Все правила, описанные далее, могут быть применены к потоку или глобально.

```
stream example_stream {
  input fake://fake;
  on_play auth://blacklist;
}
```

Где **blacklist** — имя одного из настроенных бэкендов. Конечно же, вы можете создать более одного авторизационного правила.

Заблокировать

Вся настройка происходит через **/etc/flussonic/flussonic.conf**.

```
auth_backend blacklist {
  deny ip 1.1.1.1;
  deny ip 2.2.2.2;
  deny ip 10.10/16;
  allow default;
}
```

Это правило запрещает просмотр с двух определенных адресов (1.1.1.1, 2.2.2.2) и целой подсети (10.10.0.0/16).

Строка **allow default;** означает, что по умолчанию нужно всем разрешать просмотр, кроме адресов, перечисленных в deny. Подробнее об опции

Разрешить

```
auth_backend whitelist {
  allow ip 192.168.0/24;
  allow ip 10.10/16;
  allow ip 8.8.8.8;
}
```

Это правило разрешает просмотр только из указанных сетей и конкретного IP-адреса. Остальные соединения будут блокироваться.

```
auth_backend multi {
  allow ip 192.168.0/24;
  backend http://examplehost/stalker_portal/server/api/
    chk_flussonic_tmp_link.php;
}
```

Это правило разрешает просмотр из локальной сети, а остальные обращения через IPTV Middleware.

5.4.4 Domain lock

Защита от вставки плеера на других сайтах работает с клиентами, передающими HTTP-заголовок Referer или другое поле, которое содержит адрес веб-страницы, с которой отправлен запрос на проигрывание. Как правило, это относится к флеш-плееру. К примеру, iOS устройства не передают Referer при обращении к видео серверу.

Включите эту защиту в конфигурационном файле следующим образом. Добавьте параметр `domain` и укажите домены, с которых разрешено проигрывание:

```
stream example {
  input fake://fake;
  domains myhost.com .myhost.com;
}
live user15 {
  domains myhost.com *.myhost.com;
}
```

Важно отметить, что это достаточно простая защита, спасающая лишь от простых схем встраивания. Если вам нужен более сложный сценарий, используйте CORS для защиты плеера.

5.4.5 Как настроить два авторизационных бекенда

Если вы уже используете IPTV Middleware, такое как IPTVPortal, Stalker или подобное, то вы можете столкнуться с ситуацией, когда надо добавить определенные исключения в авторизацию.

Например, передать партнеру ссылку на стрим с паролем, но при этом не хочется модифицировать код самой Middleware.

Здесь мы расскажем, как *Flussonic* поможет решить эту проблему путем добавления небольшого блока в конфигурационный файл.

Авторизационный бекенд с исключениями

Предположим, что авторизационный бекенд вашей Middleware — `http://iptv.myservice.com/auth.p`

Вы хотите, чтобы ваш партнер смог забрать стримы по паролю PASS, а также хотите разрешить доступ из локальной сети `192.168.1.0/24`.

Добавьте в файл `/etc/flussonic/flussonic.conf`:

```
auth_backend main {
  allow ip 192.168.1/24;
  allow token PASS;
  backend http://iptv.myservice.com/auth.php;
}
stream example {
  input fake://fake;
  on_play auth://main;
}
```

Здесь мы сначала проверяем IP-адрес и токен клиента. Если они не подходят, то идем к оригинальному бэкенду.

Чтобы использовать этот (`main`) авторизационный бэкенд для потока, укажите `auth://main`.

Эти настройки есть в пользовательском интерфейсе на вкладке **Auth** в настройках потока, которую вы можете открыть после того, как перешли в поток.

Аналогичным образом можно организовать два разных авторизационных бекенда.

Два авторизационных бекенда

```
auth_backend parallel {
  backend http://examplehost.iptvportal.ru/auth/flussonic/arescrypt/;
  backend http://stalker/stalker_portal/server/api/chk_flussonic_tmp_link.
    php;
}
stream cnn {
  input udp://239.255.0.1:1234;
```

Figure 5.27: Flussonic multi-authorization

```
  on_play auth://parallel;
}
```

В этом случае реализуется параллельная авторизация на нескольких HTTP-бэкендах.

Более подробно о конструкторе бэкендов.

5.4.6 CORS для защиты плеера

CORS (Cross-origin resource sharing) — это технология современных браузеров, которая позволяет ограничить доступ с одного домена к ресурсам другого домена. По сути, когда сервер A запрашивает какую-либо страницу с сервера B, ваш браузер проверяет, что ответ сервера B разрешает доступ для сервера A, о чем сигнализирует заголовок `Access-Control-Allow-Origin` в HTTP-ответе. Бывают и другие заголовки, которые вы можете выбрать в зависимости от рассматриваемого сценария.

Flussonic позволяет использовать CORS для того, чтобы разрешить доступ к плееру на странице embed.html только определенным доменам. Можно настроить CORS индивидуально для каждого потока или создать шаблон конфигурации потока.

> Не используйте CORS для авторизации. Для этого нужен авторизационный бэкенд.

Пример 1. Чтобы разрешить вставку плеера `embed.html` с потоком `protected` только на страницу `example.com`, в конфигурации потока нужно прописать директиву `playback_headers` и параметр `header` следующим образом:

```
stream protected {
  input fake://fake;
  playback_headers {
    header Access-Control-Allow-Origin example.com;
  }
}
```

Пример 2. Использование настроек CORS в шаблоне конфигурации потока:

```
template cors {
  playback_headers {
    protocols hls;
    playback live;
    header Access-Control-Allow-Origin example.com;
  }
}

stream fake {
  input fake://;
  template cors;
}
```

5.4.7 GeoIP

GeoIP2 — библиотека, позволяющая определять страну, в которой находится компьютер клиента, по его IP адресу. Она использует свободные базы данных геолокации GeoLite2 (Country, Country IPv6, City и ASN), которые хранятся в файлах `.mmbd`.

База данных содержит блоки IP в качестве ключей и названия стран (городов, ASN) в качестве значений. Эти данные более полные и точные, чем полученные при помощи реверсивного поиска в DNS.

> Flussonic Media Server поддерживает формат **GeoIP2**, а формат GeoIP больше не поддерживается.

Назначение библиотеки GeoIP2 Базы данных GeoIP2 позволяет в настройках потока разрешить просмотр видео только из определённых стран:

```
allowed_countries RU UA KZ;
```

Использование базы данных GeoIP2 в составе Flussonic

По умолчанию *Flussonic* использует библиотеку, которая поставляется вместе с *Flussonic* и содержит одну базу данных - `GeoLite2 Country`.

> В комплект поставки *Flussonic* входит снимок базы данных GeoIP2, доступный на момент публикации. Мы обновляем сведения по мере возможности. Если ваш сервис требует актуальные данные GeoIP2, приобретите базу данных у стороннего поставщика и подключите ее к *Flussonic*, как описано ниже.

Использование отдельной библиотеки GeoIP2

Обновления баз данных GeoIP2 могут выходить чаще, чем выходят релизы нашего сервера, поэтому иногда база, поставляемая с *Flussonic*, может устаревать.

Установите отдельно от *Flussonic* модуль GeoIP2 и настройте *Flussonic* на работу с ним. Так вы сможете:

- Получать самые свежие данные GeoIP2 без необходимости обновления сервера *Flussonic*.
- Использовать не только базу по странам, но и по городам или ASN.

Чтобы использовать отдельную библиотеку GeoIP2:

- Установите базы GeoIP2 (см. Инструкции по установке и обновлению баз данных).
- В настройках *Flussonic* пропишите путь до базы данных, которую собираетесь использовать. Для этого добавьте в конфигурационный файл директиву `geoip PATH_TO_DATABASE;`, в которой укажите путь до файла базы данных:

```
geoip tmp/geo/GeoLite2-City.mmdb;
```

Теперь *Flussonic* будет брать данные из указанной базы.

Если указанная база данных недоступна, будет использоваться поставляемая с *Flussonic*.

5.4.8 Ограничение количества сессий на пользователя (защита от кражи)

Ограничение сессий

Для того, чтобы любой Ваш пользователь, получивший доступ к потокам, не устроил полное зеркалирование на свой сервер (например, с целью дальнейшей перепродажи), во *Flussonic Media Server* есть возможность ограничить количество одновременно просматриваемых потоков.

Таким образом, даже получив доступ ко всем потокам, пользователь сможет просматривать одновременно лишь N потоков, а попытки зеркалировать все потоки ни к чему не приведут.

Ограничение накладывается на каждого пользователя, который имеет свой `UserId` и задаётся с помощью авторизации.

Детали

Для того чтобы ограничить количество сессий до 2, нужно в авторизационном бэкенде отдавать следующие заголовки:

```
X-UserId: some-user-id
X-Max-Sessions: 2
```

Если после такой авторизации пользователь попытается просматривать одновременно три потока, то просмотр одного из них будет прерван.

Бан

После того как сессия была забанена, любые попытки переоткрыть её в течении `X-AuthDuration` будут отклонены *Flussonic Media Server*.

Таким образом, если указать `X-AuthDuration: 3600` и открыть лишний поток, то после блокировки этот поток нельзя будет открыть, используя старый токен в течении часа.

После того как сессия будет заблокирована на очередной запрос HLS-плейлиста клиент получит ответ `403 Forbidden`. В случае же с RTSP, RTMP, HTTP MPEG-TS сокет будет просто молча закрыт.

Каждая заблокированная сессия сопровождается записью в лог, например:

```
14:58:51.598 <0.391.0> [stream-name] session_limiter:174 Ban session_id:
   <<"604551981e3e787b897afbaf35bb9f4d168d70b9">> for user_id:
   <<"8471796306">> and token: <<"5cfb82ecaf56ebfe7ac32a9020c86ef1d231d49e
   ">> due to exceeded session limit
```

Мягкое ограничивание

Некоторые Middleware не могут генерировать новый токен на каждый запрос HLS-потока. При переключении между потоками это может создать проблемы, так как сессии на старых потоках будут помечены как лишние и будут заблокированы.

Специально для этих случаев во *Flussonic Media Server* есть механизм мягкого ограничения сессий.

Иногда блокировка не происходит в первую проверку, необходимо время, чтобы понять, что все сессии действительно используются. Тогда она происходит во вторую или третью волны. Таким образом, после появления лишних сессий, они обычно блокируются через 30-90 секунд.

Для того, чтобы включить этот режим, необходимо указать параметр `soft_limitation=true` к опции `on_play`, например:

```
stream foobar {
  on_play http://FLUSSONIC-IP:8081/my_auth_script.php soft_limitation=true;
}
```

Опцию `soft_limitation` также можно установить в пользовательском интерфейсе на странице **Conifg** > **Auth**:

X-Unique: true

Заголовок `X-Unique` считается устаревшим, вместо него предполагается использовать `X-Max-Sessions` описанный выше.

```
X-UserId: some-id
X-Unique: true
```

Эквивалентно:

```
X-UserId: some-id
X-Max-Sessions: 1
```

Кроме того, если указаны и `X-Max-Sessions` и `X-Unique`, то приоритет отдаётся `X-Max-Sessions`. Таким образом:

```
X-UserId: some-id
X-Max-Sessions: 5
X-Unique: true
```

Эквивалентно:

```
X-UserId: some-id
X-Max-Sessions: 5
```

Figure 5.28: Soft limitation

5.4.9 Защита доступа к потокам без бекенда

В этой статье приведен пример того, как можно реализовать систему авторизации без написания собственного бэкенда.

Схема работы авторизации: 1. Ваш сайт генерирует токен по несложной формуле и хеширует его с помощью секретного ключа. 2. Клиент открывает поток с полученным токеном. 3. *Flussonic* генерирует токен по той же формуле, используя тот же ключ (и дополнительно используя имя потока и (необязательно) IP-адрес клиента). 4. Хеши совпали — доступ разрешен, не совпали — запрещен.

Настройка Flussonic для использования авторизации по токену

В поставке *Flussonic* есть вся необходимая логика для проверки генерируемых токенов. Достаточно просто указать опцию `securetoken` и пароль для авторизации:

```
stream example-stream {
  input fake://fake;
  on_play securetoken://SECRETKEY;
}
```

Если вы хотите исключить из проверки IP-адрес клиента, добавьте опцию `no_check_ip=true` в конфигурацию:

```
stream example-stream {
  input fake://fake;
  on_play securetoken://SECRETKEY?no_check_ip=true;
}
```

Можно включить авторизацию (директиву **auth**) как для одного потока, так и глобально.

Код для сайта

Чтобы сгенерировать токен, *Flussonic* должен знать следующее:

- (необязательно) IP-адрес клиента,
- имя потока,
- секретный ключ,
- текущее время,
- (необязательно) `user_id`, уникальный идентификатор пользователя, который используется для ограничения количества сессий (см. Ограничение количества сессий на пользователя).

Код на сайте должен собрать эти данные в одну строку:

`string = streamname + ip + starttime + endtime + secretkey + salt + user_id`

И получить токен по формуле:

`sha1(string) + salt + endtime + starttime`,

где:

- `ip` — IP адрес клиентского устройства.
- `streamname` — название потока.

- `starttime` — это текущее время UTC (в Unix Timestamp).
- `endtime` — время окончания жизни токена (обычно это текущее время + несколько часов). По прошествии этого времени токен перестанет работать и его надо будет запрашивать заново.
- `secretkey` — это ключ, указанный в файле `/etc/flussonic/flussonic.conf`.
- `salt` — строка из случайных символов. Необходима, чтобы для одинаковых входных данных генерировались разные токены.
- `user_id` — строка, получаемая от системы биллинга/middleware (необязательно).

Если клиентские устройства находятся за прокси или их IP могут часто меняться, вы можете исключить IP-адрес клиента при формировании токена.

PHP пример

```
<?php
$flussonic = $_GET['host']; // This script gets Flussonic address from a
    query. String 'http://flussonic-ip'
$key = 'SECRETKEY'; // The key from flussonic.conf file. KEEP IT IN SECRET.
$lifetime = 3600 * 3; // The link will become invalid in 3 hours.

$stream = $_GET['stream']; // This script gets the stream name from a query
    . string (script.php?stream=bbc)

$ipaddr = $_SERVER['REMOTE_ADDR']; // (v20.07) Set $ipaddr = 'no_check_ip'
    if you want to exclude IP address of client devices from checking.
$desync = 300; // Allowed time desync between Flussonic and hosting servers
     in seconds.
$starttime = time() - $desync;
$endtime = $starttime + $lifetime;
$salt = bin2hex(openssl_random_pseudo_bytes(16));

$hashsrt = $stream.$ipaddr.$starttime.$endtime.$key.$salt;
$hash = sha1($hashsrt);

$token = $hash.'-'.$salt.'-'.$endtime.'-'.$starttime;
$link = $flussonic.'/'.$stream.'/embed.html?token='.$token.'&remote='.
    $ipaddr;
$embed = '<iframe allowfullscreen style="width:640px; height:480px;" src
    ="'.$link.'"></iframe>';

echo $embed;
?>
```

Rails пример

config/routes.rb:

```
Rails.application.routes.draw do
 ...
  get '/securetoken/:id', to: 'securetoken#index'
end
```

app/controllers/securetoken_controller.rb:

```
class SecuretokenController < ApplicationController

  def index

    flussonic = 'http://flussonic-ip'
    secret = 'SECRETKEY'

    streamname = params[:id]
    lifetime = 3600 * 3
    starttime = Time.now.to_i - 300
    endtime = Time.now.to_i + lifetime
    salt = rand(8**8).to_s(8)

    hash = Digest::SHA1.hexdigest(streamname + request.remote_ip +
   starttime.to_s + endtime.to_s + secret + salt)
    token = hash + '-' + salt + '-' + endtime.to_s + '-' + starttime.to_s
    @url = flussonic + '/' + streamname + '/' + 'embed.html?token=' + token
  end
end
```

app/views/securetoken/index.html.erb:

```
<iframe allowfullscreen style="width:640px; height:480px;" src="<%= @url
   %>"></iframe>
```

5.4.10 Axinom DRM

Flussonic использует протокол SPEKE для обмена ключами с Axinom DRM, используя XML-формат CPIX. Для работы с Axinom DRM дополнительно требуется специальный заголовок авторизации, включающий пару `Tenant ID` и `Management Key`. Вы можете настроить Axinom DRM для потока или файлов в VOD-локации в пользовательском интерфейсе или в файле конфигурации:

В интерфейсе

Перейдите на вкладку **Auth** в профиле потока или VOD-локации и выберите Axinom в списке **Require DRM authorization**.

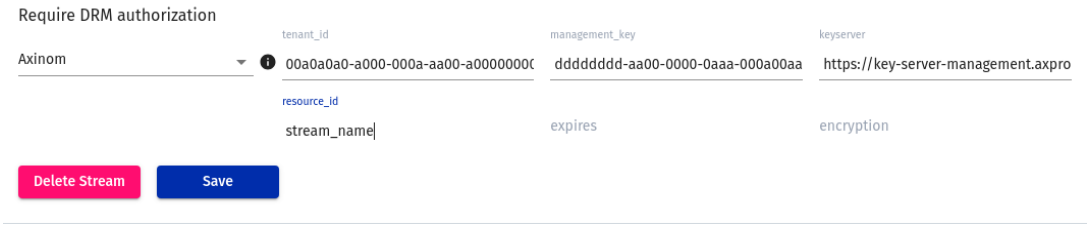


Figure 5.29: Axinom drm

В конфигурационном файле

Добавьте параметр `drm` в конфигурацию потока или VOD следующим образом:

```
stream example_stream {
  input udp://239.0.0.1:1234;
  protocols dash hls;
  drm axinom keyserver=http://example.com management_key=dddddddd-aa00
   -0000-0aaa-000a00aaaa0a resource_id=stream_name tenant_id=00a0a0a0-a000
   -000a-aa00-a00000000000;
}
```

Используемые параметры

Чтобы получить значения параметров, используемых для настройки:

- Войдите в свою учетную запись на портале Axinom.
- Перейдите в **My Mosaic**, а затем на вкладку **DRM**.
- Вам понадобятся следующие параметры из раздела **Key Service config**:

Параметры, значение которых вы задаете произвольно:

- `resource_id` — идентификатор потока. Он будет использован как `content_id` в Axinom. Вы можете оставить поле пустым, и тогда Flussonic сделает `content_id` равным имени потока или файла, но в этом случае `content_id` изменится при переименовании потока или файла, а вместе с ним изменится и `keyID` (см. ниже).

Полный список настроек для Axinom DRM вы можете найти в 7Cbody%7Cdrmtag/stream/operation/stream-save%7Cbody%7CdrmFlussonic API reference, выбрав `Axinom` в раскрывающемся списке поставщиков (`vendor`).

Получение keyID

Лицензия Axinom привязывается к `keyID`. Flussonic генерирует этот параметр автоматически на основании `resource_id` во время обмена ключами с Axinom DRM. Чтобы узнать, какой `keyID` Flussonic сгенерировал для потока, получите DASH-манифест, используя ссылку **DASH** с вкладки **Output** в профиле потока. Например:

```
curl -s https://FLUSSONIC-IP:PORT/example_stream/index.mpd | grep
   default_KID
```

Параметр `default_KID` — это и есть ваш `keyID`.

Проверка проигрывания потока

Проверить проигрывание потока можно в тестовом плеере на сайте Axinom. При получении ссылки для проигрывания потребуется в секции `Content Keys` ввести `keyID`, полученный описанным выше способом.

5.4.11 BuyDRM KeyOS

Figure 5.30: BuyDRM

BuyDRM's KeyOS platform это один из пройвадеров DRM, с помощью которого можно получать ключи для DASH, MSS,HLS.

Configuration

Flussonic поддерживает протокол CPIXv2 для получения ключей шифрования от BuyDRM. Настройка DRM для потока или VOD-локации выполняется следующим образом:

```
stream example_stream {
  input udp://239.0.0.1:1234;
  protocols dash hls;
  drm keyos end_user_cert=/etc/flussonic/buydrm/user_public_cert.pem
    end_user_private_key=/etc/flussonic/buydrm/user_private_key.pem;
}
storage drm {
  path /storage/vod;
  protocols dash hls;
  drm keyos end_user_cert=/etc/flussonic/buydrm/user_public_cert.pem
    end_user_private_key=/etc/flussonic/buydrm/user_private_key.pem;
}
```

- `end_user_cert.pem` — публичная часть сертификата пользователя BuyDRM x505.
- `user_private_key.pem` — закрытая часть сертификата пользователя BuyDRM x509.

Свяжитесь с поддержкой BuyDRM, чтобы получить файлы `end_user_cert.pem` и `user_private_key.pem`. Эти файлы необходимо разместить на сервере в каталоге `/etc/flussonic/buydrm`.

По умолчанию Flussonic передает название потока в атрибуте `CPIX@ContentId` в CPIX-запросе. Чтобы изменить значение `CPIX@ContentId` вручную, используйте параметр `resource_id`. Пример конфигурации:

```
stream example_stream {
  input udp://239.0.0.1:1234;
  protocols dash hls;
```

```
  drm keyos end_user_cert=/etc/flussonic/buydrm/user_public_cert.pem
    end_user_private_key=/etc/flussonic/buydrm/user_private_key.pem
    resource_id=MY_UNIQUE_CONTENT_ID;
}
```

Для VOD файла `/storage/vod/content.mp4` Flussonic будет читать `ContentId` из файла /storage/content.mp4.keyos_id.

Актуальный список настроек для BuyDRM вы можете найти в 7Cbody%7Cdrmtag/stream/operation/stream-save%7Cbody%7CdrmFlussonic API reference, выбрав `buydrm` в выпадающем списке поставщиков (`vendor`).

5.4.12 drmnow! DRM

Flussonic интегрирован с системой **drmnow! multi-drm**, которая работает с основными поставщиками лицензий (Apple, Google, Microsoft и Huawei) и поддерживает несколько DRM-систем (FairPlay, Widevine, PlayReady, WisePlay). Основной плюс drmnow! multi-drm заключается в возможности работы на разнообразных устройствах и браузерах. Более подробную информацию можно найти здесь.

Интеграция Flussonic с drmnow! multi-drm происходит по протоколу ***speke (cpix)***. Чтобы настроить поток для работы с этой системой, добавьте параметр `drm cpix` следующим образом:

```
stream test {
  input udp://239.0.0.1:1234;
  resource_id MYSTREAM;
  protocols dash hls mss;
  drm cpix keyserver=https://[project].nowdrm.co/drm/speke?token=[token];
}
```

Здесь:

- project (обязательный параметр) — проект из личного кабинета cdnnow!
- token (обязательный параметр) — ключ доступа из личного кабинета cdnnow!

Актуальный список настроек для drmnow! DRM вы можете найти в 7Cbody%7Cdrmtag/stream/operation/stream-save%7Cbody%7CdrmFlussonic API reference, выбрав `cpix` в выпадающем списке поставщиков (`vendor`).

Чтобы проиграть видео, нужно использовать следующие серверы лицензий:

- сервер лицензий Widevine: https://[project].nowdrm.co/widevine
- сервер лицензий Playready: https://[project].nowdrm.co/playready
- сервер лицензий Fairplay: https://[project].nowdrm.co/fairplay
- сервер лицензий Wiseplay: https://[project].nowdrm.co/wiseplay
- файл `*.der` для Fairplay: https://playrnow.pro/cdnnow.der

5.4.13 DRMtoday DRM

DRMtoday DRM — это реализация CPIX с особым процессом конфигурации и авторизации. Вы можете настроить DRMtoday DRM для потока или файлов в VOD-локации.

Добавьте параметр `drm` в конфигурацию потока или VOD следующим образом:

```
stream example_stream {
  input udp://239.0.0.1:1234;
  protocols dash hls;
  drm drmtoday auth_server=https://example.com cpix_config_id=UUID
   keyserver=https://keyserver1.mycompany.com merchant_id=mycompany
   password=api_pass resource_id=L2sItm6 username=api_user;
}
```

Здесь:

- `auth_server` — URL сервера авторизации: в DRMtoday используются отдельные серверы для авторизации и запросов ключей.
- `keyserver` — URL сервера ключей DRMtoday.
- `cpix_config_id` — UUID конфигурации CPIX, которую вы создали в DRMtoday.
- `merchant_id` — merchant UUID, назначенный вам в DRMtoday.
- `username` и `password` — учетные данные специального API-пользователя.

Полный список настроек для DRMtoday DRM вы можете найти в 7Cbody%7Cdrmtag/stream/operation/stream-save%7Cbody%7CdrmFlussonic API reference, выбрав DRM`today` в раскрывающемся списке поставщиков (`vendor`).

5.4.14 EzDRM

Flussonic поддерживает два варианта обмена ключами в системе EzDRM:

- EzDRM с использованием CPIX протокола для передачи ключей
- EzDRM с использованием собственного формата EzDRM для передачи ключей.

Настройка EzDRM (обмен ключами по CPIX)

Чтобы настроить EzDRM с обменом ключами в формате CPIX для потока или файлов, находящихся в определенной VOD-локации, добавьте параметр `drm ezdrm`:

```
stream example_stream {
  input udp://239.0.0.1:1234;
  protocols dash hls;
  drm ezdrm password="password" user="user@ezdrm.com";
}
file drm {
  path /storage/vod;
  protocols dash hls;
  drm ezdrm password="password" user="user@ezdrm.com";
}
```

, где:

- `user` и `password` — обязательные параметры. Их можно получить через ваш аккаунт на https://www.ezdrm.com.

Для VOD файла `/storage/vod/content.mp4` Flussonic будет читать `ContentId` из файла `/storage/content.mp4.ezdrm_id`.

Если указать дополнительно опцию `resource_id` — DRM ID будет использоваться в запросах потока вместо его имени.

```
stream example_stream {
  input udp://239.0.0.1:1234;
  protocols dash hls;
  drm ezdrm password="password" user="user@ezdrm.com" resource_id=12345
    asdfg12345asdfg12345asd;
}
```

Актуальный список настроек для EzDRM с обменом ключами по CPIX вы можете найти в 7Cbody%7Cdrmtag/stre save%7Cbody%7CdrmFlussonic API reference, выбрав `ezdrm` в выпадающем списке поставщиков (`vendor`).

Настройка EzDRM (обмен ключами в формате EzDRM)

Чтобы настроить EzDRM с обменом ключами в собственном формате для потока или файлов, находящихся в определенной VOD-локации, добавьте параметр `drm ezdrm_classic`:

```
stream example_stream {
  input udp://239.0.0.1:1234;
  protocols dash hls;
  drm ezdrm_classic password="password" user="user@ezdrm.com";
}
```

, где:

- `user` и `password` — обязательные параметры. Их можно получить через ваш аккаунт на https://www.ezdrm.com.

Если указать дополнительно опцию `resource_id` — DRM ID будет использоваться в запросах потока вместо его имени.

```
stream example_stream {
  input udp://239.0.0.1:1234;
  protocols dash hls;
  drm ezdrm_classic password="password" user="user@ezdrm.com" resource_id
   ="12345asdfg12345asdfg12345asd";
}
```

Актуальный список настроек для EzDRM с обменом ключами в собственном формате вы можете найти в 7Cbody%7Cdrmtag/stream/operation/stream-save%7Cbody%7CdrmFlussonic API reference, выбрав `ezdrm_classic` в выпадающем списке поставщиков (`vendor`).

5.4.15 PlayReady DRM

Flussonic поддерживает шифрование с помощью PlayReady DRM для протокола Microsoft Smooth Streaming.

Чтобы настроить PlayReady DRM для потока или файлов, находящихся в определенной VOD-локации, добавьте параметр `drm`:

```
stream example_stream {
  input fake://fake;
  drm playready keyseed=KEYSEED;
}
vod vod_files {
  path priv;
  drm playready keyseed=KEYSEED;
}
```

, где:

- `keyseed` (обязательный параметр) — произвольная строка в кодировке Base64 размером 30 байтов. PlayReady использует `keyseed` для того, чтобы создать ключ шифрования.

Актуальный список настроек для PlayReady DRM вы можете найти в 7Cbody%7Cdrmtag/stream/operation/stream save%7Cbody%7CdrmFlussonic API reference, выбрав `playready` в выпадающем списке поставщиков (`vendor`).

Вы можете сгенерировать `keyseed`, выполнив команду в Linux:

```
dd if=/dev/urandom bs=1 count=30 | base64
```

Для тестирования видеопотока на тестовом сервере PlayReady, следует использовать значение `keyseed=test`.

5.4.16 Сервер ключей

Пример самого простого сервера ключей для AES-128 или SAMPLE-AES шифрования ниже:

`cas-server.php`:

```
<?php
  header("HTTP/1.0 200 OK");
  $resource = $_GET["file"];
  $number = $_GET["number"];
  error_log("Server is requesting key ".$number." for ".$resource." from ".
    $_SERVER["REMOTE_ADDR"]);
  header("X-Key-Url: http://".$_SERVER["HTTP_HOST"]."/user-key.php?name=".
    $resource."&number=".$number);

  $input = $resource.".".$number;
  $key = hash('md4',$input);
  header("Content-Length: ".strlen($key));
  echo $key;
?>
```

`user-key.php`:

```
<?php
  header("HTTP/1.0 200 OK");
  $resource = $_GET["name"];
  $number = $_GET["number"];

  $input = $resource.".".$number;
  $key = hex2bin(hash('md4',$input));
  header("Content-Length: ".strlen($key));
  header("Content-Type: application/octet-stream");
  error_log("User is requesting key ".$number." for ".$resource." from ".
    $_SERVER["REMOTE_ADDR"]);
  echo $key;
?>
```

Разместите эти два файла в директории веб-сервера. `cas-server.php` должен быть доступен для Flussonic, `user-key.php` должен быть доступен для клиентов.

Настройка DRM для потока осуществляется следующим образом:

```
stream tvchannel {
  input udp://239.0.0.1:1234;
  protocols dash hls;
  drm aes128 keyserver=http://192.168.0.80:4500/cas-server.php;
}
```

, где http://192.168.0.80:4500/cas-server.php - это URL PHP скрипта, расположенного на вашем сервере.

Актуальный список настроек для сервера ключей для AES-128 или SAMPLE-AES вы можете

найти в 7Cbody%7Cdrmtag/stream/operation/stream-save%7Cbody%7CdrmFlussonic API reference, выбрав `aes128` или `sample_aes` соответственно в выпадающем списке поставщиков (`vendor`).

Flussonic сам ротирует ключи для потоков раз в 10 минут.

5.4.17 Widevine DRM

Чтобы настроить Widevine DRM для потока или файлов, находящихся в определенной VOD-локации, добавьте параметр `drm`:

```
stream example_stream {
  input udp://239.0.0.1:1234;
  protocols dash hls;
  drm widevine aes_key=1234512345...45123451234 iv=12345as...45asdfg12
   signer=widevine_test;
}
```

, где:

- `aes_key` и `iv` — обязательные параметры. Их можно получить через ваш аккаунт на https://www.widevine.com/.
- `content_id` — необязательный параметр. Задает уникальный идентификатор содержимого. По умолчанию `content_id` равен названию потока.

При использовании тестового ключа необходимо указать опцию `signer`:

- `signer` — ваш уникальный ID, который Flussonic будет использовать для соединения с сервером ключей `http://license.uat.widevine.com/cenc/getcontentkey/<signer>`.

В случае использования production-ключа необходимо указать опцию `keyserver`:

- `keyserver` — задает кастомный URL для сервера ключей Widevine. Например, `keyserver=http://l`
  В этом случае параметр `signer` — необязательный, поскольку `signer` берется из URL, заданного в параметре `keyserver`.

Актуальный список настроек для Widevine DRM вы можете найти в 7Cbody%7Cdrmtag/stream/operation/stream-save%7Cbody%7CdrmFlussonic API reference, выбрав `widevine` в выпадающем списке поставщиков (`vendor`).

Chapter 6

Internet Streaming

6.1 Ingest

6.1.1 Публикация из OBS Studio в Flussonic Media Server

Публикация из OBS Studio в Flussonic Media Server

С помощью программы Open Broadcaster Software (OBS) можно публиковать поток со своего компьютера в Flussonic Media Server. Это может использоваться для стриминга игр, вебинаров и любых других трансляций с компьютера в интернет.

Например, можно стримить в социальные сети.

Содержание:

Публикация в статический поток в Flussonic Media Server

Настройка через конфигурационный файл В *Flussonic Media Server* достаточно создать поток без источников и указать, что вы разрешаете в него публикацию.

В конфигурационном файле `/etc/flussonic/flussonic.conf` для потока пропишите:

```
stream published {
  input publish://;
}
```

Для применения настроек не забудьте выполнить команду:

```
service flussonic reload
```

Подробнее в статье Публикация видео на сервер в статический поток.

Трансляция статического потока из OBS Studio

Скачайте и установите OBS Studio. Откройте программу и перейдите в настройки.

Откройте меню **Вещание**:

- **Тип вещания**: Пользовательский сервер вещания

- **URL**: `rtmp://flussonic-ip/published`
- **Ключ потока**: оставить пустым

Где `published` — имя вашего потока.

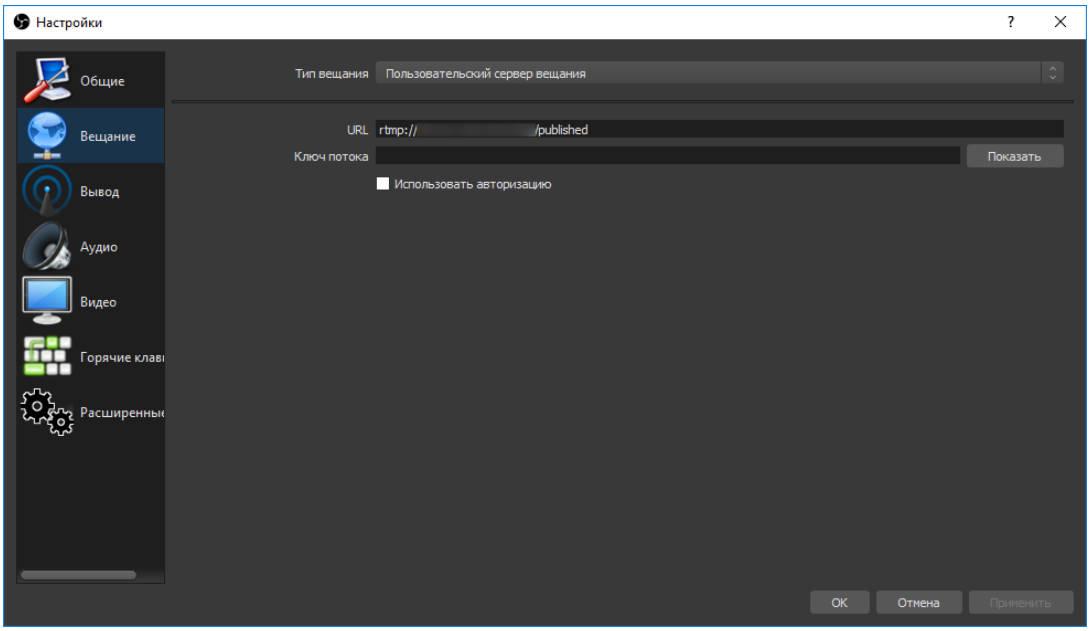


Figure 6.1: OBS Studio

Нажмите **OK** для сохранения.

Публикация по динамическому имени в Flussonic Media Server

Если вы заранее не знаете под каким именем будет публиковаться поток или этих потоков может быть очень много, то вы можете указать *префикс публикации*.

См. подробнее: Публикация по динамическому имени.

Настройка через конфигурационный файл: В конфигурационном файле `/etc/flussonic/flussonic.co` для потока пропишите:

```
stream published {
  input publish://;
}
```

Для применения настроек не забудьте выполнить команду:

```
service flussonic reload
```

Подробнее в статье Публикация по динамическому имени.

Трансляция динамического потока из OBS Studio

Скачайте и установите OBS Studio. Откройте программу и перейдите в настройки.

Откройте меню **Вещание**:

- **Тип вещания**: Пользовательский сервер вещания
- **URL**: `rtmp://flussonic-ip/chats/tempname`
- **Ключ потока**: оставить пустым

Где `chats` — имя префикса. Что именно идет после `chats` — это дело клиента, но Flussonic Media Server заранее не знает, какое именно это будет имя.

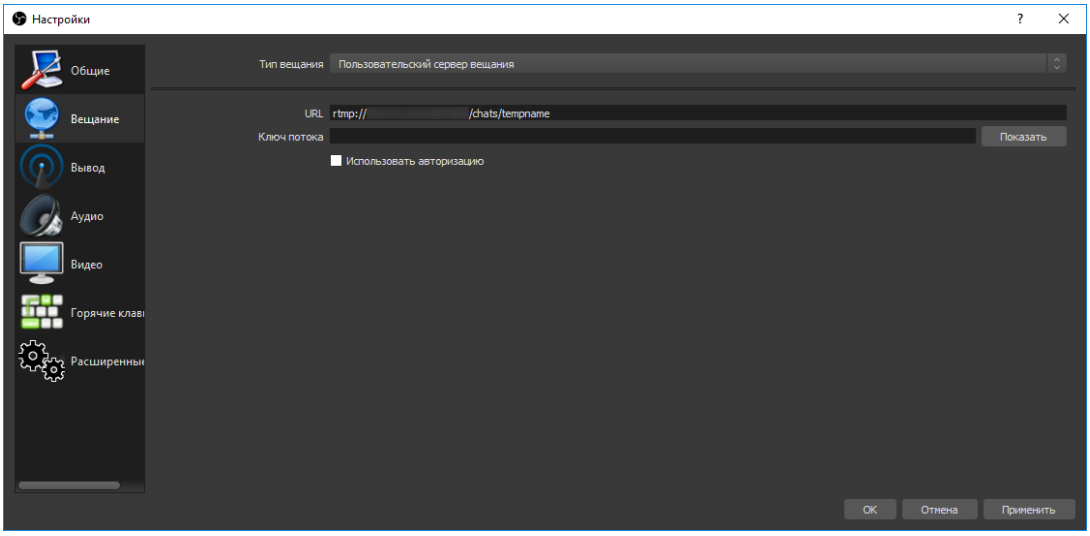


Figure 6.2: Media Server

Нажмите **OK** для сохранения.

В главном окне OBS Studio нажмите **Запустить трансляцию**.

Трансляция уже началась и вы можете ее наблюдать в административном интерфейсе Flussonic Media Server. Пока это просто чёрный экран. Остановим трансляцию и настроим OBS Server.

Настройка OBS Studio

Откройте главное окно OBS Studio и создайте сцены. К примеру, «Заглушка», «Прямой эфир», «Перерыв», «Конец».

Все сцены и источники в OBS Studio общие и не могут иметь одинаковые названия. Это означает, что если вы назвали источник «Прямая трансляция», то не можете назвать так же сцену.

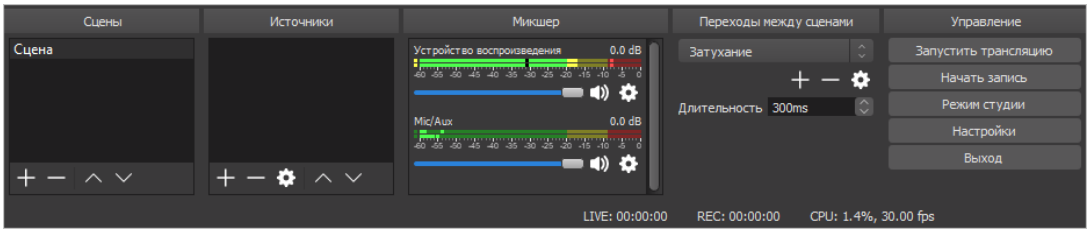


Figure 6.3

В каждую сцену вы можете добавить различные источники трансляций. Источниками могут быть как целый экран, так и отдельное открытое окно. Например, запущенное приложение, браузер или даже отдельная вкладка браузера. Также, вы можете в любое место экрана вывести текст, источник медиа или поток с веб-камеры.

Вы можете перестраивать порядок источников при предпросмотре, перетаскивая их по списку или используя кнопки со стрелками вверх и вниз. Источник, располагающийся выше другого в списке, будет более приоритетным и может скрывать находящиеся ниже него. Вы можете включать и отключать источники во время трансляции.

Пример настройки захвата экрана:

Когда источник выбран в списке, вы видите красную рамку вокруг него. Это ограничивающая рамка, которая может использоваться для перемещения источников при предпросмотре, а также чтобы увеличивать или уменьшать их.

Горячие клавиши, которые доступны при предпросмотре для изменения позиции и размера источника:

- Нажмите «Ctrl», чтобы отключить привязку источника/границы.
- Нажмите «Alt» и перетащите ограничивающую рамку для обрезки.
- «Ctrl+Alt» — подогнать по размеру экрана.
- «Ctrl+S» — растянуть на весь экран.
- «Ctrl+D» — разместить по центру экрана.
- «Ctrl+R» — сброс размера/положения источника.

В меню **Микшер** регулируется громкость подключенных аудиоканалов.

В меню **Переходы между сценами** можно выбрать, как будет происходить переключение между сценами: затуханием или обрезанием (резкое переключение).

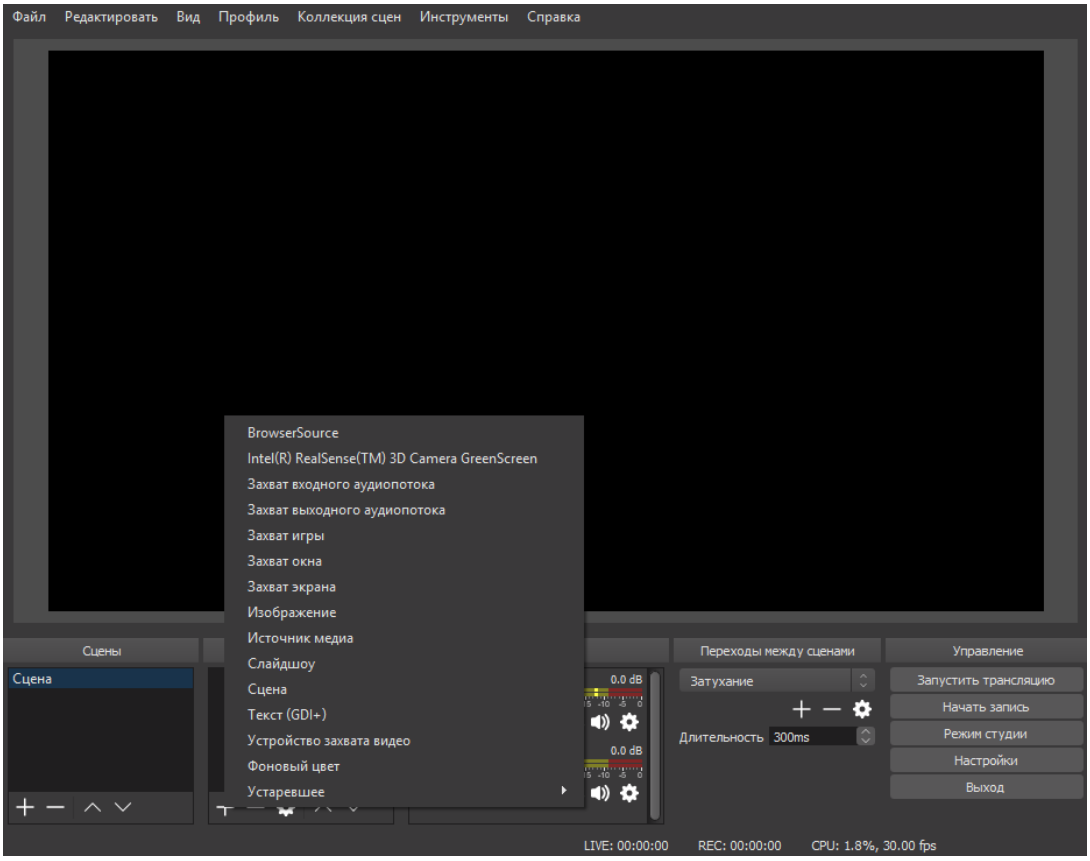


Figure 6.4

6.1.2 Захват SDI на Decklink

Flussonic Media Server работает с платами Blackmagic: Decklink SDI или Decklink HDMI, а также Decklink Quad 2, у которых 8 портов.

Вы сможете:

* Захватывать видео с плат захвата Blackmagic Decklink SDI или Decklink HDMI

- Передавать видео на Blackmagic Decklink SDI или HDMI
- Читать телетекст из полученного с Decklink потока.
- Читать метки врезки рекламы из полученного с Decklink потока. Метки врезки рекламы преобразуются из VBI SCTE-104 в формат SCTE-35, пригодный для отправки в MPEG-TS и HLS
- Передавать телетекст (Teletext B) из MPEG-TS в видео, передаваемое на SDI карты Decklink.

Содержание

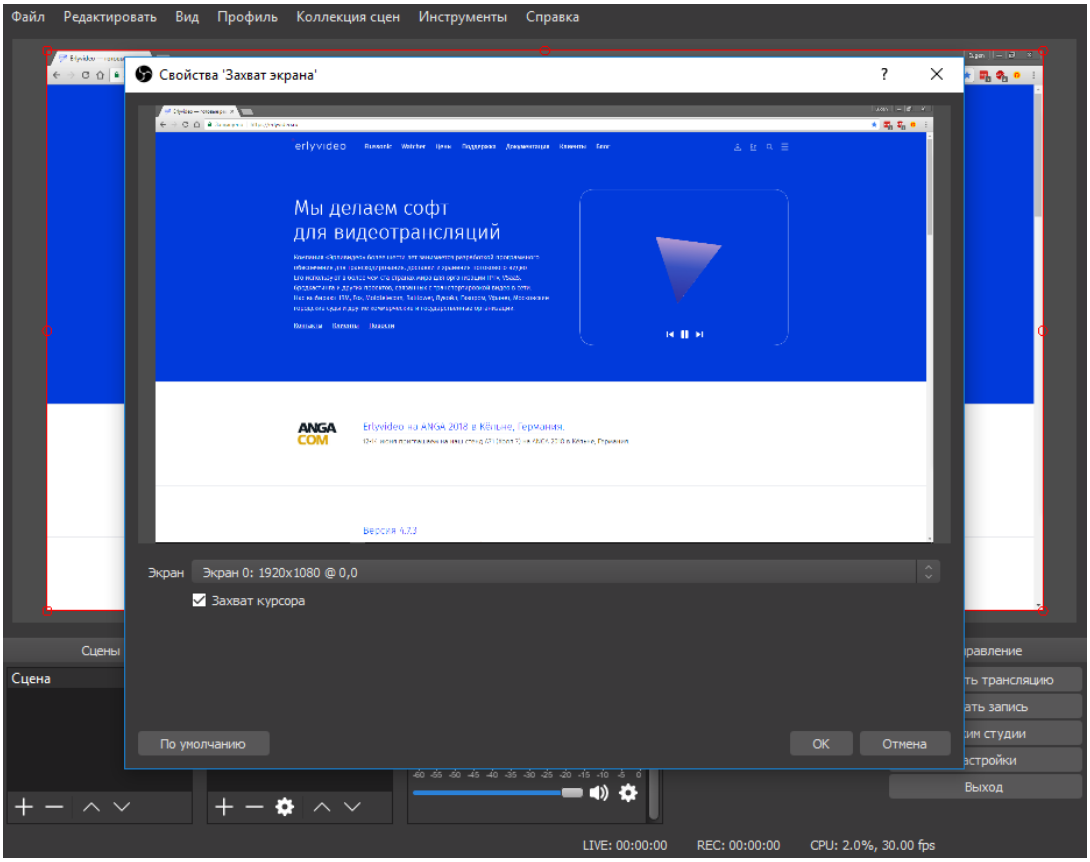


Figure 6.5

- Установка драйвера для Decklink Blackmagic
- Захват видео с платы захвата Decklink SDI или HDMI
- Транскодирование видео с плат Decklink
- Устранение чересстрочности в прогрессивных потоках
- Захват SD видео — указание SAR
- Дуплексный режим работы DeckLink
- Работа карт DeckLink в режиме 24x7

Также полезно:

- Чтение телетекста и субтитров из VBI и передача в MPTS/SPTS

Установка драйвера для платы захвата

- Вы можете скачать Linux версию драйвера с официального сайта Blackmagic по ссылке

- Установите пакет:

```
cd Blackmagic_Desktop_Video_Linux_12.1/deb/x86_64
dpkg -i desktopvideo_12.1a9_amd64.deb
```

Если у вас установлена другая версия, вы можете удалить ее командой:

```
dpkg -r desktopvideo
```

Захват видео с платы захвата Decklink SDI или HDMI

Выполните следующую команду, чтобы проверить, успешно ли завершилась установка драйвера:

```
BlackmagicFirmwareUpdater status
```

Если установка прошла успешно, то Вы должны увидеть список доступных устройств для захвата.

После установки драйвера Blackmagic и обновления прошивки, настройте поток таким образом:

```
stream sdi {
  input decklink://0;
}
```

Flussonic Media Server подключится к указанному первому устройству (0) и запустит на нем автоконфигурацию для поиска активного разрешения.

Некоторые модели Decklink не поддерживают автоматический поиск активного разрешения и для них необходимо указывать режим вручную с помощью опций `mode` и `vinput`. Например, для Intensity Pro с подключенным к нему по HDMI источником 720p и 50 fps поток нужно настроить так:

```
stream sdi {
  input decklink://0 mode=hp50 vinput=hdmi;
}
```

Вы также можете настроить параметры захвата видео с Decklink SDI во *Flussonic UI*:

- Перейдите на вкладку **Streams** на странице **Media** в боковом меню. Затем откройте настройки потока, настроенного на захват видео с Decklink SDI (с источником `decklink://0`), кликнув по имени потока. Если у вас нет такого потока, создайте его.
- Перейдите на вкладку **Input** и нажмите **Options**.
- Задайте нужные значения параметров в разделе **Decklink**:

Figure 6.6: Decklink UI

Транскодирование видео с плат SDI

Если необходимо транскодировать захваченный поток из Decklink SDI, добавьте директиву `transcoder` в настройки потока:

```
stream sdi {
  input decklink://0;
  transcoder vb=3096k ab=64k;
}
```

Параметр транскодирования `external=false` теперь используется по умолчанию для SDI, HDMI и других "сырых" видеопотоков, предотвращая чрезмерную нагрузку на сервер, возникающую при `external=true`. При транскодировании нескольких потоков на Nvidia NVENC убедитесь, что в опции выставлено одинаковое значение на всех потоках.

Параметры транскодирования больше не указываются отдельно для источника `input decklink://`, так как теперь Flussonic умеет получать видео из SDI в виде сырых кадров. Ранее необходимо было сразу транскодировать SDI-поток опцией `enc=` в настройках захвата, потому что Flus-

sonic не мог работать с ним в виде некодированного видео.

Что это даёт:

- Повышение качества видео. Мы избегаем двойного транскодирования в потоках с несколькими источниками и транскодированием, поскольку все источники потока транскодируются теперь один раз согласно параметрам, указанным в `transcoder`.

```
stream sdi {
  input decklink://0;
  input fake://fake;
  transcoder vb=3096k ab=64k;
}
```

- Экономия ресурсов (по той же причине).
- ”Бесшовное” переключение между SDI и другими источниками потока.
- Удобство настройки SDI источника через UI — теперь не нужно указывать опции транскодера отдельно для SDI источника, нет необходимости редактировать файл `flussonic.conf` и затем применять конфигурацию.
- Это позволит использовать аппаратные транскодеры для кодирования видео, полученного из Decklink SDI (в будущих версиях Flussonic).

Если вы не укажете настройки транскодирования в `transcoder`, то поток работать **не** будет.

Устранение чересстрочности в прогрессивных потоках

Flussonic может устранять чересстрочность в прогрессивных потоках для улучшения качества видео. Для этого следует использовать метод деинтерлейсинга **CUDA yadif**:

```
stream test {
  input decklink://1 vinput=sdi;
  transcoder vb=4000k hw=nvenc preset=slow fps=50 deinterlace=yadif ab=128k
    ;
}
```

Захват SD видео с плат SDI

Flussonic поддерживает видео с неквадратными пикселями (анаморфное видео) при захвате с SDI карт. Зачастую это видео SD (standard definition) качества.

Например, при захвате каналов в формате PAL, в выходном потоке может появиться искажение пропорций изображения если соотношение сторон пикселя не 1:1. Подавляющее большинство устройств ожидает, что соотношение сторон пикселя 1:1.

Чтобы *Flussonic* сохранил пропорции в выходном видео без искажения картинки, следует указать **sar** входящего потока:

```
stream test {
  input decklink://1 vinput=hdmi sar=16:11;
}
```

Flussonic исходя из `sar` вычисляет разрешение выходного видео. В примере с `sar=16:11` входящее анаморфное видео 720x576 пройдет внутри *Flussonic* с разрешением 1048x576.

Эта настройка работает при захвате с плат Decklink и StreamLabs.

Дуплексный режим работы DeckLink

Flussonic позволяет установить дуплексный режим для карт Decklink. При таком режиме порты можно использовать по отдельности для ввода или вывода, или как комбинацию ввода и вывода. Подробнее о том, как настроить дуплексный режим для карт DeckLink, см. Дуплексный режим работы.

Работа карт DeckLink в режиме 24x7

Поработав с картами DeckLink в лаборатории и в продакшене, мы заметили, что есть высокая вероятность смещения таймкодов или некорректной их передаче, особенно при длительном использовании. Эти карты хороши для коротких трансляций, но не для работы 24x7.

Для эксплуатации в критически важных системах, где важна надежность и стабильность, мы рекомендуем использовать карты Dektec.

Если кто-то еще хочет с нами поработать, чтобы добавить карточки в список рекомендуемых - пишите, протестируем.

6.1.3 Отправка потока на Decklink SDI

Flussonic Media Server может не только захватывать, но и передавать поток на плату захвата и вывода Decklink SDI или HDMI.

Вывод на плату Decklink SDI или Decklink HDMI

Для вывода на Decklink укажите параметр `push decklink://`:

```
stream test {
  input udp://239.0.0.1:1234;
  push decklink://0;
}
```

Flussonic декодирует и затем передает поток на указанный номер устройства или порт на самой карте (например, 0). При необходимости можно указать опцию `deinterlace=true` для устранения чересстрочности.

Режимы карты Decklink

Обычно карта Decklink поддерживает ограниченный набор режимов. Каждый режим — это комбинация размера кадра и кадровой частоты, закодированные в формате Decklink. Например, 1080i50 означает размер кадра 1920x1080 и кадровую частоту 50000/1000 FPS. Отправляя поток на карту Decklink, вы можете задать значение режима в параметре `format`. Например:

```
stream test {
  input ...;
  push decklink://0 video_format=1080i50;
}
```

Возможные режимы работы Decklink описаны в API

Дуплексный режим работы

Чтобы указать для карты DeckLink SDI режим работы Duplex, позволяющий выбрать направление входа и выхода, используйте следующую конфигурацию в глобальных настройках DeckLink:

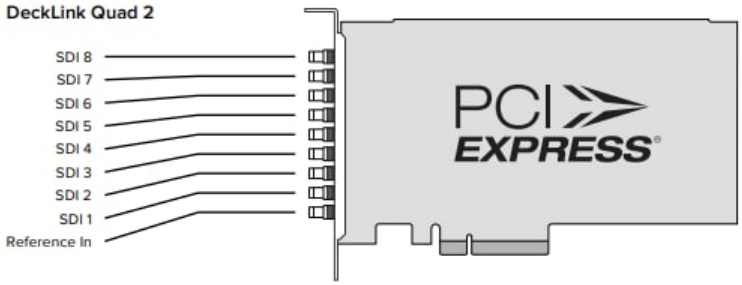
```
decklink 0 {
  profile two_half;
}

decklink 1 {
  profile two_half;
}
```

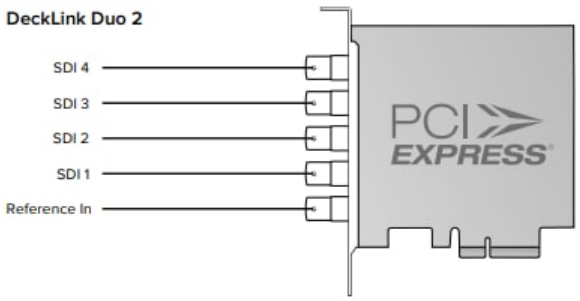
Пример выше показывает конфигурацию полудуплексного режима для карты DeckLink Duo 2.

Карты DeckLink Quad 2 и DeckLink Duo 2 имеют нестандартную нумерацию при сопоставлении (маппинге) физических и логических портов (см. *Рис. 1*), что влияет на конфигурацию дуплексных режимов.

Рис. 1. Сопоставление (маппинг) логических и физических портов карт DeckLink Quad 2 и DeckLink Duo 2.



Sub-device index	2 sub-devices profile (bmdProfileTwoSubDevicesHalfDuplex)	1 sub-device profile (bmdProfileOneSubDeviceFullDuplex)
0	SDI 1	SDI 1 (in/key) & SDI 2 (out/fill)
1	SDI 3	SDI 3 (in/key) & SDI 4 (out/fill)
2	SDI 5	SDI 5 (in/key) & SDI 6 (out/fill)
3	SDI 7	SDI 7 (in/key) & SDI 8 (out/fill)
4	SDI 2	-
5	SDI 4	-
6	SDI 6	-
7	SDI 8	-



Sub-device index	2 sub-device profile (bmdProfileTwoSubDevicesHalfDuplex)	1 sub-device profile (bmdProfileOneSubDeviceFullDuplex)
0	SDI 1	SDI 1 (in/key) & SDI 2 (out/fill)
1	SDI 3	SDI 3 (in/key) & SDI 4 (out/fill)
2	SDI 2	-
3	SDI 4	-

Figure 6.7: Decklink Duo2 and Quad2 mapping

Давайте настроим карту DeckLink Quad 2 так, чтобы все порты использовались либо как вход, либо как выход.

В этом случае карта должна работать в режиме `two_half`.

Чтобы установить режим `two_half` для карты DeckLink Quad 2, используйте приведенную ниже конфигурацию:

```
decklink 0 {
  profile two_half;
}
decklink 1 {
  profile two_half;
}
decklink 2 {
  profile two_half;
}
decklink 3 {
  profile two_half;
}
```

6.1.4 H323

Flussonic Media Server может звонить и захватывать видео по VoIP протоколу H323, например, с устройств Polycom.

Пример конфигурации:

```
stream polycom1 {
  input h323://192.168.100.150 vb=2000k id="Flussonic";
}
```

Flussonic Media Server подключится по указанному адресу и будет кодировать видео с битрейтом определенным опцией `vb`. Звук автоматически будет кодироваться в кодек AAC.

Параметр `id` позволяет задать имя, которое будет высвечиваться на конечном устройстве при подключении Flussonic.

6.1.5 Адаптивная публикация по WebRTC

При публикации из браузера на *Flussonic*, *Flussonic* управляет браузером, чтобы тот подстраивал битрейт публикации под пропускную способность канала. Это позволяет исключить потерю пакетов при недостаточной пропускной способности интернет-соединения. Если снизить ширину канала, клиент должен уменьшить битрейт публикации, если расширить канал, клиент должен увеличить битрейт публикации.

Эта функция используется по умолчанию и обычно не требует настройки, поскольку параметры по умолчанию подобраны оптимально. Тем не менее, вы можете менять значения по своему усмотрению, чтобы добиться еще большей эффективности.

Настройки ABR публикации через веб-интерфейс

Для указания дополнительных настроек источника публикации, нажмите **options** напротив URL источника. Настройки адаптивного битрейта находятся под заголовком **WebRTC** > **ABR**.

Этим настройкам в UI соответствуют следующие настройки в файле:

```
stream published_stream_name {
  input publish:// abr_loss_lower=2 abr_loss_upper=10 abr_mode=1
    abr_stepdown=50 frames_timeout=1 max_bitrate=2200 min_bitrate=500
    output_audio=aac priority=0 source_timeout=5;
}
```

Flussonic рекомендует браузеру битрейт в пределах `min_bitrate - max_bitrate` в зависимости от наличия и количества потерь пакетов при публикации. *Flussonic* рекомендует понижать битрейт при количестве потерь **более**, чем `abr_loss_upper` процентов и повышать при количестве потерь **менее**, чем `abr_loss_lower` процентов. Понижение и повышение происходит шагами размером `abr_stepdown` и `abr_stepup` соответственно. После указанного количества циклов автоподстройки (`abr_cycles`) *Flussonic* считает битрейт оптимальным и больше не анализирует его. По умолчанию `abr_cycles=5`. При `abr_cycles=0` процесс подстройки происходит всё время, пока длится публикация.

Также *Flussonic* вычисляет текущий максимальный битрейт. Он запоминает значения битрейта, после которых потери выросли до `abr_loss_upper` и принимает их среднее значение за прошедшее количество циклов новым максимальным значением битрейта (текущим).

Figure 6.8: Настройки адаптивной публикации WebRTC

6.1.6 Прием публикаций по SRT от множества авторов

Чтобы принять публикацию по SRT от множества авторов, централизованно выдав каждому автору уникальный passphrase, используйте механизм `srt_port_resolve`, предлагаемый в Flussonic для публикации по SRT в облако. Подробнее об этом решении см. на странице SRT.

Альтернатива — добавить потоки в конфигурацию Flussonic вручную. Но этот способ плохо работает, когда у вас десятки серверов и тысячи авторов: нужно будет как-то определять, на каком сервере принимать поток от того или иного автора, а местом хранения passphrase будут не синхронизируемые автоматически файлы конфигурации на серверах.

Шаг 1. Создайте конфигурационный бэкенд

Каждый автор будет публиковать свой контент на заранее выделенный ему порт, используя свой passphrase. Конфигурационный бэкенд будет сопоставлять номер порта имени потока, а имя потока — passphrase. Ниже приведен простой пример такого конфигурационного бэкенда.

```
from http.server import BaseHTTPRequestHandler, HTTPServer
import urllib.parse as urlparse

class RequestHandler(BaseHTTPRequestHandler):
    streams_config = {
        "streams": [
            {
                "name": "publish_stream_name",
                "inputs": [
                    {"url": "publish://"}
                ],
                "srt_publish": {"passphrase": "securePass"}
            },
            {
                "name": "other_publish_stream",
                "inputs": [
                    {"url": "publish://"}
                ],
                "srt_publish": {"passphrase": "otherPassword"}
            }
        ]
    }

    port_stream_mapping = (
        (2345, "publish_stream_name"),
        (2346, "other_publish_stream"),
    )

    def do_GET(self):
        parsed_path = urlparse.urlparse(self.path)
        path = parsed_path.path
        query_components = urlparse.parse_qs(parsed_path.query)

        # Handle /config_backend/streams
        if path.startswith('/config_backend/streams'):
            self.send_response(200)
            self.send_header('Content-Type', 'application/json')
            self.end_headers()
            response = self.streams_config
            self.wfile.write(bytes(str(response), "utf8"))

        # Handle /config_backend/srt_port_resolve/{port}
        elif path.startswith('/config_backend/srt_port_resolve/'):
            port = int(path.split('/')[-1])
            mode = query_components.get('mode', [None])[0]
            host = query_components.get('host', [None])[0]
```

```
            response = next((stream_name for port_number, stream_name in
    self.port_stream_mapping if port_number == port), None)

            if response and mode == 'publish':
                self.send_response(200)
                self.send_header('Content-Type', 'text/plain')
                self.end_headers()
                self.wfile.write(bytes(response, "utf8"))
            else:
                self.send_error(400, "Invalid port or mode.")
                return

        else:
            self.send_error(404, "404 Not Found")

def run(server_class=HTTPServer, handler_class=RequestHandler, port=12345):
    server_address = ('', port)
    httpd = server_class(server_address, handler_class)
    print(f'Starting httpd on port {port}...')
    httpd.serve_forever()

if __name__ == "__main__":
    from sys import argv

    if len(argv) == 2:
        run(port=int(argv[1]))
    else:
        run()
```

Чтобы запустить этот конфигурационный бэкенд:

- Создайте в любом текстовом редакторе файл `/tmp/config_server.py` с приведенным выше кодом.
- Установите Python, если еще не установлен.
- Запустите конфигурационный бэкенд на свободном порту, например 12345.

Шаг 2. Настройте Flussonic для использования бэкенда

Flussonic должен знать, откуда ему брать конфигурацию потока и на каких портах ожидать публикации по SRT. Отправьте ему адрес конфигурационного бэкенда и диапазон портов SRT с помощью API-запроса PUT /streamer/api/v3/config:

```
curl -u USER:PASSWORD -X PUT http://localhost:80/streamer/api/v3/config \
-H "Content-Type: application/json" \
--data '{"listeners": {"srt": [ {"mode":"publish","ports":{"first":2345,"
   last":12344}}]},"config_external": "http://localhost:12345/
   config_backend/"}'
```

Через 5—10 секунд после выполнения этой команды в UI должны создаться потоки `publish_stream_name` и `other_publish_stream`, которые *Flussonic* получит от конфигурационного бэкенда.

Шаг 3. Проверьте, что все работает

Опубликуйте поток во *Flussonic*:

```
ffmpeg -re -i /opt/flussonic/priv/bunny.mp4 -c copy -y -f mpegts 'srt://
   localhost:2345?passphrase=securePass'
```

Проверьте в UI, что видео проигрывается в профиле потока `publish_stream_name` на вкладке **Overview**.

Связанные задачи

- Централизованное управление конфигурацией кластера с помощью config_external

6.2 Restream

6.2.1 Публикация в социальные сети

Flussonic Media Server позволяет публиковать любой поток на внешний сервер по протоколу RTMP.

Социальные сети используют RTMP для организации прямых трансляций, а значит вы можете использовать Flussonic Media Server для отправки ваших потоков в социальные сети (можно сразу в несколько).

Сценарии применения:

- Получение видео от мобильного репортера и отправка сразу в несколько социальных сетей.
- Трансляция видео с камер видеонаблюдения.
- Трансляция собственных программ в социальные сети. В том числе по расписанию.

Обратите внимание, ключи трансляции/потока могут иметь срок жизни. Уточните этот момент в условиях сервиса, куда планируете публиковать видеотрансляцию.

Содержание:

- Публикация на Youtube
- Публикация на Facebook
- Публикация на OK

Публикация на YouTube

См. Рестриминг на YouTube в высоком качестве.

Публикация на Facebook

1) Зайдите на **Facebook** > **Прямой эфир** > **В эфир** > **Подключиться**

2) Скопируйте URL-адрес сервера и ключ потока.

3) В административном интерфейсе Flussonic Media Server перейдите в меню **«Media»** и выберите поток, который необходимо раздать.

4) Во вкладке **«Output»** найдите раздел **«Push live video to certain URLs»**.

5) Вставьте URL-адрес сервера и ключ потока в виде ссылки. Например, `rtmp://live-api.facebook.com` Нажмите **«Save»**.

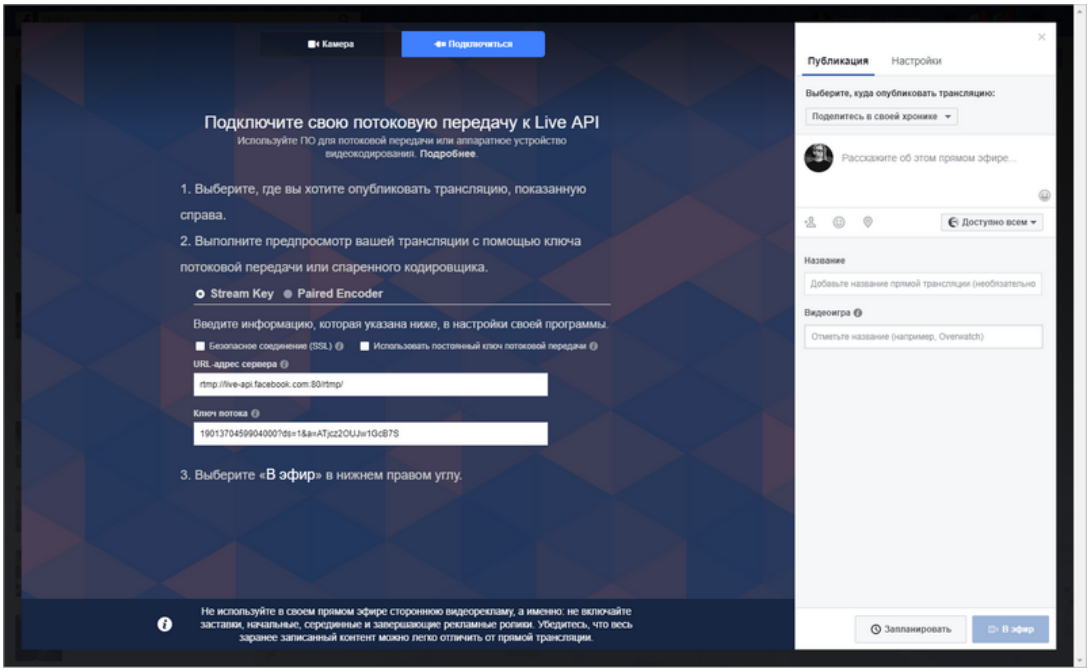


Figure 6.9

6) Вернитесь на **Facebook** > **Прямой эфир** > **В эфир** > **Подключиться** и запустите прямой эфир.

Публикация на OK

1) Зайдите на **OK.ru** > **Трансляция** > **Приложение**.

2) Скопируйте URL-адрес сервера и ключ трансляции.

3) В административном интерфейсе Flussonic Media Server перейдите в меню **«Media»** и выберите поток, который необходимо раздать.

4) Во вкладке **«Output»** найдите раздел **«Push live video to certain URLs»**.

5) Вставьте URL-адрес сервера и ключ трансляции в виде ссылки. Например, `rtmp://vsu.mycdn.me/input`
Нажмите **«Save»**.

6) Вернитесь в **OK.ru** > **Трансляция** > **Приложение** и запустите прямой эфир.

Figure 6.10

6.2.2 Рестриминг на YouTube в высоком качестве

Используйте Enhanced RTMP, чтобы отправлять видео на YouTube в высоком качестве в кодеках AV1 и HEVC. Подробнее об Enhanced RTMP читайте в этой статье.

Flussonic Media Server может отправить по Enhanced RTMP любой поток, например, публикацию по WebRTC, IP-камеру, серверный плейлист и т.д. Далее рассмотрим пример с рестримингом потока, полученного по Enhanced RTMP.

Шаг 1. Подготовка Flussonic к приему публикации по Enhanced RTMP

Чтобы принять публикацию:

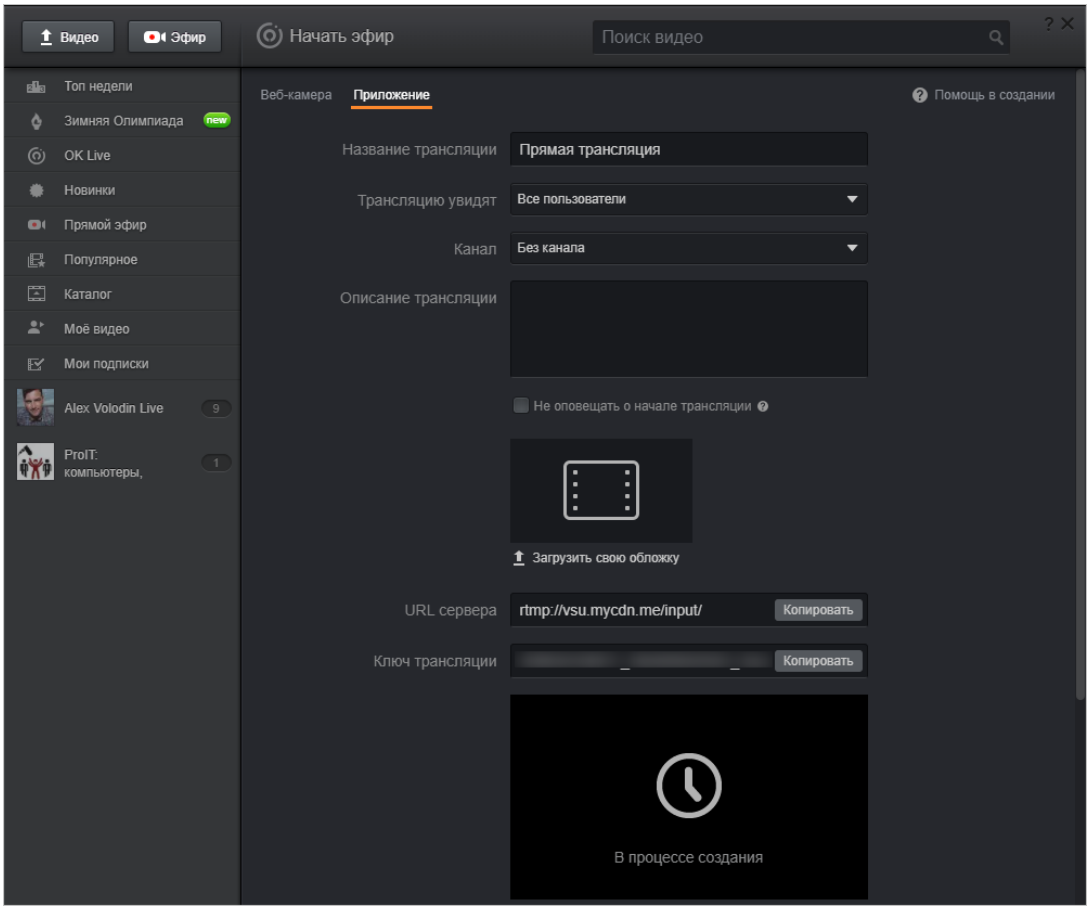


Figure 6.11

- На вкладке **Config** -> **Settings** задайте порт для приема публикации по RTMP. Обычно используется порт 1935.
- Создайте новый поток, нажав **+** на странице **Media** -> **Streams**. В форме создания задайте имя, например `published`, и установите флажок **Publication**.

Шаг 2. Получение URL для отправки видео на Youtube

- На YouTube выберите **Творческая студия YouTube** > **Контент** > **Прямые трансляции** и нажмите **Начать**. Вы увидите настройки трансляции.
- Вам понадобится URL-адрес сервера и ключ трансляции.

Шаг 3. Отправка из Flussonic на YouTube по Enhanced RTMP

Настройте отправку видео из Flussonic на YouTube:

Figure 6.12

- Перейдите на вкладку **Output** созданного на шаге 1 потока `published`.
- В разделе **Push video to certain URLs** укажите скопированные из творческой студии YouTube URL и ключ трансляции в виде ссылки. Например, так: `rtmp://a.rtmp.youtube.com/live2/7p9v-`
  Нажмите **«Save»**.

Шаг 4. Рестриминг

- Перейдите на вкладку **Input** потока `published`. Скопируйте ссылку **RTMP** в группе **Publish links: start publishing to get preview**.
- Опубликуйте видео в Flussonic из любого приложения, поддерживающего Enhanced RTMP, например с помощью ffmpeg:

Для отправки Enhanced RTMP требуется версия ffmpeg не ниже 6.1.1.

```
ffmpeg -i /opt/flussonic/priv/bunny.mp4 -c:v libx265 -preset fast -c:a copy
    -f flv rtmp://localhost:1935/static/published
```

Если все настроено верно, вы увидите трансляцию файла в своем канале на YouTube.

Что делать, если в потоке нет аудио

Стандарт YouTube требует наличия аудиопотока для всех видео. Может оказаться так, что в вашем видео нет звука, например, если это поток с IP-камеры или опубликованный по WebRTC из браузера. Flussonic Media Server может добавить пустую беззвучную дорожку в поток, чтобы он нормально проигрывался на YouTube.

Чтобы добавить беззвучную дорожку в поток:

- Перейдите на вкладку **Input** в профиле потока и нажмите **Options** рядом с адресом источника.
- Выберите в списке **Output audio** значение **Add_AAC**.

6.3 Playback

Figure 6.13: Publishing urls

6.3.1 Наложение логотипа

Flussonic Media Server позволяет наложить изображение на видео двумя способами:

- С помощью embed.html плеера. Поверх плеера накладывается прозрачный слой с изображением. Этот способ не создает дополнительной нагрузки на сервер и отлично подходит для вставки видео на сайт.
- С помощью транскодера. Транскодер «вшивает» изображение в видеодорожку так, что логотип не получится удалить или скрыть. Это ресурсоёмкий процесс. Такой способ подходит для видео, которое проигрывается на ТВ-приставках.

Наложение логотипа с помощью embed.html плеера

Вы можете наложить лого с помощью embed.html плеера на необходимый поток двумя способами:

- В Flussonic UI
- Через конфигурационный файл Flussonic

Логотип будет отображаться как в live-потоке, так и в DVR-плеере.

В Flussonic UI

- Перейдите на **Media** > **Streams** и откройте настройки потока, кликнув на имя потока.
- Перейдите на вкладку **Output** и найдите раздел **Logo**.
- Загрузите логотип, кликнув **Select** > **Add New** и выбрав изображение с логотипом. Вы можете загрузить несколько изображений.

> Необходимо загрузить логотип в формате .png.

- (Необязательно) Измените размер изображения и его расположение на видео с помощью следующих параметров:
- Выберите необходимое изображение, кликнув на радиокнопку напротив.
- Сохраните настройки, кликнув **OK** > **Save**.
- Проверьте, что логотип отображается на видео.

Через конфигурационный файл Flussonic

- Загрузите изображение на сервер с помощью метода `Flussonic-API: PUT /streamer/api/v3/l`
- Откройте конфигурационный файл `flussonic.conf`.
- В настройках потока добавьте директиву `logo` и укажите название файла с логотипом в параметре `path`, начиная с `@`.
- (Необязательно) Измените размер изображения и его расположение на видео с помощью следующих параметров:
- Проверьте, что логотип отображается на видео.

```
stream example {
  input udp://239.0.0.1:1234;
  logo path=@logo.png height=100 width=100 left=0 top=0;
}
```

В примере использованы следующие параметры:

- `path` — имя файла с логотипом, начиная с `@`.
- (Необязательный) `height`, `width` — размер изображения логотипа в пикселях. Если задан только один из этих параметров, то второй будет изменен пропорционально. Не указывайте эти параметры, чтобы отобразить логотип в исходном размере.
- (Необязательный) `left`, `top`, `right`, `bottom` — положение логотипа, заданное в виде смещения в пикселях от левого, верхнего, правого и нижнего края видео. Например, чтобы отобразить логотип в правом нижнем углу: `right=0, bottom=0`. Не используйте одновременно параметры `left` и `right`, `top` и `bottom`.

Наложение логотипа с помощью транскодера

Такой логотип будет ”вшиваться” в видеодорожку и отображаться на всех устройствах и в записях архива.

Пример конфигурации:

```
stream example {
  input udp://239.0.0.1:1234;
  transcoder vb=2048k logo=/storage/logo.png@10:10 ab=128k;
}
```

Здесь `10:10` — это координаты от левого верхнего угла экрана.

Чтобы разместить лого в других частях экрана, используйте более сложную формулу, как в следующих примерах:

- Для размещения в центре:

```
stream example {
  input udp://239.0.0.1:1234;
  transcoder vb=2048k logo=/storage/logo.png@(main_w-overlay_w-10)/2:(
   main_h-overlay_h-10)/2 ab=128k;
}
```

- Для размещения в левом нижнем углу:

```
stream example {
  input udp://239.0.0.1:1234;
  transcoder vb=2048k logo=/storage/logo.png@10:(main_h-overlay_h-10) ab
    =128k;
}
```

- Для размещения в правом верхнем углу:

```
stream example {
  input udp://239.0.0.1:1234;
  transcoder vb=2048k logo=/storage/logo.png@(main_w-overlay_w-10):10 ab
    =128k;
}
```

- Для размещения в правом нижнем углу:

```
stream example {
  input udp://239.0.0.1:1234;
  transcoder vb=2048k logo=/storage/logo.png@(main_w-overlay_w-10):(main_h-
    overlay_h-10) ab=128k;
}
```

Подробнее про настройку транскодера см. Настройки транскодера.

6.3.2 Вставка видео на сайт (embed.html)

У сервера Flussonic Media Server есть специальная страница — **embed.html**, которая предназначена для вставки видео на сайт и просмотра видео через браузер. Технически `embed.html` — тот же плеер, что используется в административном интерфейсе Flussonic Media Server.

Страница с плеером доступна по ссылке:

`http://HOSTNAME/STREAMNAME/embed.html`

Страница автоматически определяет браузер и выбирает поддерживаемый видео-протокол. Для большинства устройств на сегодня — HLS (используется плеером по умолчанию).

> Проигрывание видео в браузере может начаться без звука по причине политики, принятой разработчиками браузера. По ссылке объясняется политика и условия, при которых звук включится сам.

Вызывается плеер двумя способами:

- При открытии `embed.html` напрямую (указав ссылку в адресной строке) видео развернется на размер окна браузера и автоматически начнет воспроизведение.

- Также `embed.html` удобен для вставки видео на сайт как части веб-страницы. HTML-код для вставки доступен на странице **Overview** в свойствах каждого потока в административном интерфейсе.

Пример:

```
<iframe style="width:640px; height:480px;" allowfullscreen src="http://
   hostname/streamname/embed.html"></iframe>
```

Код вставляет на страницу окно с плеером фиксированного размера — 640x480 пикселей. Воспроизведение начинается автоматически.

Дополнительные параметры

Для большинства задач никакой дополнительной настройки не требуется, но `embed.html` имеет параметры, которые можно задать с помощью URL. Дополнительные параметры задаются в адресной строке:

```
http://FLUSSONIC-IP/STREAM_NAME/embed.html?autoplay=false&play_duration=600
```

В этом примере видео будет воспроизводиться без автозапуска, при этои проигрывание прекратится через 10 минут.

Figure 6.14: Вставка видео на сайт

Подробное описание всех дополнительных параметров проигрывания можно найти в разделе **Query parameters** в справочнике Streaming API.

Если у потока есть несколько аудио- и видеодорожек, вы можете указать дорожки, которые хотите воспроизвести, использовав параметр ‘filter.tracks‘.

Пример доступа к видео из архива

Например, ссылка для доступа к записи телепередачи будет содержать параметры `from` и `to`:

```
http://FLUSSONIC-IP/STREAM_NAME/embed.html?from=1511300552&to=1511300852
```

Такие ссылки лучше формировать с помощью серверных скриптов на основе программы передач (EPG) для организации CatchUp сервиса.

DVR-плеер

Чтобы просмотреть архив записи для потока, откройте DVR-плеер при помощи ссылки:

```
http://FLUSSONIC-IP/STREAM_NAME/embed.html?dvr=true
```

Плеер позволяет проиграть видео из архива, для больших архивов доступен календарь, а не только временная шкала. Интерфейс плеера позволяет задать масштаб временной шкалы, включить ускоренное воспроизведение и сохранить указанный интервал в виде MP4 файла.

Figure 6.15: DVR плеер

Для DVR-плеера доступны все дополнительные параметры адреса, кроме **ago**.

Интерфейс плеера позволяет автоматически генерировать ссылки формата **embed.html?dvr=true&from=1511**
без использования дополнительных утилит. Просто откройте нужный момент в архиве и нажмите

на часы, чтобы открыть ссылку с параметром **from**.

См. также:

- Обо всех способах проиграть архив можно прочитать в разделе Проигрывание архива

Сейчас эта функциональность доступна с использованием э параметра `streams`, который возможно, будет изменен. А параметров `embed.html` можно найти в справочнике Streaming AP

Мультиоконный режим DVR-плеера

В некоторых ситуациях может быть необходимо просмотреть архивы с нескольких видеопотоков в одном плеере с общей временной шкалой. Например, вы можете захотеть синхронно просмотреть записи с нескольких камер наблюдения, чтобы видеть одно и то же помещение с разных точек зрения. В этом случае можно использовать DVR-плеер в мультиоконном режиме.

Для этого можно добавить в ссылку для проигрывания DVR параметр `streams` и перечислить в нем через запятую все необходимые потоки:

```
http://FLUSSONIC-IP/STREAM_NAME/embed.html?dvr=true&streams=cam01,cam02,
    cam03,cam04
```

Вместо STREAM_NAME в этот URL-адрес можно подставить имя любого потока, т.к. для проигрывания будут использоваться потоки, перечисленные в параметре `streams`.

В результате все архивы будут проигрываться в отдельных окнах внутри DVR-плеера.

Авторизация потоков

Flussonic Media Server имеет встроенные механизмы базовой защиты от вставки плеера на других сайтах. Более подробно про такую защиту вы можете прочесть в разделах Domain lock и CORS для защиты плеера.

Figure 6.16: Multiwindow DVR

6.3.3 Рекомендации по созданию клиентского приложения

Вы можете использовать *Flussonic* в качестве сигнального сервера при разработке веб-сайта или приложения для обмена видео и/или аудио через WebRTC. Например, с помощью *Flussonic* вы можете создать приложение для видеоконференций (соединение «многие-ко-многим»), веб-сайт для проведения вебинаров или мастер-классов (соединение «один-ко-многим») и т.п. Ниже вы найдете рекомендации по использованию инструментов *Flussonic* в своем проекте.

Не забудьте настроить *Flussonic* на прием публикаций из вашего приложения, а также ознакомьтесь со способами воспроизведения потоков из *Flussonic* по WebRTC.

Для создания кода используйте **библиотеку Flussonic WebRTC player**. Ее можно установить одним из описанных ниже способов.

Инструкция по установке, описание классов библиотеки и код примера доступны на npm.

Установка с помощью NPM и webpack

Для варианта с импортом библиотеки в ваш проект через Webpack необходимо загрузить пакет:

```
npm install --save @flussonic/flussonic-webrtc-player
```

и импортировать его в ваше приложение:

```
import {
  PUBLISHER_EVENTS,
  PLAYER_EVENTS,
  Player,
  Publisher,
} from "@flussonic/flussonic-webrtc-player";
```

См. также демо-приложение ниже.

Установка без NPM и webpack

В секцию скриптов вашего HTML-файла добавьте:

```
<script src="https://cdn.jsdelivr.net/npm/@flussonic/flussonic-webrtc-player/dis
```

Полный пример кода страницы с плеером приведен ниже.

Пример плеера — с Webpack и без Webpack

Демо-приложение, использующее Webpack для импорта компонентов:

- .

Пример плеера WebRTC на JavaScript, который получает компоненты через `<script>`:

- Код библиотеки *Flussonic* WebRTC Player для реализации плеера есть в CDN https://www.jsdelivr.com, откуда его нужно импортировать в свою веб-страницу. Для этого добавьте в секцию скриптов вашего HTML-файла строку: `<script src="https://cdn.jsdelivr.net/npm/@fluss`

Полный пример страницы с плеером на JavaScript (похожий код есть в составе демо-приложения):

```
<!DOCTYPE html>
<html>
  <head>


        <style>
      .app {
        display: flex;
        flex-direction: column;
        justify-content: space-between;
        height: calc(100vh - 16px);
      }
      .container {
        margin-bottom: 32px;
      }
      .video-container {
        display: flex;
      }
      .controls {
      }
      .config {
      }
      #player {
        width: 640px; height: 480px; border-radius: 1px
      }
      .button {
        height: 20px;
        width: 96px;
      }
      .preview {
        position: absolute;
        right: 0;
        bottom: 0;
        z-index: 100;
      }

      .preview-text {
        position: absolute;
        left: 0;
        top: 0;
        padding: 8px;
        background: black;
        color: white;
        z-index: 10;
      }

      #preview-video {
        width: 320px;
        height: auto;
        max-width: 320px;
        max-height: 240px;
      }
```

```
  </style>
  <script src="https://cdn.jsdelivr.net/npm/@flussonic/flussonic-webrtc-
 player/dist/index.js"></script>
</head>
<body>
  <div class="app">
  <div class="preview">
      <div class="preview-text">preview</div>
      <video id="preview-video" controls="false" muted autoplay
 playsinline />
  </div>
    <div class="video-container">
      <video
              id="player"
              controls
              muted
              autoplay
              playsinline
      >
      </video>
      <pre id="debug"></pre>
    </div>

  <div class="container">
    <div class="config" id="config">
      <span id="hostContainer">
        <label for="host">Host: </label><input name="host" id="host"
 value="" />
      </span>
      <span id="nameContainer">
        <label for="name">Stream: </label><input name="name" id="name"
 value="" />
      </span>
    </div>
    <div class="controls" id="controls">
      <select id="quality">
        <option value="4:3:240">4:3 320x240</option>
        <option value="4:3:360">4:3 480x360</option>
        <option value="4:3:480">4:3 640x480</option>
        <option value="16:9:360" selected>16:9 640x360</option>
        <option value="16:9:540">16:9 960x540</option>
        <option value="16:9:720">16:9 1280x720 HD</option>
      </select>
      <button id="publish" class="button">Publish</button>
      <button id="play" class="button">Play</button>
      <button id="stop" class="button">Stop all</button>
    </div>
    <div class="errorMessageContainer" id="errorMessageContainer"></div>
  </div>

<script>
  let wrtcPlayer = null;
```

```
 let publisher = null;

 const { Player, Publisher, PUBLISHER_EVENTS, PLAYER_EVENTS } = this.
FlussonicWebRTC;

 const getHostElement = () => document.getElementById('host');
 const getHostContainerElement = () => document.getElementById('
hostContainer');
 const getNameElement = () => document.getElementById('name');
 const getNameContainerElement = () => document.getElementById('
nameContainer');
 const getPlayerElement = () => document.getElementById('player');
 const getPlayElement = () => document.getElementById('play');
 const getPublishElement = () => document.getElementById('publish');
 const getStopElement = () => document.getElementById('stop');
 const getQualityElement = () => document.getElementById('stop');

 const getStreamUrl = (
   hostElement = getHostElement(),
   nameElement = getNameElement(),
 ) =>
   `${hostElement && hostElement.value}/${nameElement && nameElement.
value}`;
 const getPublisherOpts = () => {
   const [, , height] = document.getElementById('quality').value.split
(/:/);
   return {
     preview: document.getElementById('preview-video'),
     constraints: {
       // video: {
       //   height: { exact: height }
       // },
       video: true,
       audio: true,
     },
     canvasCallback: (canvasElement) => {
         window.myCanvasElement = canvasElement;
     },
   };
 };

 const getPlayer = (
   playerElement = getPlayerElement(),
   streamUrl = getStreamUrl(),
   playerOpts = {
     retryMax: 10,
     retryDelay: 1000,
   },
   shouldLog = true,
   log = (...defaultMessages) => (...passedMessages) =>
     console.log(...[...defaultMessages, ...passedMessages]),
 ) => {
```

```
    const player = new Player(playerElement, streamUrl, playerOpts, true)
;
    player.on(PLAYER_EVENTS.PLAY, log('Started playing', streamUrl));
    player.on(PLAYER_EVENTS.DEBUG, log('Debugging play'));
    return player;
  };

  const stopPublishing = () => {
    if (publisher) {
      publisher.stop && publisher.stop();
      publisher = null;
    }
  };

  const stopPlaying = () => {
    if (wrtcPlayer) {
      wrtcPlayer.destroy && wrtcPlayer.destroy();
      wrtcPlayer = null;
    }
  };

  const stop = () => {
    stopPublishing();
    stopPlaying();

    getPublishElement().innerText = 'Publish';
    getPlayElement().innerText = 'Play';
  };

  const play = () => {
    wrtcPlayer = getPlayer();
    getPlayElement().innerText = 'Playing...';
    wrtcPlayer.play();
  };

  const publish = () => {
    if (publisher) publisher.stop();

    publisher = new Publisher(getStreamUrl(), getPublisherOpts(), true);
    publisher.on(PUBLISHER_EVENTS.STREAMING, streaming);
    publisher.start();
  };

  const setDefaultValues = () => {
      getHostElement().value = config.host;
      getNameElement().value = config.name;
  };

  const setEventListeners = () => {
    // Set event listeners
    getPublishElement().addEventListener('click', publish);
    getPlayElement().addEventListener('click', play);
```

```
      getStopElement().addEventListener('click', stop);
      getQualityElement().onchange = publish;
    };

    const main = () => {
      setDefaultValues();
      setEventListeners();
    };

    const streaming = () => {
        getPublishElement().innerText = 'Publishing...';
        getPublishElement().disabled = true;

        // drawing on publishing canvas
        if (window.myCanvasElement) {
          const ctx = window.myCanvasElement.getContext('2d', { alpha:
   false });
          ctx.filter = 'sepia(0.75)'; // Testing filters
          ctx.font = '128px sans-serif';
          ctx.fillStyle = 'white';
          ctx.textAlign = 'center';
          ctx.textBaseline = 'center';
          (function loop() {
            ctx.fillText(
              `It's publishing to Flussonic!`,
              window.myCanvasElement.width / 2,
              window.myCanvasElement.height / 2,
              window.myCanvasElement.width - 100,
            );
            setTimeout(requestAnimationFrame(loop), 1000 / 30); // drawing
   at 30fps
          })();
      }
    };

    window.addEventListener('load', main);
  </script>
    </body>
</html>
```

Скопируйте этот код в файл, например `index.html`, и откройте в браузере, чтобы проверить работу.

6.3.4 MSE-плеер

Здесь мы расскажем, как использовать наш MSE-плеер с открытым исходным кодом в Вашем frontend-проекте для того, чтобы организовать проигрывание видео с низкой задержкой. Плеер давно работает в нашем модуле `embed.html`.

Почему MSE Player?

- Использует HTML5 и не требует Flash, что значит, что он поддерживается любым клиентским устройством (браузер, смартфон).
- Имеет преимущества перед WebRTC. А точнее: WebRTC требует UDP, а MSE — HTTP, что позволяет MSE проще проходить через корпоративные файерволы и прокси. Кроме того, WebRTC и MSE поддерживают разные кодеки, например, аудио: MSE поддерживает AAC аудио кодек, а WebRTC — нет, что значит, что условный ТВ-канал проще проиграть через MSE Player.

Этот плеер вы видите в веб-интерфейсе *Flussonic* и *Watcher*, а также его можно открыть отдельно из браузера по следующей ссылке:

```
http://flussonic-ip/STREAMNAME/embed.html?realtime=true
```

О механизме рассказано в разделе HTML5 (MSE-LD) воспроизведение с низкой задержкой.

JavaScript-модуль плеера можно использовать в ваших приложениях. Мы опубликовали исходники на GitHub.

На этой странице:

- Установка модуля
- Установка треков в мультибитрейтном видео
- Полный пример использования
- Получение статистики
- Добавление элементов управления как в десктопном плеере

Установка модуля

Для установки с помощью NPM также необходимо выполнить несколько простых шагов:

Шаг 1. Запустите следующую команду:

```
npm install --save @flussonic/flussonic-mse-player
```

Шаг 2. Импортируйте его в JS:

```
import FlussonicMsePlayer from '@flussonic/flussonic-mse-player'
...
const player = new FlussonicMsePlayer(element, url, opts)
```

Пример приложения с webpack и нашим плеером MSE.

Исходный код MSE Player Вы можете найти на Github.

Методы и события FlussonicMsePlayer

```
var player = new FlussonicMsePlayer(element, streamUrl, opts)
```

Параметры:

- `element` — <video> DOM элемент,
- `streamUrl` — адрес видеопотока,
- `opts` — опции плеера.

Вы можете отслеживать работу MSE Player через Sentry, настроив параметр `sentryConfig(string)` (см. табл. ниже). В опциях (`opts`) плеера можно настроить некоторые параметры, список которых приведён ниже:

Параметры	Тип	
`progressUpdateTime`	целое число (секунды)	интервал времени, через котор
`errorsBeforeStop`	целое число	количество ошибо
`connectionRetries`	целое число	колич
`preferHQ`	Булево значение	принуд
`retryMuted`	Булево значение	поп
`maxBufferDelay`	целое число (секунды)	значение максимальной задержки буфера. Если живое
`sentryConfig`	строковый	

Методы Ниже приведена таблица с возможными методами:

| Метод | Описание | |:----|:-----| |`play()`| начать воспроизведение| |`stop()`| остановить воспроизведение| |`setTracks([videoTrackId, audioTrackId])` | установить видео-, аудиодорожки воспроизведения| |`getVideoTracks()` | получить список доступных видеодорожек (использовать в `onMediaInfo`)| |`getAudioTracks()` | получить список доступных аудиодорожек (использовать в `onMediaInfo`)|

События | Событие | Описание | |:----|:-----| |`onProgress(currentTime)` | срабатывает при воспроизведении видео и отдает текущее время проигрывания | |`onMediaInfo(metadata)`| срабатывает при получении данных потока. Содержит общую информацию о воспроизводимом видео. В случае мультибитрейт видео — информацию о всех его треках. Внутри этой функции или после срабатывания доступны методы `getVideoTracks()/getAudioTracks()`|

Установка треков в мультибитрейтном видео

Например, есть видео с тремя видео треками: v1(800k), v2(400k), v3(200k) и двумя аудио: a1(32k), a2(16k).

Чтобы по умолчанию играли v2 и a1, укажем в конструкторе `FlussonicMsePlayer` следующий URL-адрес:

`'ws://flussonic-ip/s/mse_ld?tracks=v2a1'`

Получить все доступные треки можно двумя способами:

- С помощью присвоения атрибуту `onMediaInfo` функции-колбеку, которая при загрузке видеопотока вернет его метаданные:

```
{
  width: ...,
  height: ...,
  streams: [
    {
      track_id: "v1", bitrate: ..., codec: ..., content: "video", fps: ...,
    ...
      ...

      track_id: "a1", bitrate: ..., codec: ..., content: "audio", fps: ...,
    ...
    }
  ]
}
```

- Или с помощью методов `getVideoTracks()/getAudioTracks()`, которые после срабатывания `onMediaInfo` вернут доступные видео-, аудиодорожки соответственно.

С помощью метода `setTracks([videoTrackId, audioTrackId])` вы можете установить нужные дорожки для воспроизведения.

Полный пример использования

```
<html>
  <head>

  </head>
  <style>
    .player-container {
      border: 1px solid black;
    }

    #player {
      position: relative;
      width: 100%;
    }

    .mbr-controls {
      display: none;
    }
  </style>
<body>
  <div class="player-container">
    <video id="player"/>
  </div>
  <div class="mbr-controls">
    <div>
      <label for="videoTracks">video tracks</label>
      <select name="videoTracks" id="videoTracks"></select>
    </div>
    <div>
      <label for="audioTracks">audio tracks</label>
      <select name="audioTracks" id="audioTracks"></select>
    </div>
    <button onclick="window.setTracks()">set tracks</button>
  </div>

  <button onclick="window.player.play()">Play</button>
  <button onclick="window.player.stop()">Stop</button>
  <script type="text/javascript" src="/flu/assets/FlussonicMsePlayer.js"></
   script>
  <script>
    window.onload = onLoad();

      function onLoad() {

        var element = document.getElementById('player');
        var videoTracksSelect = document.getElementById('videoTracks');
        var audioTracksSelect = document.getElementById('audioTracks');
        var mbrControls = document.querySelector('.mbr-controls');

        var url = (window.location.protocol == "https:" ? "wss:" : "ws:")+
   '//'+window.location.host+'/clock/mse_ld';

        window.player = new FlussonicMsePlayer(element, url);
```

```
        window.player.onProgress = function(currentTime) {
          console.log(currentTime);
        };

        window.player.onMediaInfo = (rawMetaData) => {
          var videoTracks = window.player.getVideoTracks()
          var audioTracks = window.player.getAudioTracks()
          var videoOptions = videoTracks.map((v, i) => (
            `<option value="${v['track_id']}">${v['bitrate']} ${v['codec']}
    ${v['fps']} ${v['width']}x${v['height']}</option>`
          ))

          var audioOptions = audioTracks.map(v => (
            `<option value="${v['track_id']}">${v['bitrate']} ${v['codec']}
    ${v['lang']}</option>`
          ))

          videoTracksSelect.innerHTML = videoOptions.join('')
          audioTracksSelect.innerHTML = audioOptions.join('')

          mbrControls.style.display = 'block'
        }

        window.setTracks = () => {
          var videoTrackId = videoTracksSelect.options[videoTracksSelect.
   selectedIndex].value
          var audioTrackId = audioTracksSelect.options[audioTracksSelect.
   selectedIndex].value

          window.player.setTracks([videoTrackId, audioTrackId])
        }
      }
  </script>
</body>
</html>
```

Получение статистики

При инициализации плеера добавьте переменную `config`:

```
this.player = new FlussonicMsePlayer(this._videoElement, url, config);
```

Переменная `config` — это объект с настройками плеера. С помощью опции `onStats`, которую следует передавать с параметром `config`, возвращается объект, содержащий статистику о буферах плеера и время получения статистики.

Добавление элементов управления как в десктопном плеере (Flussonic 20.10)

Теперь *Flussonic MSE Player* поддерживает новые элементы управления, которые есть в обычных настольных плеерах, такие как пауза, пуск или выключение звука. Элементы управления являются частью *MediaElement*, который может быть присоединен к плееру как отдельная часть после инициализации.

С помощью метода `attachMedia(element)` можно прикрепить элемент `<video />` к плееру отдельно, после инициализации плеера. Также можно передать *MediaElement* через параметры плеера, тогда он прикрепится автоматически.

Для управления плеером через *MediaElement* вам пригодятся события: `onMediaAttached`, `onPause`, `onResume`.

С помощью события `onMediaAttached` можно точно знать, когда плеер присоединен к `<video />` и готов к началу проигрывания. Пример использования:

```
onMediaAttached: () => {
      element.play()
    },
```

Плеер слушает нативные события HTTP элемента `<video />` (в котором Вы передаете плеер на веб-страницу), такие как `play/pause/unmute` и может реагировать на них. С помощью `onPause` и `onResume` можно реагировать на события паузы и восстановления проигрывания. Например, можно рисовать элементы интерфейса (большой значок паузы).

6.3.5 Публикация для большой аудитории с низкой задержкой

LL-HLS подходит для вещания с низкой задержкой на широкую аудиторию. Типичные сценарии:

- Трансляция спортивного матча: зрителей тысячи, они не готовы слышать ”гол!” позже, чем соседи.
- Онлайн конференции, большие вебинары: зрителей много, они взаимодействуют через чат и задержка может помешать общению с аудиторией.

Для таких задач мы предлагаем LL-HLS: он позволит построить эффективный CDN, сохранив при этом приемлемую задержку.

Подготовка потока

Перед тем как проигрывать, настройте публикацию в Flussonic. Чаще всего публикация делается:

- Из программ вроде OBS. Вот статья, как это сделать.
- Из браузера, с веб-камеры или презентация экрана. На странице Публикация по WebRTC мы рассказали, как это настроить.

Подготовка сервера

LL-HLS требует HTTP/2, это значит, что на сервере нужно настроить HTTPS и плееру отдавать уже `https://` ссылку. Некоторые плееры игнорируют это требование, но мы рекомендуем не пропускать это:

- Safari точно не будет работать без https. Он переключится в режим обычного HLS и низкой задержки не будет.
- HTTP/2 позволяет мультиплексировать запросы в одном соединение. Это дает лучший пользовательский опыт: видео реже будет буферизироваться.

Вот статья, которая поможет вам быстро настроить https.

Проигрывание с низкой задержкой по LL-HLS

Поток LL-HLS можно проиграть:

- В каком-либо стороннем плеере с поддержкой LL-HLS. Ссылку для проигрывания LL-HLS можно скопировать на вкладке **Output** в профиле потока.

- В нашем плеере `embed.html`, если добавить в URL параметры `realtime=true&proto=ll-hls`. Откройте эту ссылку в браузере на телефоне или компьютере:

```
https://FLUSSONIC-IP/STREAMNAME/embed.html?realtime=true&proto=ll-hls
```

6.3.6 Проведение онлайн-трансляции на тысячи сотрудников

Когда вы проводите закрытую трансляцию только для сотрудников или партнеров компании в разных городах и странах:

- **Используйте собственные или арендованные серверы**, чтобы видео не попало на внешние платформы, такие как YouTube или Vimeo.
- **Сделайте один или несколько edge-серверов** в каждом филиале в зависимости от количества зрителей, чтобы не нужен был широкий интернет-канал на origin-сервере,.
- **Адаптируйте видео для различных устройств**, чтобы проигрывать без проблем даже если экран маленький, а соединение плохое.
- **Включите запись для функции перемотки**, чтобы опоздавшие могли посмотреть трансляцию с начала.

На этой странице расскажем, как сделать все это с помощью Flussonic Media Server.

Захватите сигнал

- на ваш сервер.
- .
- **Опубликуйте видео в Flussonic** из таких программ как OBS Studio или vMix по любому протоколу: RTMP, WebRTC WHIP, SRT и т.п.

На этом же сервере (origin), где выполняется захват, настройте транскодирование и запись. Использование одного сервера для этих функций возможно, потому что для корпоративной онлайн трансляции вы захватываете всего один или несколько потоков — захват, транскодирование и запись не потребуют много ресурсов.

Адаптируйте видео для различных устройств

Адаптация видео заключается в транскодировании в мультибитрейт, который позволяет подстраиваться под разрешение и канал зрителя, например, при ухудшении связи проигрывать видео в низком качестве. Транскодируйте видео в мультибитрейт с помощью Flussonic Media Server:

- , поскольку аппаратная поддержка гарантирует оптимальное использование ресурсов сервера.
- **Арендуйте сервер с GPU или QSV** в облаке или используйте собственный сервер.

Сделайте запись трансляции

Запись позволит зрителям использовать функций перемотки. Также запись может быть нормативным требованием, например, если вы записываете собрание акционеров.

- на origin-сервере.

Арендуйте серверы для рестриминга

Один origin-сервер не сможет раздать видео на все десятки тысяч зрителей. Используйте собственные серверы в ваших филиалах или арендуйте edge-серверы в регионах вещания:

- **Спланируйте инфраструктуру**. Рассчитывайте примерно по 3-5 серверов на 5000 зрителей. Чтобы точно понять, сколько и каких edge-серверов вам нужно, обратитесь в наш отдел продаж и технической поддержки по адресу support@flussonic.com.
- на эти edge-серверы.
- с origin на edge.
- архива. На edge-серверах должен быть SSD для кэширования, чтобы снизить нагрузку на DVR на origin-сервере.

Настройте балансировку

Чтобы дать единую ссылку для проигрывания во всех регионах и при этом предотвратить перегрузку какого-либо одного сервера, направив пользователей на ближайший доступный сервер:

- , чтобы распределять запросы на проигрывание между несколькими серверами в одном регионе.
- , чтобы edge-серверы обрабатывали запросы на проигрывание только из своего региона.

Дайте зрителям ссылки для проигрывания

Видео из Flussonic можно проиграть на смартфонах и компьютерах, и даже отправить в кабельную сеть филиала.

- на своем веб-сайте для проигрывания в браузере на смартфонах и компьютерах.
- по различным протоколам, в том числе популярным HLS/DASH, или по протоколам с низкой задержкой MSE-LD, WebRTC, LL-HLS.

Chapter 7

VOD services

7.1 VOD services

7.1.1 Как посмотреть файл?

Как посмотреть файл

Задача: имеется видеофайл, нужно организовать вещание этого файла по сети.

На этой странице:

Установка Flussonic

Первым шагом надо установить Flussonic.

Подготовка файла: правильный формат

Flussonic умеет проигрывать только файлы в определенных форматах. Основной формат контейнера — mp4, кодек видео — h264, кодек аудио — aac.

Чтобы иметь возможность проверить и при необходимости изменить формат файла, на сервер нужно установить ffmpeg. Устанавливать его не обязательно на сам сервер, можно и на другой компьютер с операционными системами Linux, Windows или OSX.

Инструкции по скачиванию ffmpeg находятся на официальном сайте: https://www.ffmpeg.org/download.html.

Во многих дистрибутивах GNU/Linux ffmpeg уже есть в стандартных репозиториях. Например, в Ubuntu начиная с версии 15.04 Vivid Vervet достаточно написать в командной строке "apt-get install ffmpeg". Если у вас этого пакета нет, стоит поискать сторонние репозитории, или просто скачать статическую сборку.

Для определения формата будем использовать команду ffprobe, входящую в комплект только что установленного ffmpeg. В командной строке пишем `ffprobe /путь/к/вашему/видео/video.mp4`. Нужно указать правильный путь до видеофайла.

Вот как должен выглядеть вывод ffprobe для правильного файла:

```
ffprobe version N-61916-g46f72ea Copyright (c) 2007-2014 the FFmpeg
   developers

Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'video.mp4':
  Duration: 00:05:00.18, start: 0.012000, bitrate: 769 kb/s
    Chapter #0.0: start 0.000000, end 300.000000
    Metadata:
      title           : Chapter 1
    Stream #0:0(eng): Video: h264 (High) (avc1 / 0x31637661), yuv420p, 640
   x360 [SAR 1331:1000 DAR 2662:1125], 636 kb/s, 23.98 fps, 23.98 tbr, 24k
   tbn, 47.95 tbc (default)
    Metadata:
      handler_name    : VideoHandler
    Stream #0:1(rus): Audio: aac (mp4a / 0x6134706D), 48000 Hz, stereo,
   fltp, 128 kb/s (default)
    Metadata:
      handler_name    : SoundHandler
    Stream #0:2(eng): Subtitle: mov_text (text / 0x74786574)
    Metadata:
      handler_name    : SubtitleHandler
```

В нем мы видим несколько дорожек, причем `Video: h264` означает использование правильного кодека h264, а `Audio: aac` — правильного кодека aac.

Вот как выглядит вывод для неправильного файла:

```
Input #0, asf, from 'video.wmv':
  Duration: 00:05:00.22, start: 0.000000, bitrate: 388 kb/s
    Chapter #0.0: start 0.000000, end 300.217000
    Metadata:
      title           : Chapter 1
    Stream #0:0: Video: msmpeg4v3 (MP43 / 0x3334504D), yuv420p, 640x360,
   23.98 tbr, 1k tbn, 1k tbc
    Stream #0:1: Audio: wmav2 (a[1][0][0] / 0x0161), 48000 Hz, 2 channels,
   fltp, 128 kb/s
```

Здесь мы видим, что использован формат Windows Media (wmv) с соответствующими кодеками. Flussonic не станет его воспроизводить.

Что же делать, если формат неправильный? Чтобы превратить wmv/msmpeg/wma в mp4/h264/aac, можно воспользоваться программой ffmpeg:

```
ffprobe путьквашемуизначальномувидео//////video.wmv
   путьквашемуизмененномувидео//////video.mp4
```

На экране появится что-то вроде:

```
Stream mapping:
```

```
  Stream #0:0 -> #0:0 (msmpeg4 -> libx264)
  Stream #0:1 -> #0:1 (wmav2 -> libvo_aacenc)
Press [q] to stop, [?] for help
frame=  937 fps=180 q=-1.0 Lsize=    2320kB time=00:00:39.04 bitrate= 486.7
   kbits/s dup=1 drop=0
```

Этот процесс называется транскодирование, и может занимать очень много времени и ресурсов компьютера. Чем мощнее ваше оборудование, тем быстрее происходит транскодирование.

По этой же причине Flussonic не транскодирует файлы автоматически, предполагается что пользователи будут делать это вручную на выделенном оборудовании. Кстати, прямой эфир нельзя транскодировать заранее, поэтому Flussonic предлагает для стримов встроенный транскодер.

В результате всех вышеописанных операций мы получили файл, полностью подходящий для проигрывания с помощью Flussonic.

Подготовка файла: качество изображения

Возможно, вам захочется понизить качество или сделать мультибитрейтный файл (что обеспечит комфортный просмотр видео пользователям, подключенным на разных скоростях к интернету).

Подробности в разделе про транскодирование файлов.

Настройка Flussonic

Чтобы Flussonic начал обслуживать файлы, нужно добавить в файл конфигурации (`/etc/flussonic/flusso`
специальную настройку:

```
vod vod {
  storage /movies;
  download;
}
```

По-английски эта настройка называется "VOD location" (этот термин мы используем в документации и в технической поддержке).

`/movies` — это директория на диске сервера, в которой будут храниться видеофайлы.

Кстати, эта директория технически может находиться на NFS-шаре, но это не очень хорошее решение, т.к. NFS работает медленно и не всегда хорошо. Лучше использовать локальный жесткий диск или SSD.

После того, как добавили эту настройку, не забудьте применить конфигурацию с помощью `service flussonic reload`

Загрузка файла на сервер

Чтобы загрузить файл на сервер, можно воспользоваться веб-интерфейсом. В главном меню выберите пункты **Media** -> **VODs** -> название созданной локации -> **browse**. На открывшейся странице вы можете добавлять файлы в VOD-локацию, создавать вложенные директории для упорядочивания файлов, а также просматривать файлы и получать ссылки для проигрывания VOD-файлов.

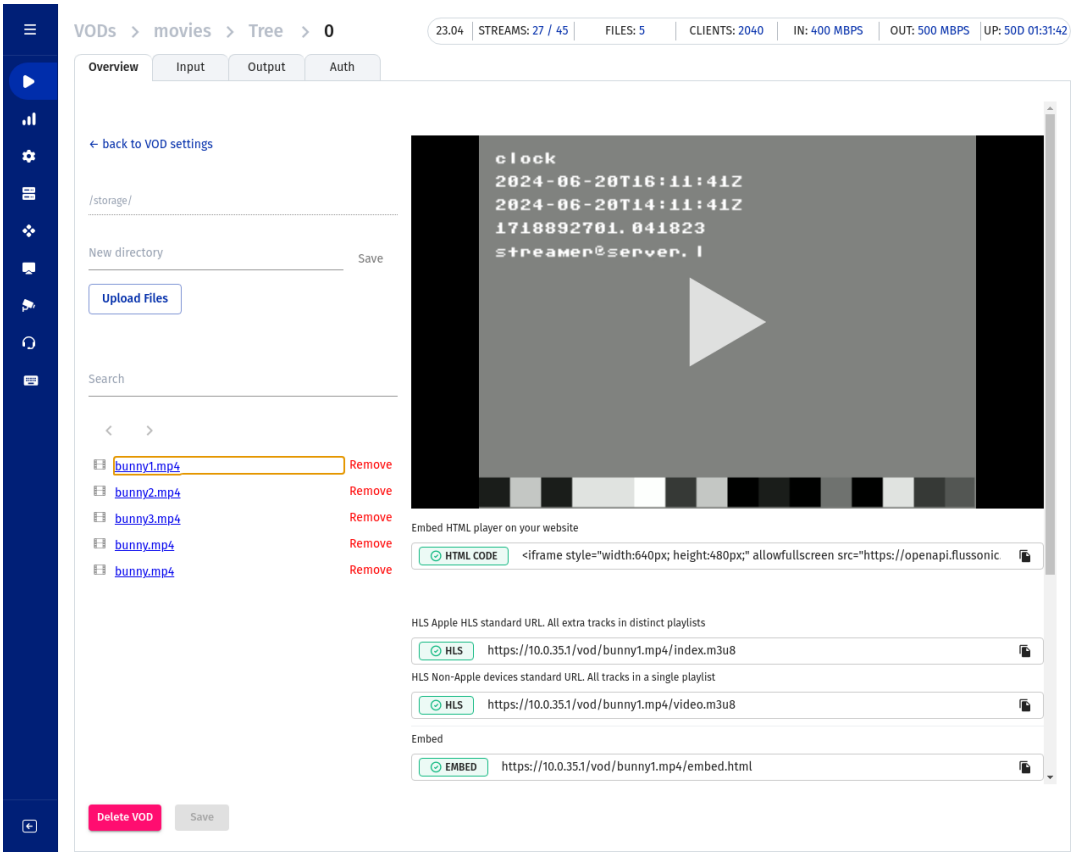


Figure 7.1: Flussonic VOD

Важно отметить, что веб-интерфейс не является единственным способом загрузки файла. Можно залить файл с помощью SSH или FTP, или любого другого способа передачи файлов по сети. Главное, чтобы файл оказался внутри директории, которая указана в файле конфигурации. Flussonic выполняется с правами root, т.е. имеет доступ к любым файлам, поэтому никаких специальных прав доступа этому загруженному файлу назначать не нужно.

Просмотр файла

В веб интерфейсе (в разделе Media->VOD) щелкните мышкой по нужному файлу - начнется его воспроизведение.

Прямо под видео написан его URL, который можно использовать для просмотра вне веб-

интерфейса. Он должен выглядеть примерно так: `http://erlyvideo:80/vod/elementary/s01e02.mp` URL заканчивается на `.m3u8` — это значит, что для воспроизведения будет использоваться протокол HLS.

Смотреть такой URL можно в любом плеере, хорошо поддерживающем HLS. Например, можно скачать и запустить видеоплеер VLC, в главном меню выбрать **Media > Open Network Stream**, или в русской версии ”Медиа->Открыть URL”, или нажать клавиатурную комбинацию Ctrl+N, и скопировать этот URL в поле ввода.

Дополнительные действия

Прочитайте документацию, посвященную описанию работы VOD Там есть ответы на пару вопросов, которые не освещены в этой статье.

7.1.2 Подготовка мультибитрейтных файлов

Для того, чтобы обеспечить комфортный просмотр видео пользователям, подключенным на разных скоростях к интернету, можно воспользоваться адаптивным стримингом. Для этого надо сделать мультибитрейтный MP4 файл и запросить для него манифест. Дальнейшее Flussonic Media Server сделает сам.

Ниже мы рассмотрим, как настроить компьютер и создать мультибитрейтный файл.

Установка программ

Вам нужно иметь установленный ffmpeg и кодеки. Процесс установки отличается для разных операционных систем.

Инструкция для Windows

- Скачайте ffmpeg: https://ffmpeg.org/download.html
- Следуйте инструкции по установке.
- Скачайте и установите K-Lite Mega Codec Pack: http://www.codecguide.com/download_k-lite_codec_pack_mega.htm. После запуска инсталлятора будет предложено несколько вариантов установки, нужно выбрать самый полный («Lots of stuff»).

Инструкция для Linux Мы рекомендуем поставить уже собранный ffmpeg отсюда: http://johnvansickle.com/ff
Или любой другой собранный ffmpeg с официального сайта: https://www.ffmpeg.org/download.html

С большой вероятностью ffmpeg, идущий в вашем дистрибутиве либо не будет уметь кодировать H264, либо будет слишком старым, чтобы подошли наши инструкции (вообще любые инструкции в интернете, которые основываются на возможностях свежих версий ffmpeg), или произойдет еще какая-нибудь другая неприятность.

Конструирование команды для ffmpeg на примере создания multi-bitrate потока

В данной статье мы рассмотрим, как создать multi-bitrate поток с использованием FFmpeg. В качестве примера возьмем видеофайл `hall.mp4`, который мы перекодируем в несколько вариантов с различными битрейтами и разрешениями, что особенно полезно для адаптивного стриминга.

Анализ исходного файла Перед началом кодирования рекомендуется проверить содержимое исходного видеофайла:

```
ffmpeg -i hall.mp4
```

Подготовка к кодированию Мы будем кодировать видео в пять различных разрешений с фиксированными битрейтами. Используем двухпроходное кодирование (two-pass encoding) для улучшения качества финального видео.

Первый проход кодирования В первом проходе FFmpeg анализирует видео и собирает статистику для оптимального распределения битрейта:

```
ffmpeg -y -i hall.mp4 \
    -pass 1 \
    -map 0:0 -map 0:0 -map 0:0 -map 0:0 -map 0:0 -map 0:1 \
    -c:v libx264 -sc_threshold 0 -x264-params "nal-hrd=cbr" -g 60 -
   b_strategy 0 -forced-idr 1 \
    -c:a libfdk_aac -b:a 160k -ac 2 \
    -b:v:0 200k -maxrate:v:0 200k -minrate:v:0 200k -bufsize:v:0 200k -
   filter:v:0 scale=-2:240 \
    -b:v:1 500k -maxrate:v:1 500k -minrate:v:1 500k -bufsize:v:1 500k -
   filter:v:1 scale=-2:360 \
    -b:v:2 1000k -maxrate:v:2 1000k -minrate:v:2 1000k -bufsize:v:2 1000k -
   filter:v:2 scale=-2:480 \
    -b:v:3 3000k -maxrate:v:3 3000k -minrate:v:3 3000k -bufsize:v:3 3000k -
   filter:v:3 scale=-2:720 \
    -b:v:4 4000k -maxrate:v:4 4000k -minrate:v:4 4000k -bufsize:v:4 4000k -
   filter:v:4 scale=-2:1080 \
    -f mp4 /dev/null
```

- `-y` — автоматическое подтверждение перезаписи файлов.
- `-pass 1` — первый проход кодирования (сбор статистики).
- `-map 0:0 -map 0:0 -map 0:0 -map 0:0 -map 0:0 -map 0:1` — выбор входных потоков: 5 видеопотоков и 1 аудиопоток.
- `-c:v libx264` — кодирование видео с использованием кодека H.264.
- `-g 60` — установка размера GOP (группы кадров) в 60 кадров.
- `-b:v:N Xk` — битрейт каждого видеопотока (от 200k до 4000k).
- `-filter:v:N scale=-2:Y` — изменение разрешения каждого видеопотока (240p, 360p, 480p, 720p, 1080p).
- `-c:a libfdk_aac -b:a 160k -ac 2` — кодирование аудиопотока в AAC с битрейтом 160k и стереоканалом.
- `-f mp4 /dev/null` — в первом проходе выходной файл не создается, статистика записывается во временные файлы.

Второй проход кодирования

```
ffmpeg -y -i hall.mp4 \
    -pass 2 \
    -map 0:0 -map 0:0 -map 0:0 -map 0:0 -map 0:0 -map 0:1 \
    -c:v libx264 -sc_threshold 0 -x264-params "nal-hrd=cbr" -g 60 -
   b_strategy 0 -forced-idr 1 \
    -c:a libfdk_aac -b:a 160k -ac 2 \
    -b:v:0 200k -maxrate:v:0 200k -minrate:v:0 200k -bufsize:v:0 200k -
   filter:v:0 scale=-2:240 \
    -b:v:1 500k -maxrate:v:1 500k -minrate:v:1 500k -bufsize:v:1 500k -
   filter:v:1 scale=-2:360 \
    -b:v:2 1000k -maxrate:v:2 1000k -minrate:v:2 1000k -bufsize:v:2 1000k -
   filter:v:2 scale=-2:480 \
    -b:v:3 3000k -maxrate:v:3 3000k -minrate:v:3 3000k -bufsize:v:3 3000k -
   filter:v:3 scale=-2:720 \
    -b:v:4 4000k -maxrate:v:4 4000k -minrate:v:4 4000k -bufsize:v:4 4000k -
   filter:v:4 scale=-2:1080 \
    -f mp4 hall_mbr.mp4
```

- `-pass 2` — второй проход кодирования.
- Выходной файл `hall_mbr.mp4` создается с несколькими видеопотоками разного качества.

Кодирование с использованием NVIDIA (GPU) Пример использования аппаратного кодировщика NVIDIA (h264_nvenc):

```
ffmpeg -analyzeduration 11M -hwaccel cuvid -c:v h264_cuvid -vsync 2 -
   avoid_negative_ts disabled -i hall.mp4 -err_detect ignore_err -loglevel
   repeat+level+verbose -spatial-aq 1 -aq-strength 8 -y  \
    -map 0:v:0 -map 0:v:0 -map 0:v:0 -map 0:v:0 -map 0:v:0 -map 0:a:0 \
    -profile:v high -preset:v slow -threads 0 \
    -c:v h264_nvenc -cbr 1 -rc cbr_hq -2pass 1 -bf 2 -b_ref_mode 2 -g 50 -
   b_adapt 0 -no-scenecut 1 -forced-idr 1 -strict_gop 1 \
    -c:a libfdk_aac -b:a 128k \
    -b:v:0 650k -maxrate:v:0 700k -minrate:v:0 650k -bufsize:v:0 700k -
   filter:v:0 "scale_cuda=640:360" -trellis 1 \
    -b:v:1 1000k -maxrate:v:1 1100k -minrate:v:1 1000k -bufsize:v:1 1000k -
   filter:v:1 "scale_cuda=768:432" -trellis 1 \
    -b:v:2 2250k -maxrate:v:2 2350k -minrate:v:2 2250k -bufsize:v:2 2250k -
   filter:v:2 "scale_cuda=960:540" -trellis 1 \
    -b:v:3 3000k -maxrate:v:3 3100k -minrate:v:3 3000k -bufsize:v:3 3M -
   filter:v:3 "scale_cuda=-1:720" -trellis 1 \
    -b:v:4 5000k -maxrate:v:4 5100k -minrate:v:4 5000k -bufsize:v:4 5M -
   filter:v:4 "scale_cuda=-1:1080" -trellis 1 \
    -f mp4 "hall_nvenc_mbr.mp4" \
    -max_muxing_queue_size 1024
```

- `-hwaccel cuvid -c:v h264_cuvid` – использование аппаратного декодера NVIDIA для декодирования входного видео.

- `-c:v h264_nvenc` – использование кодировщика NVIDIA NVENC для кодирования видео.
- `-profile:v high` – задание профиля High для улучшения качества кодирования.
- `-preset:v slow` – баланс между скоростью кодирования и качеством выходного файла.
- `-cbr 1 -rc cbr_hq` – использование режима постоянного битрейта (CBR) с высоким качеством.
- `-2pass 1` – двухпроходное кодирование для улучшения качества сжатия.
- `-bf 2 -b_ref_mode 2` – настройка би-директорных кадров для более эффективного кодирования.
- `-g 50` – установка размера группы кадров (GOP) в 50 кадров.
- `-b_adapt 0 -no-scenecut 1 -forced-idr 1 -strict_gop 1` – управление структурой GOP и сценами резкого перехода.
- `-b:v:N Xk -maxrate:v:N Xk -minrate:v:N Xk -bufsize:v:N Xk` – задание битрейта, буферизации и ограничений для каждого качества видео.
- `-filter:v:N "scale_cuda=W:H"` – масштабирование с помощью аппаратного ускорения CUDA.
- `-c:a libfdk_aac -b:a 128k` – кодирование аудиопотока с использованием AAC (libfdk_aac) с битрейтом 128 кбит/с.
- `-max_muxing_queue_size 1024` – увеличение очереди мультиплексирования для предотвращения ошибок при записи в MP4.

Кодирование участка видео

Иногда нужно перекодировать не всё видео, а только какой-то его участок. Для этого используется следующий параметр: `-ss 00:00:00 -t 00:05:00`. Здесь первая цифра показывает, с какой секунды должен начинаться кодируемый фрагмент, а вторая цифра — его продолжительность.

```
ffmpeg -analyzeduration 11M -hwaccel cuvid -c:v h264_cuvid -vsync 2 -
   avoid_negative_ts disabled -i hall.mp4 -ss 00:00:00 -t 00:05:00 -
   err_detect ignore_err -loglevel repeat+level+verbose -spatial-aq 1 -aq-
   strength 8 -y  \
    -map 0:v:0 -map 0:v:0 -map 0:v:0 -map 0:v:0 -map 0:v:0 -map 0:a:0 \
    -profile:v high -preset:v slow -threads 0 \
    -c:v h264_nvenc -cbr 1 -rc cbr_hq -2pass 1 -bf 2 -b_ref_mode 2 -g 50 -
   b_adapt 0 -no-scenecut 1 -forced-idr 1 -strict_gop 1 \
    -c:a libfdk_aac -b:a 128k \
    -b:v:0 650k -maxrate:v:0 700k -minrate:v:0 650k -bufsize:v:0 700k -
   filter:v:0 "scale_cuda=640:360" -trellis 1 \
    -b:v:1 1000k -maxrate:v:1 1100k -minrate:v:1 1000k -bufsize:v:1 1000k -
   filter:v:1 "scale_cuda=768:432" -trellis 1 \
    -b:v:2 2250k -maxrate:v:2 2350k -minrate:v:2 2250k -bufsize:v:2 2250k -
   filter:v:2 "scale_cuda=960:540" -trellis 1 \
```

```
 -b:v:3 3000k -maxrate:v:3 3100k -minrate:v:3 3000k -bufsize:v:3 3M -
filter:v:3 "scale_cuda=-1:720" -trellis 1 \
 -b:v:4 5000k -maxrate:v:4 5100k -minrate:v:4 5000k -bufsize:v:4 5M -
filter:v:4 "scale_cuda=-1:1080" -trellis 1 \
 -f mp4 "hall_nvenc_mbr.mp4" \
 -max_muxing_queue_size 1024
```

Это кодирование из предыдущей части, но сделанное только для первых 5 секунд фильма.

Конвертация файлов для потоковой передачи в Интернете

Flussonic поддерживает следующие Контейнеры и кодеки. Если ваш видеофайл закодирован в другой формат, он не будет воспроизводиться через Flussonic. Например, могут быть старые кодеки, такие как Xvid или MPEG4-Video, которые не поддерживаются новыми версиями браузеров. В таких случаях необходимо транскодировать файл.

Чтобы преобразовать файл любого формата в H.264, в командной строке перейдите в директорию с файлом, затем выполните команду:

```
ffmpeg -i input_file.avi -c:v libx264 -g 100 -c:a aac -f mp4 output_file.
   mp4
```

Аналогично можно преобразовать файл любого формата в H.265/HEVC:

```
ffmpeg -i input_file.avi -c:v libx265 -g 100 -c:a aac -f mp4 output_file.
   mp4
```

7.1.3 Мультибитрейтное проигрывание из нескольких файлов через SMIL

Если у вас есть несколько файлов с одинаковым контентом, но с разным битрейтом, вы можете использовать для мультибитрейтного проигрывания SMIL файлы.

SMIL файл — это файл в формате XML, который позволяет составлять плейлисты для разных комбинаций VOD файлов с разным битрейтом. Он работает так же, как и функциональность `auto-mbr`, описанная здесь, но дает больше гибкости: можно включить в плейлист не все файлы из папки, а лишь некоторые. Кроме того, не существует никаких правил именования файлов, необходимо лишь поместить файлы в папку, в которой находится SMIL файл, или в ее подпапки.

Вы можете использовать SMIL файлы как на локальном компьютере, так и в хранилище Amazon S3. В примере ниже мы используем локальную VOD-локацию. Если вы используете хранилище Amazon S3, выполняйте такие же шаги, но в настройках VOD-локации укажите URL хранилища, как указано здесь.

Настройка VOD-локации

Предположим, что вы уже создали VOD-локацию.

В созданную VOD-локацию добавьте опцию `auto_mbr`.

* Через файл:

```
vod vod1 {
  storage /storage;
  auto_mbr;
}
```

- Через UI:

Откройте **Files (VOD)** > зайдите в VOD-локацию > на вкладке **Output** отметьте **Enable MBR from multiple files**.

Подготовка файлов

Поготовьте файлы с одинаковым контентом, но разным битрейтом, например: `bunny_450`, `bunny_750`, `bunny_1100`. Имена файлов могут быть любыми.

Поместите файлы в одну папку в VOD-локации. Некоторые файлы можно поместить в подпапки (например, вы можете поместить файл `bunny_110` в подпапку `folder`).

Создание SMIL-файла

С помощью текстового редактора создайте файл `*.smil` (например, `my.smil`) в той же папке, где находятся видеофайлы. Структура SMIL файла должна быть следующей:

```
<?xml version="1.0" encoding="UTF-8"?>
<smil title="">
    <body>
        <switch>
            <video height="240" src="bunny_450.mp4" systemLanguage="eng"
   width="424">
                <param name="videoBitrate" value="450000" valuetype="data
   "></param>
                <param name="audioBitrate" value="44100" valuetype="data
   "></param>
            </video>
            <video height="360" src="bunny_750.mp4" systemLanguage="eng"
   width="640">
                <param name="videoBitrate" value="750000" valuetype="data
   "></param>
                <param name="audioBitrate" value="44100" valuetype="data
   "></param>
            </video>
            <video height="720" src="folder/bunny_1100.mp4" systemLanguage
   ="eng" width="1272">
                <param name="videoBitrate" value="1100000" valuetype="data
   "></param>
                <param name="audioBitrate" value="44100" valuetype="data
   "></param>
            </video>
        </switch>
    </body>
</smil>
```

Единственный обязательный и значимый параметр для каждого файла: `src` — относительный путь к файлу в папке, где находится SMIL файл. Flussonic игнорирует все остальные параметры и определяет размер, язык и битрейт каждого видеофайла автоматически.

Вы можете создать любое количество SMIL файлов в одной папке, чтобы подготовить плейлисты для разных комбинаций файлов.

Проигрывание плейлиста

Ниже описано, как проиграть плейлист.

- HLS плейлист:

```
http://FLUSSONIC-IP/vod1/my.smil/index.m3u8
```

- DASH плейлист:

```
http://FLUSSONIC-IP/vod1/my.smil/index.mpd
```

Flussonic создает плейлист HLS или DASH из файлов, перечисленных в SMIL файле. Плеер проигрывает этот плейлист как один мультибитрейтный файл.

7.1.4 Стриминг файлов из облачного хранилища

При организации сервиса Video on Demand (VoD) с заранее неизвестным объемом библиотеки файлов и неизвестным количеством зрителей мы рекомендуем использовать облачное хранилище, например, Amazon S3/AWS, OpenStack Swift, Ceph и т.п. Холодное хранение в облаке может быть дешевле равномерного дискового хранилища, кроме того облачное хранилище лучше масштабируется при росте сервиса.

С помощью *Flussonic Media Server* вы можете транслировать видеофайлы, которые находятся на облачных хранилищах или на HTTP серверах.

Настройка через UI

Для организации надежного вещания с облачного хранилища необходимо выполнить следующие шаги:

- Подготовьте инсталляцию *Flussonic Media Server* с локальным диском достаточного большого размера, чтобы туда мог поместиться кеш горячего контента. В качестве стартового размера можно взять, например, 256GB SSD.
- Соберите ключи доступа к хранилищу. Например, для Amazon S3 надо знать ACCESS_KEY и SECRET_KEY.
- Создайте VOD-локацию в *Flussonic Media Server* на странице **Media -> VODs**, указав адрес хранилища, например:

Можно передать любые параметры в query string. При обращению к файлу `http://FLUSSONIC-IP/vod/bu`
запрос к хранилищу превратится в `http://storage/prefix/bunny.mp4?key=12345`.

- Настройте локальный файловый кеш для этой локации на вкладке **Input**.

- Затем нажмите **Browse** рядом с адресом хранилища на вкладке **Overview** и проверьте, что файлы проигрываются.

Настройка VOD через конфигурационный файл

Те же самые настройки можно сделать и через конфигурационный файл `/etc/flussonic/flussonic.con`
или в редакторе **Config editor** в UI. Ниже приведен пример конфигурации VOD.

Обратите внимание на использование кэша, чтобы не было частых запросов к облаку.

```
vod public {
  storage http://s3.amazonaws.com/publicbucket;
  cache /cache 100G;
```

Figure 7.2: Настройка VOD из облака

```
}
vod private {
  storage s3://ACCESS_KEY:SECRET_KEY@s3.amazonaws.com/privatebucket;
  cache /cache 100G;
}
```

Связанные задачи

- Как посмотреть файл: о стриминге из дискового хранилища.

Figure 7.3: VOD cache

7.1.5 Мультибитрейтный плейлист из файлов

Создание мультибитрейтного содержимого из нескольких файлов

Пусть у вас есть копии фильма в нескольких файлах с разным качеством, и вы не хотите создавать из них один мультибитрейтный файл. Теперь можно проигрывать все эти файлы, используя один HLS или DASH-плейлист. Клиентский плеер сможет выбирать битрейт точно так же, как в ситуации с одним мультибитрейтным файлом.

Flussonic Media Server может отдавать несколько файлов с разным битрейтом как один мультибитрейтный ресурс. HLS или DASH-плейлисты в этом случае содержат информацию об этих файлах как о разном качестве одного файла.

Вам нужно подготовить файлы и включить автоматическое создание мультибитрейтного ресурса для VOD-локации.

Вы можете использовать VOD-локацию как на локальном компьютере, так и в хранилище Amazon S3. В примере ниже мы используем локальную VOD-локацию. Если вы используете хранилище Amazon S3, выполняйте такие же шаги, но в настройках VOD-локации укажите URL хранилища, как указано здесь.

Подготовка файлов Поместите файлы в одну директорию и назовите их так, чтобы имя файла начиналось как имя директории, в которой они находятся. То есть имена файлов должны соответствовать маске `DIR_NAME*.mp4`, где `*` — любые допустимые символы. Например:

Имя директории: **DIR_NAME**, имена файлов: **DIR_NAME-1.mp4**, **DIR_NAMEabc.mp4** и т.д.

См. **Шаг 2** ниже.

Включение создания мультибитрейтного ресурса Предположим, что вы уже создали VOD-локацию для обращения к файлам, которые будут использоваться для создания мультибитрейтного плейлиста.

Шаг 1. В созданную VOD локацию добавьте опцию `auto_mbr`.

- Через файл:

```
vod vod1 {
  storage /storage;
  auto_mbr;
}
```

- Через UI:

Откройте **Files (VOD)** > зайдите в VOD-локацию > на вкладке **Output** отметьте **Enable MBR from multiple files**.

Шаг 2. Поместите в директорию файлы, например, такие:

`/storage/movies/bunny/bunny.480x360.mp4`

`/storage/movies/bunny/bunny.720x480.mp4`

`/storage/movies/bunny/bunny.1080x720.mp4`

Размеры видео *Flussonic* определяет сам, поэтому необязательно указывать их в названии файла. После слова `bunny` в имени файла может идти произвольный набор допустимых символов.

Шаг 3. Теперь можно запрашивать плейлист по такому URL:

- для HLS:

```
http://FLUSSONIC-IP/vod1/bunny/index.m3u8
```

- для DASH:

```
http://FLUSSONIC-IP/vod1/bunny/index.mpd
```

Из примера видно, что список воспроизведения запрашивается на директорию, а не на один файл.

При запросе одного из плейлистов на директорию: `/vod/bunny/index.m3u8` или `/vod/bunny/index.mp` *Flussonic* составляет HLS или DASH-плейлист из файлов `/vod/bunny/bunny*.mp4`. Плеер "думает", что это один файл.

> Клиент сможет прочитать содержимое только тех директорий, для которых в конфиге указана опция `auto_mbr`. Иначе *Flussonic* вернет ошибку `404`.

7.1.6 Кэш

Для ускорения раздачи VOD можно использовать кэш.

Для оригинальных файлов из облака или с HTTP сервера используется `cache`.

Для файлов на SSD диске приходится использовать схему с промежуточным SSD кэшированием, т.е. кэшировать сегменты с помощью `segment_cache`.

Файловый кэш на SSD

Если файлы берутся из облака или с простого HTTP сервера (например, другого Flussonic Media Server), то нужно настроить файловый кэш. То есть зону, куда файл будет скачиваться целиком.

Этот механизм позволит выстроить полноценный файловый CDN из нескольких Flussonic Media Server, поскольку даже скачивание файла с вторичного сервера приведет к кэшированию файла на нём.

И, конечно, Flussonic Media Server не будет скачивать одни и те же данные с источника дважды.

> Для кэширования файлов **не** используйте разделы SSD, смонтированные с опцией `noatime`.

Для настройки файлового кэша пропишите в конфигурации:

```
vod vod_remote {
  storage s3://minioadmin:minioadmin@minio:9001/test;
  cache /storage/cache 400G;
  download;
}
```

При такой конфигурации можно проигрывать файлы, скачивая их с указанного сервера. Они будут сохраняться в кэше, причем сохраняться будет только та часть данных, которая была запрошена.

Кэширование по количеству обращений Для файлов можно указать условие для помещения файла в кэш — этим условием будет количество запросов файла клиентами.

Опция **misses=3** говорит серверу, что файл нужно поместить в кэш, если к нему было более трех обращений:

```
vod vod_remote {
  storage s3://minioadmin:minioadmin@minio:9001/test;
  cache /storage/cache 400G misses=3;
  download;
}
```

Настройка кэша в веб-интерфейсе Чтобы выбрать настройки кэширования через Flussonic UI:

- В разделе **VODs** кликните файл, который нужно кэшировать.
- Отредактируйте настройки на вкладке **Input** в секции **Cache**.

Figure 7.4: VOD cache

Сегментный кэш для SSD

На сегодняшний день один из самых главных способов радикально ускорить работу сервера по раздаче контента с дисков — использование SSD накопителей.

Так как твердотельные накопители стоят существенно дороже, чем обычные диски, то приходится использовать схему с промежуточным SSD кэшированием.

Flussonic Media Server умеет самостоятельно кэшировать на диске запрашиваемые отрезки видео по протоколу HLS, что позволяет сильно ускорить раздачу.

Для настройки сегментного кэша пропишите в конфигурации:

```
vod vod1 {
    storage /mount/hdd1;
    storage /mount/hdd2;
    storage /mount/hdd3;
    segment_cache /mount/ssd1 20G 48h misses=2;
}
```

При такой конфигурации Flussonic Media Server будет сам поддерживать заполнение кэша на уровне 20 гигабайт, стирая файлы старше **двух суток** и кэшируя только те файлы, к которым было больше **двух обращений**. Опция `segment_cache` указывается только один раз.

> Мы не рекомендуем использовать для `segment_cache` обычные диски (HDD), используйте SSD.

Chapter 8

Ad Insertion

8.1 Ad Insertion

8.1.1 Настройка врезки рекламы

С Flussonic вы можете настроить серверную врезку рекламы (SSAI) через авторизационный бэкенд. Рекламу можно врезать по расписанию или по меткам SCTE. Подробнее об этих способах см. Способы врезки рекламы на стороне сервера.

> Потоки с рекламой, врезанной с помощью *Flussonic*, проигрываются только по протоколам HLS и DASH.

Шаг 1. Создайте VOD-локацию

Для врезки рекламы нужно использовать только те файлы с рекламой, которые находятся в настроенной VOD-локации на сервере *Flussonic*. Не указывайте пути к внешним ресурсам (HTTP) и напрямую к файловой системе сервера.

Чтобы создать VOD-локацию:

- Перейдите на вкладку **Media** - **VODs** и нажмите **+**.
- Укажите имя локации `ad_vod`.
- Укажите путь на диске или в облаке, где будете хранить файлы с рекламой. В примере мы добавим путь `/opt/flussonic/priv`, и там уже есть ролик `bunny.mp4`, который мы будем использовать далее в этой инструкции. Когда вы будете использовать реальные ролики, добавляйте их в свою директорию VOD.

Шаг 2. Подготовьте рекламные ролики

Перед врезкой рекламные ролики необходимо подготовить. Это можно сделать в любом видеоредакторе или с помощью API Flussonic file_processor.

Во внешнем видеоредакторе При подготовке роликов во внешнем ПО учитывайте следующие требования:

- **Характеристики** (`media_info`) основного и рекламного потоков должны совпадать. Параметры рекламного потока **должны быть идентичны** параметрам основного потока, иначе *Flussonic* проигнорирует рекламный ролик и не врежет его.
- Значение GOP рекламного ролика должно быть равно **одной** секунде.
- Первые 1-5 секунд преролла могут пропускаться, из-за особенности работы большинства HLS-плееров. В таком случае вставьте несколько чёрных кадров в начало ролика.

Figure 8.1: Flussonic VOD

С помощью API Flussonic file_processor Сначала нужно получить `media_info` потока, в который будете врезать рекламу. Используйте запрос API GET /name/info.json:

```
curl http://user:password@localhost/streamer/api/v3/streams/demo | jq .
    stats.media_info >> fp.json
```

Заменяйте логин и пароль в командах `curl` на имя пользователя и пароль, которые используете для входа в интерфейс администратора.

Результат будет записан в файл `fp.json`. Откройте его для редактирования любым текстовым редактором и дополните следующим образом:

```
{
```

```
  "input_files": ["/opt/flussonic/priv/bunny.mp4"],
  "output_file": "bunny_fp",
  "filters": [{
    "group": "transcoder",
    "transcoder":

---
your media_info
---

  }]
}
```

Создайте `file_processor`:

```
curl -X PUT http://user:password@localhost/streamer/api/v3/file_processor -
   d '{"path":"ad_vod"}' -H "Content-Type: a
pplication/json"
```

Теперь можно передать файл `fp.json` в метод POST /file_processor/jobs:

```
curl -X POST http://user:password@localhost/streamer/api/v3/file_processor/
   jobs --data @fp.json --header "Content-Type: application/json"
```

В выводе последней команды найдите значение параметра `output_file`, оно будет иметь вид `bunny_fp.processed.1711966939428660.mp4`. Это название подготовленного рекламного ролика, который вам нужно указать в авторизационном бэкенде. Файл уже находится в VOD-локации, которую вы указали для `file_processor`.

> Подготовка может занять некоторое время. Сначала на диске будет создан файл с расширением .tmp. Не удаляйте и не перемещайте его. Дождитесь, пока file_processor закончит работу и появится итоговый файл.

Шаг 3. Авторизационный бэкенд для врезки рекламы

Чтобы использовать врезку рекламы SSAI, реализуйте авторизационный бэкенд. Ваш авторизационный бекенд должен вернуть структуру JSON, описанную в схеме API.

Как сделать авторизационный бэкенд:

- Ниже приведены примеры JSON для врезки рекламы по SSAI двумя способами. Поскольку в потоке demo, который мы создадим на следующем шаге, нет меток SCTE, то используйте вариант с врезкой по расписанию.

!!! note Эти структуры JSON не выполняют авторизацию как таковую, т.к. не содержат логики проверки токенов. В таком виде их можно примерять только для тестирования врезки рекламы.

Чтобы действительно выполнять авторизацию сессий проигрывания, следуйте инструкции на станице Авторизация.

- На вкладке **Media** - **Streams** в Flussonic Admin UI создайте поток `demo` с источником `fake://fake`
- Откройте вкладку **Auth** в профиле потока, выберите **Authentication type** -> **custom** и укажите адрес авторизационного бэкенда `http://localhost/ad_backend.json`.

Врезка рекламы по SSAI по расписанию

```
{
    "ad_inject": {
        "v": 2,
        "preroll": ["ad_vod/bunny_fp.processed.1711966939428660.mp4"],
        "midroll_interval": 180,
        "midroll": ["ad_vod/bunny_fp.processed.1711966939428660.mp4", "
    ad_vod/bunny_fp.processed.1711966939428660.mp4"]
    }
}
```

При такой конфигурации *Flussonic Media Server* покажет ролик `ad_vod/bunny_fp.processed.171196693` перед просмотром, а затем раз в **три минуты** (180 секунд) будет показывать перечисленные мидроллы в том порядке, в котором они указаны в списке `midroll`.

Врезка рекламы по SSAI по меткам SCTE-35

```
{
    "ad_inject":
    {
        "v": 2,
        "preroll": ["ad_vod/bunny_fp.processed.1711966939428660.mp4"],
        "midroll_insert_by": "splicing",
        "midroll": ["ad_vod/bunny_fp.processed.1711966939428660.mp4", "
    ad_vod/bunny_fp.processed.1711966939428660.mp4"]
    }
}
```

При такой конфигурации *Flussonic Media Server* покажет ролик `ad_vod/bunny_fp.processed.171196693` перед просмотром, а затем при появлении меток SCTE-35 *Flussonic* будет показывать все перечисленные мидроллы до окончания рекламного блока. Ролики проигрываются циклически в том порядке, в котором они указаны в списке `midroll`.

Пояснения к параметрам

- `"v": 1` — врезка рекламы в виде отдельных сегментов, или `"v": 2` — врезка рекламы в сегменты основного потока.

- `"preroll": "ad_vod/bunny_fp.processed.1711966939428660.mp4"` — путь к VOD-локации с прероллом. Замените имя файла на то, которое вернул скрипт на шаге 2.
- `"midroll_insert_by": "splicing"` — метод врезки рекламы по SCTE-меткам, или `"midroll_insert_by": "interval"` — метод врезки рекламы через равные интервалы времени.
- `"midroll_interval": 180` — интервал в секундах для показа мидроллов, если `"midroll_insert`
- `"midroll": ["ad_vod/bunny_fp.processed.1711966939428660.mp4", "ad_vod/bunny_` — список мидроллов для врезки. Замените имена файлов на те, которые вернул скрипт на шаге 2.

Шаг 4. Проверьте работы врезки

Убедитесь, что всё работает, проиграв поток по HLS или DASH. Например, воспользуйтесь ссылкой **Embed** на вкладке **Output** потока.

`https://FLUSSONIC-IP/demo/embed.html`

Если все в порядке, перед трансляцией потока `demo` вы увидите клип Big Buck Bunny.

Такие плагины, как AdBlock, могут блокировать рекламные ролики. Если что-то пошло не так, попробуйте **отключить плагины**.

> Плеер на вкладке **Overview** в профиле потока не будет проигрывать рекламу. Не используйте его для проверки этой функции.

Отслеживание событий о показе рекламы зрителям

Собирайте статистику по показам рекламы, настроив логирование события 7Cevents–30%7Ceventtag/events/op events-list/response%7Cevents–30%7Cevent‘ad_inject‘ с помощью event_sink.

Chapter 9

IP cameras

9.1 IP cameras

9.1.1 Обработка звука с IP-камер

Обработка звука с IP-камер

Большинство IP камер умеют отдавать звук только в кодеках PCMA/PCMU (также известных как G.711a и G.711u).

Flussonic Media Server умеет записывать аудио такого формата в архив и отдавать его по тому протоколу, который поддерживает этот кодек — RTMP.

Другие протоколы не подразумевают передачи этого кодека, поэтому отправить такой звук по HLS или выгрузить в MP4 так, чтобы клиент услышал звук, не получится.

Для того, чтобы все клиенты могли услышать звук, следует включить транскодирование звука с IP камеры.

Для этого надо сначала установить пакет `flussonic-transcoder`:

```
apt-get install -y flussonic-transcoder
```

Теперь включите транскодирование звука на потоке с камеры следующим образом:

```
stream camera1 {
 input rtsp://localhost:554/origin output_audio=aac;
}
```

Так *Flussonic Media Server* осуществит транскодирование звука в кодек AAC и сможет отдавать его всем клиентам по всем протоколам.

9.1.2 Миксер

Flussonic Media Server умеет создавать новый поток, используя видео и аудио из других live потоков. Это можно использовать, например, чтобы наложить музыку поверх камеры наблюдения.

Настройка

Создайте новый поток и укажите в качестве источника протокол `mixer://` и имя двух потоков: откуда взять видео и откуда взять аудио:

```
stream mix {
  input mixer://stream1,stream2;
}
```

где:

- **stream1** — имя live-потока из которого Flussonic Media Server возьмет видеодорожку
- **stream2** — только звук.

> Миксер работает только с live-потоками, уже заведенными во Flussonic Media Server. Не используйте миксер с VOD файлами и не указывайте источник прямо в строке с `mixer://`.

Пример применения

Например, у нас есть поток `cam1` с камеры видеонаблюдения (h264 video + pcmu звук), но камера расположена высоко на столбе и ничего кроме шума ветра не слышно.

Логично выключить звук совсем, захватив только видео:

```
stream camera {
  input fake://fake;
}
stream silent {
  input rtsp://localhost/camera tracks=1;
}
```

А можно создать новый поток с помощью миксера, который наложит аудио с другого источника. Например, радио:

```
stream origin {
  input fake://fake;
}
stream cam1 {
```

```
  input rtsp://localhost/origin tracks=1;
}
stream radio {
  input shout://localhost/origin/shoutcast;
}
stream cam1radio {
  input mixer://cam1,radio;
}
```

Figure 9.1: Flussonic stream mixing

В такой конфигурации мы получаем поток `cam1radio`, который можно вставить на сайт. Зрителям будет интереснее смотреть на камеру, слушая новости, а в архив будет сохраняться оригинальный поток `cam1`, включая оригинальный звук с камеры. Это может быть полезно, если произойдет ЧП.

Вы также можете архивировать исходное видео и аудио с помощью функции DVR:

```
stream cam1 {
  input rtsp://cam1.local/h264;
  dvr /storage 7d;
}
```

9.1.3 Мозаика

Несколько потоков можно ”склеить” в одну мозаику и показывать как один поток. Создается мозаика с помощью транскодирования.

Сборка мозаики из потоков

Через веб-интерфейс Watcher можно включить только клиентскую мозаику из камер. Подробнее об этом написано в документации Watcher.

Чтобы создать серверную мозаику:

Установите пакет `flussonic-transcoder`:

Замечание. Пакет `flussonic-transcoder` необходим только в случае, если вы планируете использовать CPU для выполнения транскодирования. Если вы используете Nvidia NVENC, то он не нужен.

```
apt-get -y install flussonic-transcoder
```

Добавьте поток с источником `input mosaic://...`:

```
stream cam1 {
  input rtsp://IP-CAMERA-ADDRESS:PORT/camera1;
}
stream cam2 {
  input rtsp://IP-CAMERA-ADDRESS:PORT/camera2;
}
stream cam3 {
  input rtsp://IP-CAMERA-ADDRESS:PORT/camera3;
}
stream cam4 {
  input rtsp://IP-CAMERA-ADDRESS:PORT/camera4;
}
stream mosaic0 {
  input mosaic://cam1,cam2,cam3,cam4?fps=20&preset=ultrafast&bitrate=1024k&
   size=340x240&mosaic_size=16;
}
```

После `mosaic://` идет через запятую список камер, которые будут использоватся в мозаике.

В опциях можно указывать настройки, которые будут использоваться в энкодере.

Опция `fps=20` жестко указывает скорость видео. Для камер можно указывать `fps=video`, чтобы привязать кадры мозаики к первой камере.

Опция `size=320x240` настроит размер каждой камеры в мозаике. Если от камеры поток с картинкой больше, то она будет уменьшена до этого размера.

Опция `mosaic_size` указывает, на сколько камер будет рассчитана мозаика. Это может быть удобно для того, чтобы фиксировать размер мозаики.

9.1.4 Timelapse

Экспорт ключевых кадров в MP4-файл

Flussonic Media Server позволяет выгрузить только ключевые кадры в виде MP4 файла. Это может быть полезно для создания timelapse-видео.

Такой файл можно выгрузить на компьютер клиента, обратившись по адресам:

```
http://FLUSSONIC-IP/STREAMNAME/archive-1350274200-4200.mp4?timelapse
```

запрос файла из ключевых кадров на скорости 25 fps.

```
http://FLUSSONIC-IP/STREAMNAME/archive-1350274200-4200.mp4?timelapse=20
```

запрос файла с коррекцией fps, длительность файла будет 20 секунд.

9.1.5 Flussonic RAID для DVR

Flussonic RAID для DVR — это программный RAID-массив, который обеспечивает высокую надежность и удобство при записи видеоданных на десятки дисков. Flussonic RAID имеет существенные преимущества перед похожими решениями:

- Нет необходимости покупать дорогой аппаратный RAID-контроллер, например, на 60 дисков. Все диски используются в режиме JBOD (Just a Bunch of Disks). Можно использовать диски разных размеров. Каждый диск нужно отформатировать отдельно и вмонтировать в систему в определенный каталог. После этого вы настраиваете Flussonic, и он начинает следить за их состоянием и самостоятельно распределять данные по ним.
- Надежность: При сбое в работе какого-либо диска запись продолжится на другие диски в массиве. Может пострадать только часть данных, записанная на отказавший диск.
- Непрерывная работа: Можно добавлять диски или удалять их из RAID прямо во время записи архива, и не требуется перезапуск Flussonic — изменения применятся сразу.
- Автоматическая ”бесшовная” миграция данных между дисками в RAID, что делает возможным, например, удалить все данные с какого-либо диска без прерывания работы с этим DVR архивом (записи или проигрывания).
- Автоматическое распределение записываемых данных между дисками в RAID. Flussonic сам решает, на какой диск оптимальнее будет записать данные. Поток данных может быть больше, чем можно успеть записать на один диск за приемлемое время — поэтому Flussonic равномерно распределяет запись между дисками. А чтобы снизить затраты электроэнергии, можно ограничить число дисков, на которые одновременно может производиться запись.
- Защита от записи в случае, если произошла ошибка с монтированием дисков. Такая защита предотвратит запись всех данных в корневой раздел.

На этой странице:

Настройка программного массива дисков для Flussonic Media Server

Существующие архивы перенести в RAID нельзя, можно только начать запись заново. Чтобы начать работу с RAID, нужно сконфигурировать сервер, добавив настройки дискового массива (это по сути глобальные настройки DVR) и в потоках указать этот массив для записи архива.

Порядок настройки различается для Flussonic Watcher и для Flussonic Media Server.

Массив дисков для записи DVR создается на уровне операционной системы, когда вы монтируете диски, а затем весь массив управляется программно Flussonic-сервером.

Настройки массива — это глобальные настройки DVR. Flussonic позволяет задать эти настройки в конфигурационном файле `/etc/flussonic/flussonic.conf` или в веб-интерфейсе.

Порядок настройки DVR таков. Сначала зададим глобальные настройки массива, например:

```
dvr my_raid {
  root /storage/raid;
  raid 0;
  metadata idx;
  disk volume1;
  disk volume2 keep;
  disk volume3 migrate;
}
```

Затем в настройках потока укажем, что архив следует записывать в массив `my_raid` с глубиной семь дней:

```
stream channel5 {
  input fake://fake;
  dvr @my_raid 7d;
}
```

Поток примет глобальные настройки, указанные в настройках массива `dvr my_raid`. Некоторые из них можно будет переопределить в настройках потока.

Пример DVR RAID с полным набором настроек Дисковый массив имеет три вида настроек:

- Глобальные настройки DVR массива
- Настройки, которые можно переопределить в настройках отдельного потока
- Параметры, управляющие процессом записи на диски.

Ниже описание всех настроек.

Глобальные настройки DVR, которые действуют только для DVR на дисковом массиве:

- — базовый каталог, в котором смонтированы диски и находятся индексы. Пример: `root /dvr/raid;`

- — включает возможность работы с массивом (допустимое значение — 0). Если вы включили эту опцию, проверяется наличие активных дисков. Flussonic проверит major device корневого пути и файлов в каталогах и не разрешит запись, если это не смонтировавшаяся папка. Пример: `raid 0;`
- — количество дисков, на которые будет идти запись данных. При большом количестве дисков вести запись сразу на все неэкономно по затратам электроэнергии, поэтому можно ограничиться несколькими дисками. Если не указывать эту опцию, то запись будет производиться на все диски, имеющие достаточно свободного места. Пример: `active 2;`
- — директория под root для хранения кэшированных метаданных. Эту директорию *создавать вручную не нужно*, она будет создана при применении настроек. Мы рекомендуем использовать SSD для быстрого доступа к архиву. Пример: `metadata idx;`
- — путь до примонтированного диска. Указанные в опции `disk` пути должны быть реальными точками монтирования. Например:

```
Filesystem                          Size  Used Avail Use% Mounted on
/dev/mapper/pve-vm--15--disk--1  7.9G  5.7G  1.8G  77% /
/dev/loop0                       196G  4.0G  182G   3% /dvr/_raid_/d1
/dev/loop1                       196G   4.0G  182G   3% /dvr/_raid_/d2
```

Настройки, которые применимы к отдельным потокам (при указании в глобальных настройках DVR они будут по умолчанию применяться ко всем потокам. Их можно переопределить в настройках отдельного потока):

- **limits** — ограничения на размер и глубину архива. Например, `90% 3G 1d`.
- **replicate** — репликация DVR. Можно указать порт в необязательном параметре `port=1234`. Пример: `replicate port=2345;`
- **copy** — копирование данных по частям в другое место. Пример: `copy /opt/storage;`
- **schedule** — расписание записи в архив. Пример: `schedule 3-6,5-8,23-5;`

Описания режимов для управления записью на диски вы можете найти в API Reference.

Монтирование дисков в Linux

Так как Flussonic RAID это программный RAID, то диски необходимо монтировать как обычные отдельные диски `ext4`. Вы можете использовать и другую файловую систему, но мы настоятельно рекомендуем `ext4`, если нет серьезного повода использовать что-то другое.

Приводим реальную конфигурацию одного из наших лабораторных серверов:

```
root@dvr:~# lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0   9.8T  0 disk /storage/d1
```

```
sdb      8:16   0   9.8T  0 disk /storage/d2
sdc      8:32   0   9.8T  0 disk /storage/d3
sdd      8:48   0   9.8T  0 disk /storage/d4
sde      8:64   0 119.2G  0 disk└─
sde1   8:65   0 119.2G  0 part /

root@dvr:~# cat /etc/fstab
# / was on /dev/sda1 during installation
# OS SSD DISK
UUID=5081dd01-6b97-4166-b05c-e9f59476b553 /               ext4    errors=
    remount-ro 0       1

#sda
UUID=8c5bcc39-8599-4545-a373-f63a441b53b8 /storage/d1   ext4    defaults,
    nofail,x-systemd.device-timeout=5      0       2
#sdb
UUID=f4888c12-6faa-4ac1-b4fb-e04c3e4ddc31 /storage/d2   ext4    defaults,
    nofail,x-systemd.device-timeout=5      0       2
#sdc
UUID=7feedd26-feb4-47ad-99b8-ec732cbd87aa /storage/d3   ext4    defaults,
    nofail,x-systemd.device-timeout=5      0       2
#sdd
UUID=ed41fad5-92fd-4737-87ff-c6a859aeed10 /storage/d4   ext4    defaults,
    nofail,x-systemd.device-timeout=5      0       2
```

Важно:

- Операционная система должна быть установлена на SSD диск.
- Все HDD, используемые в массиве Flussonic RAID, должны быть примонтированы в одну корневую директорию.

Настройки в конфигурационном файле Flussonic:

```
dvr raid {
  root /storage;
  raid 0;
  disk d1;
  disk d2;
  disk d3;
  disk d4;
}
```

Если у вас есть какие-либо вопросы по поводу монтирования дисков в Linux или конфигурации Flussonic RAID, задайте их своему системному администратору или нашей службе поддержки клиентов.

Статистика времени выполнения

Статистика о состоянии RAID доступна в API вызове `dvr_get`. См. HTTP v3 API.

Статистика показывает состояние дисков в RAID:

- `status` — примонтирован или не примонтирован
- `blobs_count` — количество блобов на диске
- `size` — размер в байтах
- `usage` — процент загруженность диска
- `io_usage` — загруженность диска из статистики /proc/devstat.

Если выполняется миграция, в ответе отображается скорость миграции, расчетное время окончания миграции и время последнего изменения значений:

- `migration_speed` — скорость копирования последнего блоба, байт в секунду
- `migration_eta` — приблизительное время окончания миграции, UTC seconds
- `migration_updated` — время последнего изменения значений `migration_speed` и `migration_eta`, позволяет понять, насколько устарели данные.

После окончания миграции эти параметры становятся `undefined`.

Глобальные настройки DVR в веб-интерфейсе

В этом разделе мы расскажем, как записать архив в наш программный RAID-массив.

Чтобы добавить глобальные настройки DVR, включая RAID массивы: Перейдите в **Config** > **DVR** > **Add DVR** и определите глобальные настройки RAID массива. Чтобы добавить более одного массива, ещё раз воспользуйтесь кнопокй **Add DVR**.

Поле **Copy chunks to this location** нужно для копирования архива статического потока в указанную директорию.

> По-другому открыть глобальные настройки можно из настроек (любого) потока в разделе **DVR** этого потока, кликнув ссылку **Edit DVR configurations** под **Global DVR config**.

Чтобы применить глобальные настройки к потоку и переопределить некоторые из них: В настройках данного потока в разделе **DVR** кликните в поле **Global DVR config** и выберите ранее созданный массив в появившемся списке. В примере был выбран один массив `my_raid`:

На странице **DVR** появятся некоторые значения, взятые из глобальных настроек (Path, Copy chunks to this location, and so on) — это те настройки, которые вы можете переопределить для потока если необходимо.

Figure 9.2: Flussonic DVR RAID

Архив потока будет записываться в выбранный RAID.

Чтобы открыть глобальные настройки из настроек потока, в разделе **DVR** этого потока кликните ссылку **Edit DVR configurations** под **Global DVR config**.

Как заменить сбойный диск в Flussonic RAID

Замена сбойного диска для Flussonic RAID в общем случае состоит из следующих шагов:

- Включить режим Abandon для диска во вкладке ”DVR” для выбранного стримера. Описание режимов диска можно посмотреть в документации здесь:

Figure 9.3: Flussonic DVR RAID

https://flussonic.com/doc/api/reference/#tag/dvr/operation/dvr_disk_save|body|mode

- В начале следующего часа проверить, что запись на этот диск больше не осуществляется:

```
lsof -p pidof streamer | grep 'devicename'
```

где `devicename` - название диска в /dev/.

- Заменить ”отказавший” диск на новый, проверить его название - оно не должно измениться при сохранении точки подключения в сервере.
- Разметить и отформатировать новый диск, смонтировать его по тому же пути, что был у

старого диска - проверить в `/etc/fstab` и секции ”dvr watcher” в `/etc/flussonic/flussonic.con` или во вкладке ”DVR” выбранного стримера.

- Переключить режим для этого нового диска в рейде из ”Abandon” в ”Normal”.
- В начале следующего часа Flussonic начнет писать архив на этот новый диск.

Если сервер не поддерживает замену дисков в ”горячем режиме”, то диск нужно заменить с отключением сервера.

Part III

Technologies

Chapter 10

Media Server

10.1 Media Server

10.1.1 Flussonic Media Server

Flussonic Media Server — это серверное программное обеспечение для создания высоконагруженных сервисов доставки видео любого масштаба. Flussonic Media Server может принимать, хранить, транскодировать и доставлять видео любому количеству зрителей и на любые устройства.

Flussonic написан на языке программирования Erlang, который позволяет добиться:

- исключительной производительности при параллельной обработке данных,
- высокого уровня отказоустойчивости сервера,
- масштабирования сервиса от одного сервера до сложной распределенной сети.

Области применения

- IPTV- и OTT-сервисы,
- VSaaS и CCTV,
- видеостриминг.

Возможности Flussonic Media Server

- **Прямой эфир (Live streaming)**. Захват прямого эфира со спутника, IP-камеры, платы захвата или программы для кодирования видео. Поддержка приёма публикации по VMix, OBS, Atemi, конвертерам HDMI-RTMP, пультам видеомонтажа и браузерам. Поддержка приёма публикации по SDI, ASI, SRT, RTMP, WebRTC, WebRTC с ABR и проигрывания по WebRTC (WHEP) с настройкой ретрансмитов. Проигрывание мультибитрейтных потоков в прямом эфире и из архива по HLS, DASH или MSS через Интернет. Проигрывание мультибитрейтных потоков в прямом эфире по WebRTC с низкой задержкой.
- **Захват из DVB-карт захвата**. Захват каналов с DVB-T/C/S карт. Захват WebRTC (WHIP), H.323, UDP MPEG-TS, TCP MPEG-TS и HTTP MPEG-TS, M4S/M4F.
- **Передача в DVB-сети**. Подготовка MPTS для передачи на спутник в соответствии со стандартами качества TR-101290. Публикация MPTS через DVB-C карту для передачи в DVB-C сети.
- **Приём субтитров DVB**. Конвертация DVB-субтитров из картинок в текст.
- **Видео по запросу (Video on Demand, VOD).** Раздача файлов с адаптивной подстройкой битрейта и выбором языка.
- **Собственный канал из VOD-файлов**.
- **DVR архив**. Запись видеопотоков на диск и управление архивом, поддержка нужной глубины архива и объёма заполнения жёсткого диска. Проигрывание DVR в мультибитрейте по HLS, DASH и MSS через Интернет.

- **Проигрывание с задержкой**. Проигрывание мультибитрейтных потоков в прямом эфире с задержкой по HLS, DASH или MSS через Интернет.
- **DRM**. Поддержка Widevine, PlayReady, Solocoo, Conax, EzDRM, BuyDRM KeyOS и других DRM-систем. Полный список см. в 7Cdrm–0%7Cvendortag/stream/operation/stream-get/response%7Cdrm 0%7CvendorFlussonic API Reference.
- **Облачное хранилище**. Flussonic может забирать файлы с HTTP-серверов и облачных хранилищ типа Amazon S3 и OpenStack Storage (Swift), а также сохранять архив в облаке.
- **Транскодер**. Поддержка видеокодеков H.264, H.265, AV1, MPEG-2 video, деинтерлейсинга, мультибитрейта, аудиокодеков AAC, MP3, Opus, MPEG-2 audio, AC3, PCMA, PCMU.
- **Интеграция с CDN**.
- **Проигрывание MPTS мультикаст и юникаст, а также SPTS в мультикаст-группу для локальных сетей**.
- **Веб-плеер для HLS, DASH, MSS и WebRTC**.
- **Врезка рекламы**. Врезка рекламы на стороне сервера.
- **Резервирование захвата источника на нескольких серверах**.
- **API для интеграции**. Интеграция с Middleware и веб-сайтом сервиса.
- **Статистика по просмотру каналов**.
- **Инструкции по готовым решениям типовых задач**. Захват, транскодирование, запись в архив с доступом, пакетайзер, плеер, мониторинг.
- **Проектная поддержка**.

Попробовать Flussonic Media Server

Чтобы попробовать Flussonic Media Server, сделайте следующее:

- Запросите триал, заполнив форму на сайте.
- Следуйте инструкции в Быстром старте Media Server.

Что нового Информацию о возможностях последних версий Flussonic Media Server вы можете найти в нашем блоге.

10.1.2 Глоссарий

Здесь можно познакомиться с терминами, которые встречаются в документации на Flussonic Media Server.

Adaptive bitrate streaming (Адаптивный стриминг)

Адаптивный стриминг — способ видео стриминга по HTTP, когда исходное содержимое кодируется с разным качеством.

Deinterlacing (Деинтерлейсинг)

Деинтерлейсинг – это конвертирование чересстрочного видео в прогрессивное.

В чересстрочном видео нечетные и четные строки кадра демонстрируются как отдельные поля. Сначала на экране отображаются нечетные строки, затем – четные. Два таких поля вместе составляют один видеокадр. Чересстрочное видео хорошо подходит для вещания, т.к. изображения могут демонстрироваться на экране с низкой пропускной способностью. Но у чересстрочного видео также есть и недостаток: при быстром движении видео может быть нечетким, т.к. в один момент времени захватывается лишь часть изображения, и по краям видеокадра могут быть заметны искажения.

В прогрессивном же видео нечетные и четные строки отображаются одновременно, то есть видеокадр отображается на экране целиком.

Деинтерлейсинг необходим для комфортного просмотра старых телепрограмм на современных компьютерах и мобильных устройствах.

DVR

Это возможности Flussonic по записи потоков в архив и работы с архивом — воспроизведения по разным протоколам или экспорта в файл MP4.

Frame (Кадр)

Видео кадр — это одно из многих статичных изображений, из которых состоит видео. Это минимальная составная часть видеотрека. У каждого кадра есть начало и продолжительность.

Frame duration (Продолжительность кадра)

Для видеотрека продолжительность кадра — это время от начала кадра до начала следующего кадра.

Этот параметр важен для некоторых протоколов. В обычной ситуации продолжительность кадра равна разнице между моментами начала двух соседних кадров. Однако иногда (когда соединение прерывается) возможны разрывы в видео. В результате разница между моментами начала двух соседних кадров не равна продолжительности кадра. Такая проблема рассматривается как разрыв между кадрами и решается по-разному в зависимости от протокола.

GOP (Group of Pictures)

Group of Pictures (GOP, группа изображений) — упорядоченная цепочка следующих друг за другом изображений в кодированном видеопотоке. Кадры объединяются в группы для целей межкадровой компрессии, без которой передача видеопотока по сети расходовала бы огромное время и трафик. Сжатие выполяет программа-кодировщик (encoder).

Сжатый поток представляет собой следующие друг за другом GOP. На стороне получателя декодер составляет видимые кадры из кадров, заключенных в GOP.

GOP состоит из I-кадра и следующих за ним P- и B-кадров:

- I-кадр (keyframe, опорный) – первый кадр в GOP. Содержит полное изображение, которое сжимается независимо от других кадров (без ссылок на них).
- P-кадр, B-кадр – следующие кадры в GOP.

GOP size (Размер GOP)

Размер GOP (расстояние в кадрах между соседними ключевыми кадрами) — количество кадров в одном GOP. Размер GOP у потока бывает переменным и постоянным. Когда Flussonic перекодирует видео, он создает все GOP постоянного размера. Вы можете настраивать размер GOP в опциях транскодера Flussonic.

Multicast (Мультикаст)

Multicast – способ вещания видео, при котором UDP пакеты передаются от одного источника группе подписчиков по мультикаст IP адресу в локальной сети.

Подробнее о мультикасте читайте в документации

Prepush

Prepush – метод для более плавного проигрывания видео по протоколам HTTP MPEG-TS, RTMP и RTSP (при отправлении пакетов по TCP). Видеостриминговый сервер сохраняет в буфере каждый GOP, прежде чем передать его клиенту. При подключении клиента сервер отправляет клиенту первый GOP из буфера и продолжает трансляцию потока с отставанием, равным продолжительности

GOP. Клиент при этом поддерживает буфер продолжительностью, равной продолжительности GOP (если перевести GOP в секунды). Если соединение клиента с сервером прерывается или замедляется, то он проигрывает видео из буфера, что сглаживает неравномерность сетевой передачи.

Publishing (Публикация)

Публикация — передача видео на Flussonic Media Server от программ и устройств, которые являются инициаторами начала видеотрансляции. Flussonic при этом сторона, ожидающая соединения.

К публикации на Flussonic относится:

- Передача видео с любого мобильного устройства на сервер Flussonic.
- Передача видео из OBS (Open Broadcaster Software) или vMix на сервер Flussonic. Подробнее
- Передача видео с HTML страницы в браузере через WebRTC на сервер Flussonic. Подробнее

А это мы не называем публикацией:

- Получение мультикаста
- Захват потока из какого-либо источника (в этом случае Flussonic является инициатором соединения).

Segments (Сегменты)

В протоколах DASH и HLS сегмент — это единица разбивки видео на части для передачи и буферизации, измеряется в секундах. В сегменте может быть больше одного GOP и сегмент должен быть кратен GOP. Сегмент не может быть меньше, чем GOP (если измерять количеством секунд).

Устройство, передающее видео по протоколам DASH и HLS, передает видео в виде сегментов продолжительностью три секунды и плейлиста, в котором эти сегменты перечислены. Прежде, чем начать проигрывать видео по этому протоколу, клиент загружает буфер. Если соединение клиента с сервером прерывается или замедляется, то он проигрывает видео из буфера, что сглаживает неравномерность сетевой передачи. Как правило, клиент загружает три сегмента, прежде чем начать проигрывание потока.

Transcoder (Транскодер)

Транскодер — компонент, который выполняет прямое цифровое преобразование исходного видеопотока, чтобы создать мультибитрейтный поток, изменить параметры видео (кодек, размер изображения, битрейт) или наложить логотип.

Video codec (Видео-кодек)

Это технология сжатия ”сырого” видео для последующей упаковки в контейнер для передачи по определенному протоколу.

Video container (Видео-контейнер)

Контейнер (или транспорт) — это формат, в который кодированные данные в файле или в потоке упаковываются для передачи по сети. Пакеты с аудио и видеоданными передаются на транспортном уровне по модели OSI.

Формат контейнера самодостаточен и независим от протокола доставки, т.е. можно упаковать данные, но не передавать их по сети, а сохранить и проиграть в плеере локально.

Video streaming protocol (Протокол потоковой передачи видео)

Протокол потоковой передачи видео — это правила обмена данными, командами и ответами на них между двумя участниками видеосвязи (клиент-сервер либо peer-to-peer).

При подготовке данных для передачи по сети:

- Сначала видео- и аудио-данные нужно сжать
- Затем упаковать в контейнер для передачи по определённому протоколу.

10.1.3 Модель данных Flussonic

Понятие ”media” в Flussonic

Термином **media** в *Flussonic* называют любой проигрываемый поток или файл. У каждого media есть имя и набор данных.

Для удобного и эффективного управления этими данными мы используем модель данных, которая позволяет разделить media на отдельные элементы. Для потоков и файлов используется одна и та же модель данных. Давайте рассмотрим составные части media в *Flussonic*.

Составные части media

Трек (track) Каждое media содержит отдельные дорожки или **треки**, которые представляют собой видео, аудио или текст (например, субтитры). Например, в фильме может быть один видеотрек, три аудиотрека (на английском, немецком и русском языках) и три соответствующих трека с субтитрами.

Каждый трек характеризуется **контентом**, то есть физическим содержимым (видео, аудио или текст) и набором других параметров. Набор параметров трека зависит от его контента. Например, для видео трека можно задать ширину и высоту передаваемого изображения, а также **кадровую частоту (frame rate)** — скорость, с которой изображения сменяют друг друга на экране. Для аудио трека можно указать другой набор параметров, например, язык и **частоту дискретизации (sample rate)** — количество отсчетов непрерывного по времени сигнала, взятых в секунду с микрофона (или другого источника) для его дискретизации. Текстовые треки очень просты и не имеют никаких специальных параметров.

Flussonic автоматически присваивает каждому треку идентификатор, например, ”v1, v2, ...” — для видео треков, ”a1, a2, ...” — для аудио треков, ”t1, t2, ...” — для треков с WebVTT субтитрами, ”l1, l2, ...” — для треков с DVB-субтитрами или телетекстом.

Каждый трек, независимо от контента, можно разделить на **кадры**.

Кадр (frame) Кадр — это минимальная составная часть трека. Кадры бывают как видео, так и аудио или текстовыми. Для видео трека кадр — это одно из многих статичных изображений, из которых состоит видео.

У каждого кадра есть начало (timestamp) и **продолжительность (frame duration)**. Понятие продолжительности кадра различается для аудио и видео.

Для аудио трека продолжительность кадра зависит от частоты дискретизации. Например, компакт-диски обычно записываются с частотой 44.1 кГц, что означает, что каждую секунду снимается 44100 отсчетов. В этом случае продолжительностью аудио кадра можно считать 1/44100 секунды.

Для видео трека продолжительность кадра — это время от начала кадра до начала следующего кадра. Этот параметр важен для некоторых протоколов. В обычной ситуации продолжительность кадра равна разнице между моментами начала двух соседних кадров. Однако иногда (когда

соединение прерывается) возможны разрывы в видео. В результате разница между моментами начала двух соседних кадров не равна продолжительности кадра. Такая проблема рассматривается как разрыв между кадрами и решается по-разному в зависимости от протокола. Например, воспроизведение по HLS в этом случае продолжится, а по DASH — прервется, и при этом начнется новый период (читайте подробнее здесь).

Важная особенность модели данных *Flussonic* заключается в том, что кадры никогда не перекрывают друг друга. Перекрытие кадров может привести, например, к такой проблеме как наложение субтитров. *Flussonic* позволяет избежать этой проблемы, т. к. каждый кадр начинается не раньше, чем начало предыдущего кадра + его продолжительность.

GOP Чтобы передать видео по сети с ограниченной полосой пропускания, как правило, необходимо его сжать. Помимо сжатия самих кадров, используется более продвинутая технология — межкадровое сжатие. Она заключается в том, что полностью отправляются лишь некоторые кадры (называемые **опорными кадрами (keyframes)**), а вместо остальных кадров отправляется лишь разность между текущим кадром и опорным. Приемник (декодер) использует опорный кадр вместе с этими разностями, чтобы воссоздать нужный кадр с приемлемым качеством.

В целях межкадрового сжатия кадры в треке группируются в **GOP**. GOP (Group of Pictures, группа изображений) — упорядоченная цепочка следующих друг за другом изображений в кодированном видеопотоке или файле. Каждый GOP состоит из I-кадра (опорного кадра) и следующих за ним P- и B-кадров:

- **I-кадр (keyframe, опорный)** – первый кадр в GOP. Содержит полное изображение, которое сжимается независимо от других кадров (без ссылок на них). Каждый GOP начинается с опорного кадра.
- **P-кадр** – содержит разницу между изображением на предыдущем P-кадре и изображением на текущем кадре. Сжимается со ссылкой на ключевой кадр.
- **B-кадр** – содержит ссылки на соседние кадры (на один или несколько I и P кадров).

Типичный GOP содержит повторяющийся шаблон из нескольких B- и P-кадров, расположенных после опорного кадра. Например, шаблон может быть таким:

I B B P B B P B B P B B

В идеальной ситуации опорные кадры должны выбираться при смене сцены (метод *scene detection*). Однако большинство программ для обработки видео настроены на работу с GOP одинакового размера. Поэтому в большинстве случаев используются равные по размеру GOP, например, в телевидении стандартный GOP состоит из 28 кадров.

Группировка в GOP применима только к видео кадрам. Соответствующие аудио и текстовые кадры синхронизируются с GOP.

Важно понимать, что GOP не имеет смысла без опорного кадра. То есть воспроизвести видео только из середины GOP невозможно.

Каков оптимальная длина GOP? **Почему GOP не должен быть слишком длинным?** Потому что слишком длинный GOP может привести к увеличению *zap time* – времени между переключением канала с помощью пульта дистанционного управления и началом воспроизведения нового канала. Если зритель нажимает на пульт управления до того, как завершился предыдущий GOP, на экране появится неактуальное изображение. Эта проблема может быть критичной для видеоигр или видеозвонков.

Для решения этой проблемы *Flussonic* использует функцию **prepush**: сохраняет в буфере каждый GOP, прежде чем передать его клиенту. При подключении клиента сервер отправляет клиенту первый GOP из буфера и продолжает трансляцию потока с отставанием, равным продолжительности GOP. Клиент при этом поддерживает буфер продолжительностью, равной продолжительности GOP (если перевести GOP в секунды). Если соединение клиента с сервером прерывается или замедляется, то он проигрывает видео из буфера. Это сглаживает неравномерность сетевой передачи, хотя может привести к росту задержки.

Почему GOP не должен быть слишком коротким? Потому что чем длиннее GOP, тем лучше сжатие. Короткий GOP сжимать сложнее.

Разные приложения используют разную длину GOP, но обычно эта длина укладывается в диапазон 0,5 – 2 секунды.

Открытый GOP В некоторых случаях сжать видео можно еще сильнее, используя так называемые **открытые GOP (open GOPs)**. В открытых GOP P-кадры ссылаются на кадры, предшествующие опорному кадру. Это позволяет снизить битрейт еще на 5-7 %. Однако использование открытых GOP может привести к проблемам, если используются **сегменты**.

Сегмент (segment) Сегмент — элемент следующего уровня в нашей модели данных. Он содержит один или более GOP, а также аудио и текстовые кадры (синхронизированные с видео кадрами). Сегменты нужны для некоторых протоколов, таких как HLS и DASH.

Сегмент может совпадать с GOP, а может и не совпадать. Но в любом случае он всегда кратен GOP и начинается с опорного кадра.

Важная особенность транскодера *Flussonic* заключается в том, что сегменты всегда синхронизированы для всех видео треков. Если видео кодируется с помощью какого-либо другого (не очень хорошего) транскодера, возможна ситуация, когда для одного видео трека начал воспроизводиться новый сегмент, а для другого видео трека все еще воспроизводится предыдущий. В этом случае плееру будет сложно переключиться на другой видео трек, а значит, эта ситуация неприемлема для мультибитрейтного стриминга. В *Flussonic* все сегменты имеют одинаковый размер, и момент начала проигрывания (timestamp первого опорного кадра) совпадает для одного сегмента в разных видео треках. Поэтому все видео треки проигрываются синхронно.

Имейте в виду, что продолжительность аудио кадров не совпадает с продолжительностью видео кадров, поэтому возможна ситуация, когда новый сегмент уже воспроизводится, а аудио кадры еще добавляются к предыдущему сегменту. Это нормальная ситуация.

Мы храним сегменты изолированно друг от друга. И это может привести к проблемам, если

используются открытые GOP, т. к. P-кадры не могут ссылаться на кадры из предыдущих сегментов. В этом случае изображение иногда может распадаться, так что для получения лучшего качества изображения необходимо дождаться следующего сегмента.

10.2 Ingest

10.2.1 Требования к формату входных потоков и файлов

В этой статье описаны требования и рекомендации к формату входного потока и файла. Ниже вы найдете список протоколов, медиаконтейнеров и кодеков, которые поддерживает *Flussonic Media Server* для входных потоков с источников.

Особенности формирования входных потоков

Формат входящего потока определяется источником, но иногда вы можете выбрать его самостоятельно. Например, в настройках аппаратного кодера или видеокамеры. Когда нет возможности выбрать формат входного потока заранее, необходимо транскодирование (подробнее см. далее).

Flussonic Media Server поддерживает определенный набор протоколов, кодеков и контейнеров на входе. Все остальные либо не будут проигрываться, либо их проигрывание будет сопряжено с ошибками. Например, с ошибками в DVR.

Flussonic различает форматы потоков (live) и файлов (VOD).

Формат потока

Ниже приведена таблица протоколов и соответствующих им контейнеров и видео-/аудиокодеков, поддерживаемых *Flussonic Media Server* для входных потоков. Актуальный список кодеков, поддерживаемых *Flussonic*, для входных потоков вы можете найти в 7Cinputs%7Cstats%7Cmedia-info%7Ctracks%7Ccodectag/stream/operation/stream-get/response%7Cinputs%7Cstats%7Cmedia-info%7Ctracks%7CcodecFlussonic Media Server API Reference.

Протоколы	Медиаконтейнер	Видеокодек	
HLS	MPEG-TS	H.264, H.265 (HEVC), AV1	
WebRTC	-	H.264, VP8, VP9, AV1	
RTMP	FLV	H.264	
RTSP	-	H.264, H.265 (HEVC)	
RTP	-	H.264, H.265 (HEVC)	
HTTP/UDP	MPEG-TS	H.264, H.265 (HEVC), MPEG-2 video	
SRT	любой	H.264, H.265 (HEVC), MPEG-2 video, VP8, VP9, AV1	AAC, EAC3, MP3,
Shoutcast/ICEcast	-	-	
H323	-	H.264, H.265 (HEVC)	

Подробнее про публикацию видео из браузера по WebRTC можно прочитать на странице Публикация по WebRTC.

Формат файла

Ниже приведена таблица контейнеров и видео-/аудиокодеков, поддерживаемых *Flussonic Media Server* для файлов.

Контейнер	Видеокодек	Аудиокодек
MP4 (.mp4, .f4v, .mov, .m4v, .mp4a, .3gp, .3g2)	H.264, H.265 (HEVC)	MP3, AAC (все профили)

Узнайте подробнее о VOD-файлах на странице Файлы VOD.

Частота кадров в секунду (FPS)

Если поток с источника имеет частоту кадров в секунду менее 10, то *Flussonic Media Server* считает такой источник нерабочим. Оптимальнее всего выставить значение кадров в секунду равное 15 или выше.

Некоторые камеры наблюдения могут отдавать поток рывками длиной в пару секунд из-за низких настроек FPS. В таком случае достаточно увеличить значение FPS в настройках камеры.

> Увеличение значения FPS повышает нагрузку на камеру. Так некоторые камеры на высоких настройках могут либо перегреться и зависнуть, либо не смогут поддерживать хороший стабильный поток. Для таких камер нужно поддерживать баланс между качеством и работоспособностью. Это можно сделать, например, сократив количество одновременных подключений к камере до одного, т.е. чтобы видео с камеры получал только *Flussonic Media Server*.

Как получить нужный формат

Когда формат входного потока предопределен источником и нет возможности выбрать его заранее (например, поток со спутника), то необходимо транскодирование. В *Flussonic Media Server* есть встроенный транскодер, умеющий конвертировать потоки различных форматов в H.264/AAC и не только.

Подробнее о транскодировании и его настройке см. Транскодирование и Транскодер.

Таким образом, *Flussonic* сможет принять входной поток без необходимости транскодирования, если:

- Формат потока находится в списке поддерживаемых.
- Частота кадров потока не меньше 10 кадров в секунду, а лучше 15 и выше.

Если ваш входной поток не соответствует вышеприведенным требованиям, вы можете преобразовать его в поток нужного формата с помощью встроенного транскодера.

10.2.2 Переключение источников

По разным причинам видео может поступать нестабильно или вовсе стать недоступным. Чтобы избежать ситуации, когда у потребителей отсутствует видео, необходимо подготовить альтернативные источники, чтобы вещать их при отсутствии основного источника до тех пор, пока он не восстановится. *Flussonic* умеет автоматически плавно переключать источники.

Содержание:

Резервные источники видео

Чтобы обеспечить резервирование источников, в настройках потока следует перечислить несколько разных источников видео, и тогда *Flussonic Media Server* будет переключаться между ними, если выбранный ранее источник стал недоступен. Можно использовать как видео потоки, так и файлы.

Под термином ”недоступен” подразумевается либо немедленное отключение, либо отсутствие кадров в течение 60 секунд (180 секунд для источников по hls://, playlist://, timeshift://).

Как работает переключение Если пришлось переключиться на второй источник, то *Flussonic Media Server* будет периодически перепроверять первый источник на работоспособность.

Если от пропавшего источника снова приходят кадры, Flussonic дождется ключевого кадра и только тогда переключится на этот источник. Этим самым мы обеспечиваем качественное переключение без задержек. Поэтому даже при переключении на видео с большим GOP, например, приходящее по HLS, Flussonic обеспечит качественное переключение.

Резервные источники ДОЛЖНЫ иметь идентичный набор аудио и видео дорожек, как у источника, если вы хотите получить лучший пользовательский опыт для ваших зрителей.

Пример

Поток `example_stream` имеет два источника. Переключение на второй источник произойдет, если в течение 20 секунд от первого не придет ни одного кадра.

```
stream failover_example_stream1 {
  input udp://239.0.0.1:1236 source_timeout=20;
  input tshttp://localhost:80/clock2/mpegts;
}
stream clock1 {
  input fake://fake;
  push udp://239.0.0.1:1236 multicast_loop;
}
stream clock2 {
  input fake://fake;
}
```

Когда происходит переключение

Flussonic Media Server следит только за тем, чтобы от источника были кадры, и переключается на другой источник, если кадры не поступают определенное количество секунд.

Он не переключит источник при пропадании звука или видео, или при росте количества MPEG-TS CC ошибок.

Настройки переключения источников

source_timeout Для каждого источника можно указать количество секунд, которое *Flussonic Media Server* будет ожидать кадры от этого источника. По умолчанию таймаут равен 60 секундам (180 секундам для hls://, playlist://, timeshift://).

Можно задавать `source_timeout` как для всего потока, так и для каждого из источников. Опция `source_timeout` всего потока не применяется, если указаны `source_timeout` для его видео-источников. Пример:

```
stream backup_timeout {
    input publish:// source_timeout=10;
    input fake://fake source_timeout=5;
    source_timeout 20;
}
```

Если переключение происходит на ваш взгляд часто, можно увеличить `source_timeout`, чтобы не было "перепрыгиваний" с одного источника на другой. Наоборот, чтобы не ждать подолгу переключения, можно уменьшить таймаут.

Значение `source_timeout` не играет роли при переключении источника вручную.

priority Для источников можно указать приоритет, в порядке которого *Flussonic* будет переключаться на источник. Самый высокий приоритет у источника с `priority=1`, более низкий у источника с `priority=2`, и т.д.

По умолчанию, высший приоритет имеет первый источник в списке в настройках, низший приоритет — последний источник в списке. Если для каких-то источников в списке приоритет не указан, то применяется нумерация по умолчанию.

Flussonic проверяет приоритеты после того, как определены все активные источники.

Если приоритет недоступного источника такой же как и текущего активного, то *Flussonic Media Server* не будет периодически проверять такой недоступный источник и не будет пытаться на него переключиться.

```
stream example_stream {
  input fake://fake priority=2 source_timeout=30;
  input tshttp://10.2.4.5:9000/channel/5 priority=1 source_timeout=10;
}
```

Правила переключения источников в соответствии с их приоритетом и состоянием (работает поток или нет) распространяются и на публикуемые источники, позволяя гибко управлять показом публикуемого потока.

Переключение источников вручную

Flussonic поддерживает принудительное ручное переключение.

Как переключить источники вручную, не дожидаясь таймаута:

- В конфигурации потока поменять порядок, в котором указаны источники. Применяется, если не указан `priority`.

Например, переместите источник в верх списка, и *Flussonic* переключится на него.

- В конфигурации потока изменить приоритеты переключения

Например, поменяйте местами приоритеты, и *Flussonic* переключится на источник с более высоким приоритетом.

- Через API явно включить другой источник.

Запись в архив

Если на потоке настроен архив, то *Flussonic Media Server* пишет видео из активного источника в архив.

В случае, если последним адресом указан локальный файл:

```
stream backup_example_stream1 {
  input tshttp://example.com/origin;
  input file://vod/bunny.mp4;
  dvr /storage;
}
```

то видео из этого файла также будет писаться в архив.

Чтобы пользователям показывать заглушку вместо потока, но не писать её в архив, надо использовать заглушку

Файл в качестве запасного источника

Можно использовать статическое видео (видеофайлы) в качестве источника данных для отработки отказа.

Flussonic поддерживает два разных способа использования файлов, при этом поведение системы различается.

Указание файла как *источника* потока, используя схему 'input file://'

```
  stream backup_example_stream2 {
    input tshttp://10.0.4.5:9000/channel/5;
    input file://vod/bunny.mp4;
  }
```

Можно использовать как MP4, так и MPEG-TS файлы (.ts).

Указание файла как *заглушки* для потока через опцию 'backup' Заглушка для основного потока задается с помощью опции `backup`. Когда основной поток недоступен, *Flussonic* будет показывать файл, не переключаясь с недоступного источника. Это полезно в ряде ситуаций.

```
  stream backup_example_stream3 {
    input tshttp://10.0.4.5:9000/channel/5;
    backup vod/bunny.mp4;
  }
```

Чем 'backup' отличается от 'input file://'

В отличие от переключения источников с опцией `input file://<VOD location>`, при использовании заглушки технически не происходит переключения на другой источник. Это

особенно полезно при публикации видео на *Flussonic*, чтобы не заставлять публикующего клиента переподключаться при каждом сбое связи.

Файл-заглушка, указанная через `backup <VOD location>`, по умолчанию не транскодируется и не пишется в архив, если вы не настроили иначе. Файловый источник `input file://<VOD location>` будет всегда записываться в архив.

Когда стоит применять `backup` вместо `input file://`:

- При плохом соединении с публикующим видео клиентом *Flussonic* продолжает принимать кадры, не обрывая соединение с клиентом. Это позволяет клиенту не обрывать сеанс публикации и не начинать его снова. При перебоях зрители видят файл-заглушку, и понимают, что трансляция ещё не окончена.
- Когда все источники работают со сбоями, из-за чего *Flussonic* часто переключается между ними, лучше показывать файл-заглушку. Если же указать файл как ещё один источник, то зрители увидят его только после того, как истекут таймауты для каждого из сбоящих источников.
- Если вы записываете основной поток в архив, но не хотите записывать в архив проигрывание файла, чтобы в дальнейшем файл не попал в просмотр.
- С помощью таймаутов потока и заглушки можно гибко управлять переключением источников в процессе публикации.

Опции файла-заглушки (‘backup‘)

Заглушка принимает опции:

```
stream example {
    input udp://239.0.0.1:1234;
    backup vod/bunny.mp4 video_timeout=5 audio_timeout=10 timeout=20 dvr=
   true transcode=true;
    dvr /storage;
}
```

dvr=true Если основной поток пишется в архив, то заглушка будет тоже записываться в архив, если указать эту опцию:

```
stream example {
    input udp://239.0.0.1:1234;
    backup vod/bunny.mp4 dvr=true;
    dvr /storage;
}
```

timeout=10

Figure 10.1: Опции файла-заглушки

Через какое время (в секундах) *Flussonic* переключится на заглушку, если от основного источника видеопотока перестанут поступать кадры. Важно, что источник при этом остается запущенным, что позволяет, например, оставаться на сокете клиенту-публикатору.

Опция учитывает все виды кадров.

Flussonic также может переключаться на резервный источник только тогда, когда не поступают кадры определенного типа (видео или аудио), что позволяет лучше управлять переключением, если источник имеет плохое качество. Для разных кадров можно использовать разные интервалы тайм-аута. Чтобы учитывать не любые кадры, а только аудио или видео, используйте опции `video_timeout` и `audio_timeout` (см. далее в этом списке).

Если не указывать `timeout` специально для заглушки, то в случае отсутствия кадров будет использоваться `source_timeout` основного источника.

Комбинируя настройки `timeout` и `source_timeout`, можно:

- задать более долгое время для того, чтобы мобильные клиенты успели начать публикацию и не были отключены;
- при этом переключаться за заглушку как можно скорее.

Пример:

```
stream example {
    input publish:// source_timeout=20;
    input fake://fake;
    backup vod/bunny.mp4 timeout=1;
}
```

В этом примере:

До начала публикации будет воспроизводиться поток `fake`. Затем подключается клиентское приложение, которое будет публиковать видео на *Flussonic*. Если после подключения от клиента-публикатора потока так и не пришли кадры в течение 20 секунд, то клиент принудительно отключается и начинает проигрываться fake.

Затем, когда началась публикация, файл-заглушка `backup-file.mp4` начнет проигрываться, если в течение 1 секунды не поступит ни одного кадра от публикуемого потока. Источник публикации при этом пока не отключается.

При возобновлении кадров от источника поток переключается на клиента-публикатора. Если же в течение 20 секунда кадры не приходят, то публикатор принудительно отключается и начинает проигрываться fake.

video_timeout=5 Через какое время (в секундах) *Flussonic* переключится на заглушку, если от основного источника перестанут поступать видео-кадры.

Если указать `video_timeout`, `audio_timeout` и одновременно `timeout` основного источника, переключение сработает по таймауту, который наступит быстрее. У этих опций одинаковый приоритет.

audio_timeout=10 Через какое время (в секундах) *Flussonic* переключится на заглушку, если от основного источника перестанут поступать аудио-кадры.

Если указать `audio_timeout`, `video_timeout` и одновременно `timeout` основного источника, переключение сработает по таймауту, который наступит быстрее. У этих опций одинаковый приоритет.

transcode=true См. Транскодирование файла-заглушки ниже.

Транскодирование файла-заглушки

transcode=true Если основной поток транскодируется, то заглушка также будет транскодироваться, если указать эту опцию.

```
stream backup_transcode {
    input udp://239.0.0.1:1235;
    source_timeout 5;
    backup vod/bunny.mp4 transcode=true;
    transcoder vb=1000k ab=64k;
}
```

Это позволяет изменять настройки транскодера потока без перекодирования вручную файла заглушки. Также это избавляет от необходимости готовить несколько файлов-заглушек с разным битрейтом.

”Тревожная” кнопка

Во *Flussonic* предусмотрена возможность настройки ”тревожной” кнопки.

”Тревожная” кнопка

— это механизм, необходимый для сигнализации старта/остановки потока.

Flussonic проверяет статус ”тревожной” кнопки перед тем, как запустить источник.

Для того чтобы добавить ”тревожную” кнопку, укажите путь до файла-сигнализации в одном из параметров: `allow_if` или `deny_if` источника `url` в настройках потока.

При каких условиях источник будет запущен?

- В указанном через файле **содержится 1** (`allow_if=/PATH/TO/FILE`).
- В указанном через файле **содержится 0** (`deny_if=/PATH/TO/FILE`).

Следовательно, во всех остальных случаях источник не будет запущен. А именно:

- В указанном через `allow_if` файле содержится 0 (`allow_if=/PATH/TO/FILE`).
- В указанном через `deny_if` файле содержится 1 (`deny_if=/PATH/TO/FILE`).
- Файл содержит что-то отличное от 1 и 0.
- Файл не существует.

Конфигурация выглядит следующим образом:

Figure 10.2: Flussonic fallback file

```
stream example_stream {
    input m4f://FLUSSONIC-IP/STREAM_NAME deny_if=/PATH/TO/FILE;
    input udp://239.0.0.2:1236 allow_if=/PATH/TO/FILE;
}
```

В примере выше мы определили двум источникам (`m4f://` и `udp://239.0.0.2:1236`) — одну ”тревожную” кнопку (`/PATH/TO/FILE`). Если `/PATH/TO/FILE` содержит 1, то источник `m4f://` не запустится, в то время как источник `udp://239.0.0.2:1236` — запустится. Значит, если `/PATH/TO/FILE` содержит 0, то `m4f://` запустится, а `udp://239.0.0.2:1236` — нет.

Обобщим всё вышесказанное в таблице (”+” — источник запущен, ”-” — источник остановлен):

статус файла	allow_if	deny_if
Файл содержит 1	+	-
Файл содержит 0	-	+
Файл содержит иные символы	-	-
Файл не существует	-	-

10.2.3 Live — потоковое вещание

Flussonic Media Server умеет ретранслировать потоковое видео, перепаковывая его на лету в разные форматы. Это означает, что вы можете захватить MPEG-TS поток и раздавать его одновременно тысячам получателей, например, в DASH или HLS, и публиковать в RTMP на YouTube.

Flussonic Media Server поддерживает три типа потоков:

- `static` — постоянно живущие (транслируемые).
- `ondemand` — потоки, транслируемые по запросу.
- `live` – публикуемые пользователем. См. Публикация

Содержание:

- Статические потоки
- Потоки по запросу (ondemand)
- Проигрывание потоков
- Скриншоты потока
- Файл-заглушка ’backup’ для потока
- Чем заглушка ’backup’ отличается от ’input file://’
- Подстановки
- Запись потоков (DVR)
- Сдвиг по часовой зоне (Timeshift)
- Выдача потока в UDP multicast
- Настройки потоков для IP камер наблюдения
- Включение audio-only варианта HLS
- Захват потока с другого сервера Flussonic Media Server
- DRM в live-потоках
- Обнаружение тишины в потоке

Статические потоки

Статические потоки запускаются при старте сервера и Flussonic Media Server постоянно следит за ними. Если источник пропадает (выключился транскодер, авария на антенне), то Flussonic Media Server будет пытаться переподключиться к источнику и ни при каких обстоятельствах не перестанет этого делать.

IPTV-канал или IP-камера обычно объявляются именно как статический поток.

Flussonic Media Server поддерживает различные типы источников, которые указываются в виде URL-адресов.

Пример конфигурации потоков из **/etc/flussonic/flussonic.conf**:

```
stream example_stream {
  input udp://239.0.0.1:1234;
}
```

В этой конфигурации:

- `example` — это имя потока, по которму можно обратиться к Flussonic Media Server и получить его.
- `udp://239.0.0.1:1234` — URL источников.

Важно. Имя канала должно состоять из латинских букв, чисел, точки (`.`), символов минуса (`-`) и нижнего подчеркивания (`_`). Если в имени будет что-то кроме этих символов, то работоспособность DVR и вещания мы гарантировать не можем.

Чтобы добавить поток через веб-интерфейс:

Перейдите на вкладку **Media** и нажмите **Add** рядом со **Streams**.

Введите название потока и URL адрес источника. После этого нажмите **Create**, и созданный поток появится в списке.

Замечание. По умолчанию создается статический поток. Чтобы поменять тип потока на **On demand**, кликните по **Static** рядом с названием потока.

Поток создан. Теперь можно перейти на страницу потока для проверки захвата:

Потоки по запросу (on-demand)

Если поток нужен не всё время, а только по требованию пользователя, можно указать Flussonic Media Server отключать его при неиспользовании и включать по запросу.

Для указания такого типа потока надо заменить `stream` на `ondemand`:

```
ondemand ipcam {
  input rtsp://localhost:554/source;
```

Figure 10.3: Flussonic add stream

```
}
```

Важно. Если между источником и репитером используется RTMP, RTSP или HTTP MPEG-TS, то снимать с репитера HLS не получится, потому что эти протоколы требуют наличия 10-30 секундного буфера видео в памяти. Плеер не начнет проигрывание пока этот буфер не накопится, и, следовательно, первый пришедший пользователь будет ждать это время. Единственный источник, который не подвержен этой проблеме – это другой Flussonic Media Server по протоколу HLS. Во Flussonic Media Server используются собственные расширения, позволяющие моментально стартовать видео на iPhone.

Можно регулировать время жизни потока после отключения клиента:

```
ondemand ipcam1 {
  input rtsp://localhost:554/source;
```

Figure 10.4: потоковое вещание

```
  retry_limit 10;
  clients_timeout 20;
}
```

В конфигурации выше указано следующее: пытаться переподключиться к сбойному источнику **не больше 10 раз** и после ухода последнего клиента гонять поток вхолостую **не дольше 20 секунд**.

Проигрывание потоков

Проигрывание описано в разделе про выходное видео.

Figure 10.5: Flussonic ondemand

JPEG-скриншоты потока

Flussonic Media Server может делать JPEG-скриншоты потока. Для этого надо указать опцию `thumbnails` в настройках потока:

```
stream example {
  input fake://fake;
  thumbnails;
}
```

А можно указать *Flussonic Media Server*, где забирать скриншоты. Это поможет не тратить ресурсы CPU. Многие IP-камеры имеют специальный URL со скриншотами:

```
stream example {
```

```
  input rtsp://localhost:554/source;
  thumbnails url=http://examplehost:5000/snapshot;
}
```

URL с потоком скриншотов можно найти в документации к вашей модели камеры.

Figure 10.6: Flussonic JPEG thumbnails

Последний скриншот потока доступен по адресу `http://flussonic:80/example/preview.jpg`

MJPEG поток скриншотов доступен по адресу `http://flussonic:80/example/preview.mjpeg`

Читайте также:

- Скриншоты о JPEG-скриншотах.
- Видео-скриншоты об MP4-скриншотах, экономящих ресурсы.

Файл-заглушка для потока

Если поток недоступен, можно запустить на проигрывание файл — резервный источник видео, указанный через `backup <VOD location>`. Так можно делать для любых live-потоков, включая публикуемые.

```
stream example {
  input tshttp://10.0.4.5:9000/channel/5;
  backup vod/bunny.mp4;
}
```

Необходимо указывать путь к файлу-заглушке относительно VOD-локации, например `vod/backup.mp4`, где `vod` — уникальное имя VOD-локации, при этом нельзя указывать абсолютный путь к файлу `backup.mp4`.

> Если оригинальный поток идет без звука (например, с IP камеры), то и файл-заглушка тоже должен быть без звука.

По умолчанию, файл-заглушка не пишется в архив и не транскодируется. Но это можно настроить.

Подробнее о разных способах использования файлов как запасных источников потока рассказано в разделе .

Чем 'backup' отличается от 'input file://'

В отличие от переключения источников с опцией `input file://<VOD location>`, при использовании заглушки технически не происходит переключения на другой источник. Это особенно полезно при публикации видео на Flussonic, чтобы не заставлять публикующего клиента переподключаться при каждом сбое связи.

Файл-заглушка, указанная через `backup <VOD location>`, по умолчанию не транскодируется и не пишется в архив, если вы не настроили иначе. Файловый источник `input file://<VOD location>` будет всегда записываться в архив.

Когда стоит применять `backup` вместо `input file://`:

- При плохом соединении с публикующим видео клиентом Flussonic продолжает принимать кадры, не обрывая соединение с клиентом. Это позволяет клиенту не обрывать сеанс публикации и не начинать его снова. При перебоях зрители видят файл-заглушку, и понимают, что трансляция ещё не окончена.
- Когда все источники работают со сбоями, из-за чего Flussonic часто переключается между ними, лучше показывать файл-заглушку. Если же указать файл как ещё один источник, то зрители увидят его только после того, как истекут таймауты для каждого из сбоящих источников.

- Если вы записываете основной поток в архив, но не хотите записывать в архив проигрывание файла, чтобы в дальнейшем файл не попал в просмотр.
- С помощью таймаутов потока и заглушки можно гибко управлять переключением источников в процессе публикации.

Подробнее об использовании файлов как запасных источников потока, см. в разделе .

Подстановки

Иногда имена потоков на удалённом сервере заранее неизвестны. В таких случаях возникает необходимость вычислить URL источника (`input`) на основании **запрашиваемого** потока. Для этого воспользуйтесь специальной возможностью шаблонов (`template`) по подстановке имени:

```
template nsk {
  prefix nsk;
  input rtsp://streamer:555/%s;
}
template ams {
  prefix ams;
  input hls://streamer:8081/%s/index.m3u8;
}
```

Если в шаблоне в URL источника есть паттерн `%s`, то вместо этого паттерна будет подставлена часть имени потока, **следующая** за префиксом (`prefix`) шаблона. Иначе говоря, часть URL запрашиваемого клиентом потока, предшествующая префиксу `prefix`, включая и сам префикс, отбрасывается и оставшаяся часть подставляется в URL источника.

Например, при запросе потока по такому URL: `http://FLUSSONIC-IP/ams/ort/index.m3u8`, *Flussonic* будет отдавать поток из источника со следующим URL: `hls://streamer:8081/ort/index.m3u8` т.е. часть `http://FLUSSONIC-IP/ams/` "отрезается", а оставшаяся часть `ort/index.m3u8` подставляется в URL источника `input`.

Как это использовать?

Например, вещать каналы с разных серверов для зрителей разных регионов, т.е. для каждого региона — свой сервер. Это поможет обеспечить быструю и оптимальную доставку контента до зрителя.

Запись потоков (DVR)

Во Flussonic Media Server встроена система записи потоков в так называемый архив. Архиватор потоков умеет записывать видео, предоставлять доступ к произвольному фрагменту, экспортировать части архива в виде MP4 файлов, очищать старые файлы и поддерживать заполнение хранилища на приемлемом уровне.

Для включения архива достаточно указать опцию `dvr` в конфиге потока:

```
stream foxlive {
  input tshttp://trancoder-5:9000/;
  dvr /storage 90% 5d;
}
```

Подробнее в разделе про архив и методы работы с ним.

Сдвиг по часовой зоне (Timeshift)

Flussonic Media Server умеет воспроизводить записанный в архив поток с фиксированным отставанием.

> Flussonic Media Server соблюдает фиксированное отставание четко по указанному временному интервалу. Если в архиве были ”дырки”, то пользователи не получат никакого видео за это время.

Для таймшифта есть отдельная схема источника — `timeshift://`:

```
stream channel {
  input fake://fake;
  dvr /storage 90% 5d;
}
stream channel-2h {
  input timeshift://channel/7200;
}
```

Отставание указывается в секундах.

Выдача потока в UDP multicast

Flussonic Media Server умеет ретранслировать поток из источника в локальную сеть.

Flussonic Media Server пытается выдавать UDP максимально монотонно во времени, чтобы не создавать скачкообразной нагрузки на сеть.

```
stream example_stream {
  input tshttp://localhost:80/origin/mpegts;
  push udp://239.0.4.4:1234;
}
```

Настройки потоков для IP-камер видеонаблюдения

Можно указать Flussonic Media Server запрашивать поток с камеры только по UDP. Это бывает нужно с камерами, в которых проблемная реализация TCP.

Figure 10.7: Flussonic timeshift

```
stream cam1 {
  input rtsp://localhost:553/bunny.mp4;
  rtp udp;
}
```

Обратите внимание, что из десктопных браузеров показывать H.265 сейчас фактически умеют **только** Chrome (версия 107 и выше), Microsoft Edge (версия 16 и выше) и Safari (версия 11 и выше). Из мобильных браузеров — Chrome (версия 107 и выше) и Safari для iOS (версия 11.0 и выше). Подробнее в статье Воспроизведение H265.

Если с камеры не надо забирать звук (например, он в G.726) то можно указать Flussonic Media

Server забирать только одну дорожку. Ее номер необходимо указать в конфигурации:

```
stream cam1 {
  input rtsp://localhost:554/origin tracks=1;
}
```

Для того, чтобы автоматически транскодировать аудио с камеры из G.711a или G.711u в AAC, укажите другой протокол:

```
stream cam1 {
  input rtsp2://localhost:554/origin;
}
```

Включение audio-only варианта HLS

Apple при валидации программ в AppStore может потребовать, чтобы поток был с audio only вариантом. Если добавить директиву в конфигурацию:

```
stream cam1 {
  input fake://fake;
  add_audio_only;
}
```

и при этом в потоке есть и видео, и аудио, то Flussonic Media Server будет генерировать мультибитрейтный вариантный плейлист из двух потоков: один обычный, второй только со звуком.

Захват потока с другого сервера Flussonic Media Server

Детально вопросы передачи видео между разными Flussonic Media Server описаны в разделе про кластеризацию видеопотоков в Flussonic.

DRM в live-потоках

Детально вопрос использования таких DRM как AES-128, SAMPLE-AES и Conax описан в разделе про DRM.

Обнаружение тишины в потоке

Flussonic умеет обнаруживать низкий уровень звука (отсутствие звука) на источнике входного потока и оповещать об этом. Подробнее в разделе Обнаружение тишины.

10.2.4 Публикация видео на сервер

Flussonic Media Server может принимать видео от программ и устройств, которые являются инициаторами начала видеотрансляции. Это называется ***публикация***.

Публикация может использоваться в тех случаях, когда устройство не имеет статического IP-адреса или вообще находится за NAT и *Flussonic Media Server* не сможет обратиться к этому устройству или программе за видео.

Что мы называем *публикацией* на сервер Flussonic:

- Передачу видео с любого мобильного устройства на сервер *Flussonic*.
- Передачу видео из OBS (Open Broadcaster Software) или vMix на сервер *Flussonic*. Узнать больше.
- Передачу видео с HTML-страницы в браузере через WebRTC на сервер *Flussonic*. Узнать больше.

Что мы не называем *публикацией* на сервер Flussonic:

- Получение мультикаста.
- Прием потока из какого-либо источника.

В этих ситуациях *Flussonic* неким образом подключается к источнику. Но ситуация, когда *Flussonic* не делает ничего для того, чтобы инициировать соединение, называется *публикацией*: например, публикация — это когда мобильное устройство подключается к *Flussonic* для передачи видео на сервер.

Публикация видео в социальные сети не подходит под определение публикации, которое мы используем в этой документации, так как не является публикацией на *Flussonic*.

Протоколы *Flussonic Media Server* может принимать запросы на публикацию видео по протоколам RTMP, RTSP, HTTP MPEG-TS, WebRTC и SRT.

Содержание

- Авторизация публикации
- Архив и динамические имена потоков
- Перепубликация
- Переключение источников, timeout

Публикация в статический поток

Если вы точно знаете под каким именем поток должен появиться на сервере, то вы можете создать поток и указать, что вы разрешаете в него публиковать, путём указания особого источника `publish://`.

Чтобы создать статический поток с публикуемым источником:

- В административном интерфейсе создайте поток: **Media** > **Stream** > **add**.
- Заполните **Stream name**.
- Укажите `publish://` в качестве **Source URL**. Либо, сохранив настройки, перейдите на вкладку **Input** и установите переключатель **Publication** в положение **enabled**.
- Нажмите **Create**.
- Для указания дополнительных настроек источника публикации, нажмите **options**:

В файле конфигурации этой настройке соответствует такая строка:

```
stream published {
  input publish://;
}
```

URL для публикации по разным протоколам URL для публикации можно посмотреть в веб-интерфейсе для конкретного потока:

- Кликните на имя созданного потока, чтобы открыть его настройки, и перейдите во вкладку **Input**.
- Доступные ссылки для публикации отобразятся в разделе **Publish links** (см. скриншот выше).

При публикации в статический поток вы можете использовать следующие адреса:

- **RTSP**: `rtsp://FLUSSONIC-IP/stream_name`

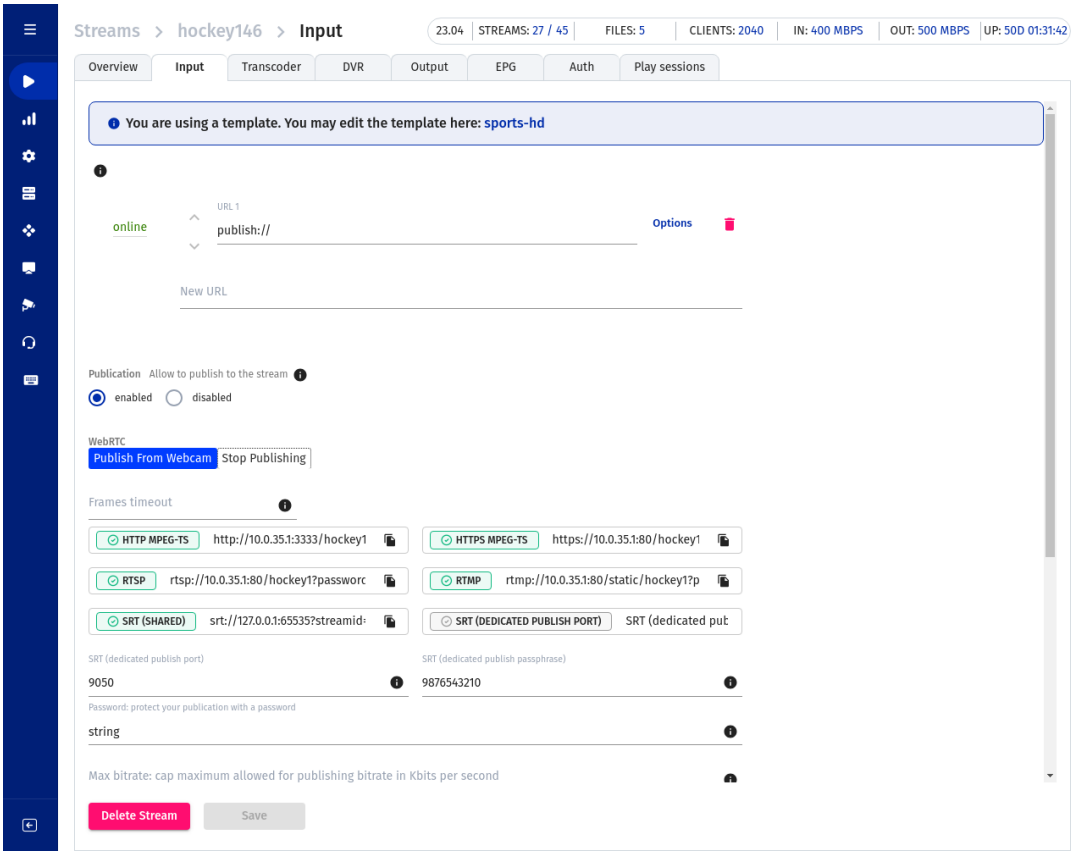


Figure 10.8: Создание статического потока с публикуемым источником

- **HTTP MPEG-TS**: `http://FLUSSONIC-IP/stream_name/mpegts`
- **RTMP**: `rtmp://FLUSSONIC-IP/stream_name` или `rtmp://FLUSSONIC-IP/static/stream_nam`
  Особенности URL для RTMP см. на странице Публикация по RTMP.
- **WebRTC**: `http://FLUSSONIC-IP/stream_name/whip`. Подробнее о публикации по WebRTC см. на странице Публикация по SRT.
- **SRT**: `srt://FLUSSONIC-IP:SRT_PORT?streamid=#!::r=STREAM_NAME,m=publish`.
  Подробнее об URL для SRT см. на странице Публикация по SRT.

Публикация по динамическому имени

Зачем использовать динамическое имя и publishing location Указывать статическое имя для публикуемых потоков может быть непрактично в следующих случаях:

- Публикация длится ограниченный период времени (по сравнению с трансляцией телеканала, например) и происходит лишь один раз.

- Может быть слишком много публикаций, чтобы создавать отдельный поток для каждой из них. Кроме того, вам пришлось бы прописывать настройки отдельно для каждого потока.
- В некоторых случаях неизвестно, под каким именем внешнее приложение-клиент будет публиковать поток. *Flussonic* не знает имя потока, пока не придет запрос на публикацию видео на сервер. Зачастую имя выбирается произвольно (например, в случае с веб-чатами).

Flussonic позволяет решить эти проблемы с помощью создания *префикса публикации* (*publishing location*), где можно указать настройки сразу для множества потоков.

Динамическое имя означает, что полное имя потока формируется из предварительно настроенного *префикса публикации* и заранее не известного имени, определенного во внешнем приложении.

Таким образом, если вы заранее не знаете под каким именем будет публиковаться поток или этих потоков будет много, то настройте *префикс публикации*:

```
template chats {
  prefix chats;
  input publish://;
}
```

Здесь `chats` — это префикс публикации. Все потоки, опубликованные под префиксом `chats`, будут иметь настройки, которые вы укажете в директиве `template`. Чтобы узнать подробнее об опциях и настройках потока см. Flussonic Media Server API.

В директиве `template` вы можете указать несколько префиксов, чтобы создать несколько мест публикации. Вы также можете указать пустой префикс (`""`) для публикации потока с любым префиксом или даже вообще без префикса. См. подробнее в разделе Шаблоны и префиксы.

URL для публикации по разным протоколам В случае публикации по динамическому имени вам надо публиковать стримы под именами с префиксом, например:

- **RTSP**: `rtsp://FLUSSONIC-IP/template_name/stream_name`
- **HTTP MPEG-TS**: `http://FLUSSONIC-IP/template_name/stream_name/mpegts`
- **RTMP**: `rtmp://FLUSSONIC-IP/template_name/stream_name`. Особенности URL для RTMP см. на странице Публикация по RTMP.
- **SRT**: `srt://FLUSSONIC-IP:SRT_PORT?streamid=#!::m=publish`. Публикация по SRT с динамическим именем поддерживается только для такого формата URL, т.е. когда задан отдельный порт для потока или группы потоков.
- **WebRTC**: `http://FLUSSONIC-IP:PORT/template_name/stream_name/whip`.

Что именно идет после `template_name` — это дело клиента. *Flussonic Media Server* заранее не знает, какое именно это будет имя.

Проверка публикации

Публикация по RTMP Опубликовать по RTMP можно, например, с помощью ffmpeg:

```
ffmpeg -re -i /opt/flussonic/priv/bunny.mp4 -vcodec copy -acodec copy -f
   flv rtmp://localhost/chats/tmp
```

При этом в веб-интерфейсе появится новый поток:

Figure 10.9: Публикация нового RTMP-потока на сервер

Подробнее о формировании URL для публикации по RTMP читайте на странице Публикация по RTMP.

Публикация по RTSP Некоторые клиенты могут публиковать видео по RTSP.

Flussonic Media Server поддерживает автоматический выбор между UDP и TCP транспортом: как захочет клиент, так и будет принимать.

Имя потока должно быть полным: `chats/my/chat-15`

```
ffmpeg -re -i /opt/flussonic/priv/bunny.mp4 -vcodec copy -acodec copy -f
   rtsp rtsp://localhost/chats/my/chat-15
```

Публикация по MPEG-TS При транскодировании потока с помощью ffmpeg можно опубликовать видео по HTTP, добавив суффикс `mpegts`:

```
ffmpeg -re -i /opt/flussonic/priv/bunny.mp4 -vcodec copy -vbsf
   h264_mp4toannexb -acodec copy -f mpegts http://localhost:80/chats/my/
   chat-15/mpegts
```

Публикация по SRT Вы можете использовать `ffmpeg`, чтобы протестировать публикацию:

```
/opt/flussonic/bin/ffmpeg -re -i PATH_TO_VIDEO -c copy -y -f mpegts 'srt://
   FLUSSONIC-IP:SRT_PORT?pkt_size=1316&streamid=#!::r=STREAM_NAME,m=publish
   '
```

, где:

- `FLUSSONIC-IP` — IP-адрес сервера *Flussonic*.
- `SRT_PORT` — порт SRT.
- `STREAM_NAME` — имя потока.
- `m=publish` — режим публикации.

Также можно публиковать поток и из другого стороннего ПО с поддержкой SRT, например OBS Studio:

Публикация по WebRTC Для проверки публикации по WebRTC можно опубликовать видео с веб-камеры или воспользоваться демо-приложением.

Защита публикации паролем

Flussonic Media Server может проверять пароль при публикации потока. Укажите пароль в конфигурационном файле следующим образом:

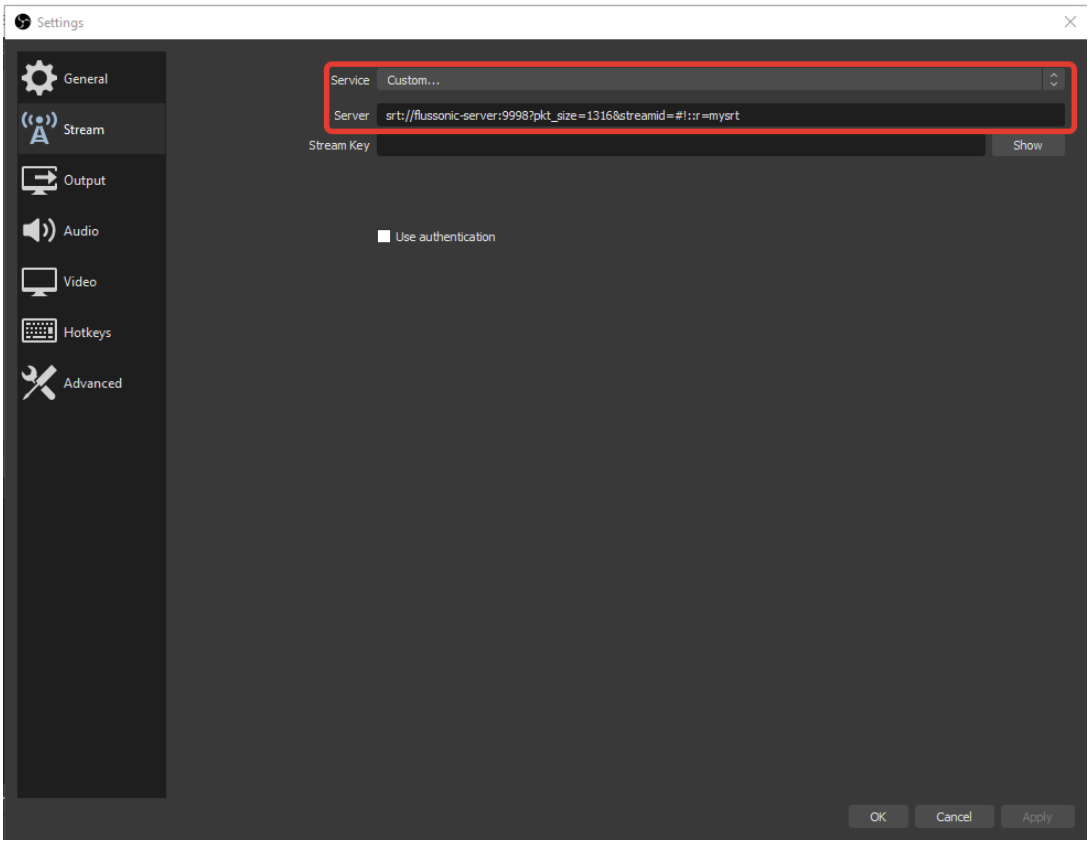


Figure 10.10: OBS srt

```
template chats {
    prefix chats;
    password mypass;
    input publish://;
}
stream published {
    password secure;
    input publish://;
}
```

Примеры для проверки публикации:

- Чтобы опубликовать в защищенную паролем зону поток по **RTMP**, укажите данные следующим образом:

```
stream name: mystream?password=mypass
```

- Чтобы опубликовать поток по **HTTP MPEG-TS**, укажите данные следующим образом:

```
http://192.168.2.3:80/s1/mpegts?password=secure
ffmpeg -re -i video.mp4 -vcodec copy -acodec copy -f flv rtmp
    ://192.168.2.3/live/mystream?password=mypass
ffmpeg -re -i video.mp4 -vcodec copy -bsf h264_mp4toannexb -acodec copy -f
    mpegts http://192.168.2.3:80/s1?password=secure
```

Авторизация публикации

Flussonic Media Server позволяет указать HTTP URL авторизационного бэкенда, который будет проверять расширенную информацию об источнике публикации перед тем, как разрешить или запретить публикацию. Подробнее см. Авторизация сессий публикации.

Архив и динамические имена потоков

Вы можете сконфигурировать архив для префикса публикации:

```
template recorded {
  prefix recorded;
  input publish://;
  dvr /storage 3d 500G;
}
```

В этом случае публикуемое видео будет записываться и будет доступно даже если публикация остановилась.

Когда клиент перестает публиковать видео, то стрим через какое-то время пропадает и *Flussonic Media Server* про него почти ничего не знает. Почти — означает, что в индексе архива информация об этом видео потоке существует и *Flussonic Media Server* не потеряет эти файлы на диске.

Система зачистки архива удалит их по расписанию.

Перепубликация

Для перепубликации потока, используйте `push` с шаблоном (`%s`):

> Мы **не** рекомендуем использовать `push` по протоколу UDP, поскольку это может привести к коллизии.

```
template pushed {
  prefix pushed;
  input publish://;
  push rtmp://CDN-SERVER:1936/mylive/%s;
}
```

При такой настройке для перепубликации потока `pushed/mystream` будет использоваться следующий адрес: `rtmp://CDN-SERVER/client43/pushed/mystream`.

Переключение источников, timeout

Правила переключения источников в соответствии с их приоритетом и состоянием (работает поток или нет) распространяются и на публикуемые источники. Поэтому можно добавить в поток с публикуемым источником другие, альтернативные, источники и переключаться на них по timeout.

Если источник публикации недоступен, по умолчанию Flussonic сразу переключается на следующий источник. Но можно и указывать свой timeout.

Пример с несколькими источниками и timeout источников:

```
stream published {
    source_timeout 3;
    input publish://;
    input file://vod/bunny.mp4;
}
```

Timeout можно указать и для каждого источника:

```
stream published {
    input publish:// source_timeout=3;
    input file://vod/bunny.mp4 source_timeout=2;
}
```

Запрещение публикации и событие publish_forbidden в логах Когда источник `input publish://` ниже приоритетом, чем остальные указанные источники потока, то это может значить, что публикация фактически не произойдет. Вы можете запрещать публикацию и вновь разрешать ее, изменив приоритет источников. Это можно сделать и во время трансляции.

Если публикация невозможна, Flussonic генерирует событие `publish_forbidden`. Например, это событие возникнет при следующей конфигурации, если файл `stub.mp4` существует и успешно проигрывается:

```
stream published {
    source_timeout 3;
    input file://vod/bunny.mp4;
    input publish://;
}
```

Чтобы разрешить публикацию, поместите источник публикации перед файлом.

10.2.5 Детекция тишины

Обнаружение тишины может быть полезно для целей тестирования, например, если вам нужно проверить свое аудиооборудование на работоспособность. Для этого хорошо иметь активный работающий источник видеопотока и возможность определять, когда в нем возникает тишина.

Flussonic позволяет включить обнаружение тишины в потоке и указать пороговое значение уровня звука. Flussonic генерирует события, чтобы сообщить вам, когда наступает тишина и когда звук появляется снова. События генерируются только для активных источников, а не для потерянных. Когда потерянный источник появляется снова, Flussonic возобновляет обнаружение тишины.

Если поток содержит несколько звуковых дорожек, Flussonic использует первую из них для обнаружения тишины.

Чтобы включить обнаружение тишины в потоке:

- Откройте файл конфигурации Flussonic.
- Добавьте опцию `silencedetect` в конфигурацию потока:

Здесь:

* `duration` (в секундах) — продолжительность непрерывного интервала времени, в течение которого тишина должна длиться для того, чтобы Flussonic сгенерировал соответствующее событие. * `interval` (в секундах) — Flussonic будет продолжает отправлять событие `audio_silence_dete` периодически один раз в указанный интервал времени, пока звук не появится в источнике. * `noise` — пороговое значение уровня звука. Звук такого и более низкого уровня Flussonic станет считать тишиной.

Конфигурация в примере означает, что если в течение 20 секунд уровень звука не превышает -30 дБ, то Flussonic будет генерировать событие `audio_silence_detected` каждые 10 секунд до тех пор, пока звук не появится.

- Подпишитесь на события `audio_silence_detected` и `audio_silence_end`, например:

Здесь:

* `audio_silence_detected` — это событие генерируется, когда уровень звука не превышает значение, указанное в `noise` в течение времени, указанного в `duration`. * `audio_silence_end` — это событие генерируется, когда звук снова появляется в источнике.

10.2.6 Копирование потоков

Flussonic позволяет создавать копию потока с помощью источника типа `copy://`. Эта функция необходима в случаях, когда невозможно или слишком дорого поддерживать несколько подключений к источнику сигнала:

- Для плат захвата SDI. Flussonic подключается к таким платам напрямую, и не может установить больше одного подключения. Если у вас только одна плата захвата SDI, а вы хотите провести нагрузочные тесты и посмотреть, как будет себя вести система при большом количестве потоков, то можно использовать опцию `copy://` для эмуляции большего количества источников, чем у вас есть на самом деле.
- Для RTSP-источников, например, IP-камер. Соединение с RTSP-источником устанавливается по принципу Unicast. Если вам нужно получать поток с камеры несколько раз, то на каждую копию нужно будет отдельное подключение, а это повышает нагрузку на сеть. Благодаря опции `copy://` вы можете получить поток один раз и "размножить" его уже на сервере Flussonic.
- Так же вы можете скопировать только определённый набор дорожек в стрим `copy://original?filte`

С остальными вариантами источников, как правило, существует техническая возможность получать несколько копий сигнала без использования специальных опций, например, никак не ограничивается количество приемов мультикаста или сигнала со спутника.

Пример использования опции `copy://` с платой Decklink для проверки производительности транскодера:

```
stream s {
  input decklink://0;
}

stream s1 {
  input copy://s;
  transcoder vb=1000k ab=64k external=false;
}
stream s2 {
  input copy://s;
  transcoder vb=1000k ab=64k external=false;
}
stream s3 {
  input copy://s;
  transcoder vb=1000k ab=64k external=false;
}
```

10.2.7 Кластер

Кластер — группа из нескольких серверов, соединенных вместе, чтобы выполнять работу, которая не может быть выполнена одним сервером.

Flussonic Media Server поддерживает несколько режимов для объединения серверов в кластер.

Учтите, что нет такого понятия как «просто кластер». Вы всегда должны понимать что именно вы хотите получить, устанавливая несколько серверов: это может быть уменьшение простоя в случае отказа, увеличение общей пропускной способности сервиса или объединение хранилищ между серверами.

Далее описано, как Flussonic Media Server может помочь вам.

Сценарии

Flussonic Media Server может быть настроен для:

- Захвата потоков на одном или нескольких серверах и автоматический рестрим их на другом сервере (немедленный failover).
- Моментальный доступ через рестример к потокам опубликованным на источнике.
- Удаленный доступ к DVR, записанному на сервере источнике.
- Захват и транскодирование потоков группой серверов с автоматическим резервированием.
- Автоматическая и управляемая балансировка клиентов между серверами в группе пиров.

Терминология

Здесь мы объясняем терминологию, которая поможет вам не запутаться:

Кластер

Группа из нескольких серверов с установленным Flussonic Media Server, чтобы работать вместе в одном сервисе.

Источник

Flussonic Media Server, который уже захватывает (или может начать захватывать потоки «по запросу») и может использоваться как источник для сервера рестримера.

Рестример

Flussonic Media Server, который может получать (или уже получает) потоки с одного или нескольких источников.

Рестриминг

Конфигурация группы источников и рестримеров позволяющая автоматически получить на рестримере live-потоки и DVR, записанный на источнике.

Пир

Flussonic Media Server который находится рядом с другим Flussonic Media Server. Только один из них может захватить некоторые потоки, в случае, если двойной захват очень дорог. Например, когда поток — это IP-камера, которая подключена через плохое соединение.

Кластерный захват

Конфигурация группы Flussonic Media Server, гарантирующая что каждый поток будет захвачен лишь единожды. Если один из пиров откажет, другие пиры начнут захватывать его потоки.

Рестриминг

Прочтите статью о кластерном рестриминге.

DVR в кластере

Узнайте про доступ к DVR в кластере из отдельной статьи.

Кластерный захват и транскодирование

Узнайте, как настроить отказоустойчивый захват и транскодирование с помощью механизма cluster ingest.

Перенаправление на пиры

Flussonic может направлять клиентов на соответствующий пир используя механизм кластерного пиринга.

Управление конфигурацией потоков извне

Flussonic даёт возможность управлять конфигурацией потоков извне с помощью механизма `config_external` и внешнего по отношению к *Flussonic* конфигурационного бэкенда. Подробнее см. Управление конфигурацией потоков извне: конфигурационный бэкенд и механизм config_external.

10.2.8 Ретрансляция потоков

Flussonic Media Server (рестример) может подключиться к другому *Flussonic Media Server* (источнику), получить список запущенных и доступных по запросу потоков, а затем рестримить их. Вы также получаете прозрачный доступ к архиву на источнике.

Flussonic Media Server позволяет указать несколько серверов-источников и построить отказоустойчивую конфигурацию.

На этой странице:

- Отличия Flussonic Media Server от HTTP-прокси
- Настройка рестриминга
- Дополнительные настройки
- Настройка нескольких источников
- Обзор протокола M4F

Отличия от HTTP-прокси

Многие CDN предлагают решение проблемы доставки видео, используя кластер из обычных HTTP-прокси серверов, которые кешируют сегменты HLS-потока и доставляют их до пользователей.

В отличие от HTTP-прокси, *Flussonic Media Server*, установленный на все серверы в сети, предоставляет следующие возможности:

- Just-in-time упаковка на рестримере. Получая поток по одному протоколу, рестример может переупаковать этот поток в доступные протоколы на выходе, например, HLS, DASH, RTMP, RTSP, HTTP MPEG-TS или UDP MPEG-TS.
- Единая авторизация пользователей по всем доступным протоколам.
- Централизированная агрегация сессий и сбор статистики.

Главное отличие рестриминга с помощью HTTP-прокси и *Flussonic Media Server* в том, что с *Flussonic Media Server* вы доставляете видео между серверами лишь раз, а на выходе получаете всю функциональность *Flussonic* на рестриминг-сервере. Это нереализуемо на HTTP-прокси, потому что прокси не работает с видео на низком уровне.

Настройка рестриминга

Чтобы включить рестриминг в кластере *Flussonic Media Server*, вам нужно использовать следующие директивы:

- `source` — чтобы указать сервер, с которого вы хотите рестримить видео.
- `cluster_key` — чтобы *Flussonic Media Server* мог забрать видео через кластерную авторизацию.

Директива `source` имеет следующий синтаксис и опции:

```
cluster_key abcd;
source streamer:8081 {
}
```

Вы должны установить одинаковые `cluster_key` на источнике и рестримере. Кластерный ключ очень важен и его нужно хранить в секрете, потому что он может быть использован для настройки удаленного сервера. Он передается в виде хэша.

Директива `source` включает автоматический захват потоков с сервера источника. В ней потоки можно разделить на несколько списков:

- белый список — эти потоки будут статическими (`static`) на рестримере;
- серый список — эти потоки будут доступны на рестримере по запросу (`ondemand`);
- черный список — эти потоки не будут видны на рестримере.

По умолчанию все работающие потоки с источника попадают в белый список, а все потоки по запросу с источника будут в сером списке рестримера.

Потоки, указанные в опции `except`, помещаются в черный список. Эта опция имеет более высокий приоритет, чем `only`.

```
cluster_key abcd;
source streamer:8081 {
  except stream1 football;
}
```

Опция `only` разделяет доступные потоки (кроме тех, что в черном списке) на белый и серый списки: `only` – в белый список, остальные не будут статическими и будут ждать запроса для запуска.

Если на сервере-источнике присутствует поток с тем же именем, что и настроенный локально на сервере-рестримере или публикуемый на сервер-рестример, то будет использоваться поток, настроенный локально. Поток с источника будет игнорироваться.

```
cluster_key abcd;
source streamer:8081 {
  only cbc football stream2;
}
```

С опциями `only` и `except` можно использовать шаблон поиска по имени потока. Эта возможность упрощает конфигурацию, когда используются live-локации на удаленном сервере, и заранее неизвестны имена потоков. Например, такая конфигурация позволяет захватить все потоки с источника, имена которых начинаются на `mylive/`:

Figure 10.11: Flussonic cluster restreaming

```
cluster_key abcd;
source streamer:8081 {
  only mylive/*;
}
```

Дополнительные настройки

Вы можете применить настройки сразу для всех потоков, запущенных через `source`:

```
cluster_key abcd;
source streamer:8081 {
  on_play http://IP-ADDRESS:PORT/php-auth-script.php;
```

Figure 10.12: Flussonic cluster restreaming

```
  backup vod/bunny.mp4;
  dvr /storage 2d 97%;
}

source test2:8082 {
  on_play http://IP-ADDRESS:PORT/php-auth-script.php;
  backup vod/bunny.mp4;
  dvr /storage 2d 97%;
}
```

Такая конфигурация автоматически применится для всех потоков, запущенных на рестримере.

Если у вас на основном сервере используется опция `backup`, то вам следует загрузить этот файл на рестример и указать путь:

```
source origin {
  backup vod/bunny.mp4;
}
```

Если вам необходимо запретить проигрывание по некоторым протоколам, вы можете использовать опцию ‘protocols‘ внутри директивы `source`:

```
source origin {
  protocols -dash -hls -rtmp;
}
```

В приведенном выше примере доступ к контенту по протоколам DASH, HLS и RTMP запрещён, так что зритель не сможет проиграть поток по этим протоколам.

Чтобы узнать все опции и настройки доступные для `source`, обратитесь к 7Csourcestag/config/operation/config get/response%7CsourcesFlussonic API Reference.

Настройка нескольких источников

Flussonic позволяет указать несколько источников на рестримере. Если у нескольких источников будут одинаковые имена потоков, такой поток будет настроен с несколькими URL. Таким образом, если первый источник упадет или поток на нём пропадет, рестример будет получать поток с другого источника.

При настройке нескольких источников с помощью кластерного захвата, вы можете обеспечить отказоустойчивый кластер.

Обзор протокола M4F

Flussonic Media Server по умолчанию использует для рестриминга свой внутренний сегментный протокол M4F.

M4F имеет следующие преимущества:

- Синхронизация потоков между источником и рестримером.
- Сохранение одинаковых временных меток (timestamp) кадров у потоков из источника и из рестримера.
- Получение идентичных полезных данных (payload) при запросе к сегменту потока как из источника, так и из рестримера. Полезные данные идентичны на всех серверах. Все серверы возвращают совместимый ответ.
- Временные метки в UTC.
- Сохранение структуры сегментов потока за счёт побайтовой копии потока из источника для всех протоколов на рестримере;

- Сохранение нумерации сегментов на источнике и рестримере.
- Наличие единой байт-структуры для передачи потоков между серверами и хранения архива. Это обеспечивает идентичность данных передаваемых между серверами по M4F и тех, что записываются в архив.
- Оповещение рестримера о появлении новых сегментов с помощью механизма push-уведомлений. Рестример получает уведомление о наличии нового сегмента и скачивает его по отдельному каналу в меру своих возможностей.
- Предоставление информации об архиве источника рестримеру.

При передаче по протоколу M4F обеспечивается точная передача времени и данных. Протокол поддерживает все те же кодеки, что и *Flussonic*.

В сравнении с M4F другие протоколы имеют следующие недостатки:

- RTMP ломает временные метки.
- Устройство RTMP и MPEG-TS осложняет синхронизацию временных меток при передаче потоков между серверами.
- RTMP и MPEG-TS не имеют способов синхронизации временных меток потока с реальным текущим временем.
- RTSP имеет механизм синхронизации временных меток потока с реальным текущим временем, но вызывает проблемы с доставкой B-кадров и некоторых кодеков.

M4F даёт абсолютное время каждого кадра с высокой точностью.

10.2.9 Схемы резервирования N+1, N+M в Flussonic Media Server

Обзор схем резервирования

Использование схем резервирования сервисов особенно важно в индустрии IPTV для обеспечения доступности и надежности системы. Простои могут привести к потере доходов и недовольству клиентов, поэтому необходимо внедрять схемы резервирования для минимизации риска сбоев.

Flussonic Media Server включает в себя функциональность кластерного захвата, которая позволяет внедрять различные схемы резервирования. В этой статье мы рассмотрим, как использовать функцию кластерного захвата для настройки схем резервирования в Flussonic, включая схемы N+1 и N+M.

Схемы резервирования N+1 и N+M — это два распространенных типа схем резервирования сервисов, которые могут использоваться для обеспечения надежности и доступности системы. В схеме N+1 имеется одно дополнительное устройство сверх минимально необходимого количества для выполнения требуемых задач, а в схеме N+M имеется M дополнительных устройств сверх минимально необходимого количества.

Одним из ключевых различий между схемами N+1 и N+M является уровень резервирования, который они обеспечивают. Схемы N+1 обеспечивают более низкий уровень избыточности, поскольку в них имеется одно дополнительное устройство, которое может быть использовано в качестве резервного в случае отказа. Схемы N+M, с другой стороны, обеспечивают более высокий уровень избыточности, поскольку они имеют M дополнительных устройств, которые могут быть использованы в качестве резервных, но если более одного из этих устройств выйдет из строя, система все равно будет подвержена риску отказа.

Еще одно различие между схемами N+1 и N+M заключается в стоимости и сложности реализации. Схемы N+1, как правило, более экономичны и просты в реализации, поскольку для них требуется только одно дополнительное устройство. Схемы N+M, с другой стороны, требуют M дополнительных устройств, что может увеличить стоимость и сложность системы.

Итак, в каких случаях следует использовать схему N+1, а в каких — схему N+M? Это действительно зависит от ваших конкретных нужд и требований. Если вам нужен высокий уровень избыточности и вы готовы заплатить более высокую цену и согласиться на более сложную систему, схема N+M может быть хорошим выбором. С другой стороны, если вы ищете более экономичное и простое решение, схема N+1 может оказаться более подходящей.

Помимо схем резервирования N+1 и N+M, существует еще несколько типов схем резервирования, которые могут быть использованы для обеспечения надежности и доступности системы. К ним относятся:

- Горячий резерв: в системе горячего резерва одно устройство используется в качестве основного, а второе — в качестве резервного. Резервное устройство находится в состоянии ожидания и готово приступить к работе, если основное устройство выйдет из строя. Этот тип схемы резервирования часто используется в критически важных системах, где важно минимизировать время простоя и обеспечить бесперебойную работу системы.
- Холодный резерв: в системе холодного резерва резервное устройство находится в спящем

состоянии и активируется только в случае отказа основного устройства. Этот тип схемы резервирования обычно дешевле в реализации, чем система горячего резервирования, поскольку резервное устройство не находится в состоянии ожидания и не требует столько обслуживания или ресурсов. Однако для активации резервного устройства в случае сбоя может потребоваться больше времени, что может привести к увеличению времени простоя.

- Кластеризация: в кластерной системе несколько устройств работают вместе для выполнения одних и тех же задач и могут автоматически взять на себя управление, если одно из устройств выходит из строя. Этот тип схемы резервирования часто используется в системах, где важно обеспечить высокую доступность и способность продолжать работу, даже если одно из устройств выходит из строя.

Введение в функциональность кластерного захвата в Flussonic и как ее можно использовать для настройки схем резервирования

Функция кластерного захвата в Flussonic Media Server позволяет конфигурировать несколько серверов Flussonic или устройств Flussonic Coder, которые работают вместе как кластер для выполнения одних и тех же задач. Каждый сервер в кластере называется ”пиром”, и каждый пир может быть настроен на выполнение определенных задач, таких как транскодирование, запись или передача видео на другие серверы.

Если один из пиров в кластере выходит из строя, потоки, которые обрабатывались на этом пире, будут автоматически перераспределены среди остальных пиров. Это помогает обеспечить бесперебойную работу системы даже в случае сбоя одного из пиров.

Функциональные возможности кластера можно использовать для настройки различных типов схем резервирования, включая схемы N+1 и N+M. В схеме N+1 имеется одно дополнительное устройство сверх минимального количества, необходимого для выполнения требуемых задач, а в схеме N+M — M дополнительных устройств. Настроив несколько пиров в кластере и используя опции и , вы можете настроить схему резервирования N+1 или N+M в Flussonic.

Подробные примеры настройки схемы резервирования N+1 и N+M с использованием функциональности кластерного захвата в Flussonic

Чтобы настроить схему резервирования с помощью функции кластерного захвата Flussonic, выполните следующие шаги:

- Настройте кластер серверов в Flussonic. Для этого необходимо указать один и тот же ключ кластера в файле конфигурации для всех серверов кластера, а также указать список других серверов кластера для каждого сервера.
- Используйте опцию в конфигурации потока, чтобы указать, что поток должен обрабатываться кластером серверов.
- Используйте опцию , чтобы указать основной сервер для потока. Если основной сервер выйдет из строя, один из других серверов в кластере автоматически возьмет на себя его

роль.

Вот пример конфигурации, которая устанавливает схему резервирования N+1 с использованием функции кластерного захвата Flussonic:

```
cluster_key abcd;
peer transcoder1;
peer transcoder2;
peer transcoder3;

stream mystream1 {
    url rtmp://source.com/live/mystream1;
    cluster_ingest;
    capture_at transcoder1;
}
stream mystream2 {
    url rtmp://source.com/live/mystream2;
    cluster_ingest;
    capture_at transcoder2;
}
```

Эта конфигурация устанавливает схему резервирования N+1, используя функцию захвата кластера Flussonic. В этой конфигурации в кластере имеется три сервера: transcoder1, transcoder2 и transcoder3. Каждый из этих серверов настроен на обработку всех потоков в системе.

Опция используется для указания одного и того же ключа кластера для всех серверов в кластере. Этот ключ используется для установления связи между серверами и обеспечения их совместной работы как единого целого.

Опции используются для указания других серверов в кластере для каждого сервера. Это позволяет серверам общаться друг с другом и координировать свою деятельность как единое целое.

Опции используются для указания потоков, которые должны обрабатываться кластером. Опция указывает, что поток должен обрабатываться кластером серверов, а опция указывает первичный сервер для потока. Если основной сервер выйдет из строя, один из других серверов в кластере автоматически возьмет на себя ответственность и продолжит обработку потока.

Важно отметить, что эта конфигурация должна быть идентичной на всех серверах в кластере. Это гарантирует, что все серверы правильно настроены для работы вместе как единое целое и могут взаимодействовать друг с другом.

В этой конфигурации кластер захватывает и обрабатывает два потока: mystream1 и mystream2. Каждый поток обрабатывается определенным основным сервером, как указано в параметре . Например, mystream1 обрабатывается транскодером1, а mystream2 — транскодером2.

Если один из основных серверов выходит из строя, то один из других серверов в кластере автоматически принимает на себя управление и продолжает обработку потоков, которые обрабатывались вышедшим из строя сервером. Например, если выйдет из строя транскодер1, то, скорее всего, его место займет транскодер3, и продолжит обработку видео потока mystream1, так как транскодер2, уже занят обработкой mystream2.

В данном примере для простоты используются только два потока, но та же концепция применима к любому количеству потоков. В реальной системе кластер может обрабатывать сотни или даже тысячи потоков, распределяя их между десятками серверов. Ключевой принцип заключается в том, что каждый поток обрабатывается определенным основным сервером, и если этот сервер выходит из строя, один из других серверов в кластере автоматически берет его на себя и продолжает обработку потока.

Используя функциональность кластерного захвата Flussonic, вы можете легко настроить схему резервирования для обеспечения надежности и доступности ваших потоков, независимо от количества потоков или серверов в системе.

Советы по устранению неполадок и отладке проблем со схемой резервирования

При настройке схемы резервирования с помощью Flussonic Media Server важно быть готовым к возможным проблемам и знать, как их устранить и отладить. Ниже приведены некоторые советы по устранению неполадок и отладке проблем со схемой резервирования:

- Проверьте лог-файлы. Flussonic Media Server регистрирует подробную информацию о своей деятельности, включая любые ошибки или проблемы, которые могут возникнуть. Просмотрев логи, вы сможете определить причину любых проблем и предпринять шаги по их устранению.
- Протестируйте процесс резервирования. Чтобы убедиться, что процесс резервирования функционирует должным образом, вы можете имитировать сбой, выключив один из серверов в кластере и проверив, что другие серверы принимают на себя управление и продолжают обрабатывать потоки, как и ожидалось.
- Проверьте конфигурацию сети. Убедитесь, что серверы в кластере правильно настроены и могут взаимодействовать друг с другом по сети. Это особенно важно, если вы используете различные сетевые интерфейсы или виртуальные локальные сети для кластера.
- Просмотрите конфигурацию. Убедитесь, что конфигурация кластера и потоков верна и соответствует желаемой схеме резервирования. Дважды проверьте параметры , и , чтобы убедиться, что они установлены правильно.
- Если вы не можете решить проблему самостоятельно, обратитесь за помощью в службу технической поддержки Flussonic. Они смогут предоставить квалифицированный совет и руководство, чтобы помочь вам решить проблему. Дополнительную информацию о том, как связаться с технической поддержкой Flussonic, можно найти на странице Поддержка.

Следуя этим советам, вы сможете эффективно устранить неполадки и отладить любые проблемы, которые могут возникнуть со схемой резервирования в Flussonic Media Server.

Резюме ключевых моментов и лучших практик для настройки схемы резервирования с помощью Flussonic

Ниже приведено краткое описание ключевых моментов и лучших практик для настройки схемы резервирования с помощью Flussonic Media Server:

- Настройте кластер серверов в Flussonic, указав один и тот же ключ кластера в файле конфигурации для всех серверов и указав список других серверов в кластере для каждого сервера.
- Используйте опцию в конфигурации потока, чтобы указать, что поток должен обрабатываться кластером серверов.
- Используйте опцию , чтобы указать первичный сервер для потока. Если основной сервер выйдет из строя, один из других серверов в кластере автоматически возьмет на себя его роль.
- Убедитесь, что каждый сервер в кластере имеет полный и точный список других серверов в кластере. Это необходимо для того, чтобы серверы могли взаимодействовать и работать вместе как единое целое.
- Выберите подходящую схему резервирования, исходя из ваших потребностей и требований системы. Варианты включают N+1, N+M, кластер и горячий резерв (1+1).
- Протестируйте и проконтролируйте схему резервирования, чтобы убедиться, что она работает правильно и обеспечивает необходимый уровень надежности и доступности.

Следуя этим рекомендациям, вы сможете эффективно настроить схему резервирования с помощью Flussonic Media Server для обеспечения надежности и доступности ваших потоков.

10.3 Playback

10.3.1 Проигрывание

Проигрывание потоков

Flussonic Media Server позволяет проигрывать потоки по разным протоколам. Чтобы посмотреть список ссылок для проигрывания, щелкните имя потока на вкладке **Media** — **Streams** и перейдите на вкладку **Output**. Вы можете скопировать тот или иной URL в буфер обмена, нажав кнопку копирования в конце строки соответствующего URL-адреса.

Figure 10.13: Output protocols

Ниже приведено подробное описание URL-адресов, которые следует использовать в плеерах для воспроизведения видео по различным протоколам, со ссылками на разделы о настройке воспроизведения через каждый протокол. Проиграть потоки по некоторым протоколам можно также в плеере предпросмотра прямо во *Flussonic UI*.

Кроме того, вы можете управлять проигрыванием потоков с помощью Streaming API.

embed.html Адрес: `http://FLUSSONIC-IP/STREAMNAME/embed.html`

В Flussonic Media Server есть специальная страница — **embed.html**, которая предназначена для вставки видео на сайт или просмотра видео через браузер. Страница автоматически определяет

браузер и выбирает поддерживаемый протокол. Для большинства устройств на сегодня — это HLS. Подробнее в статье Вставка видео на сайт (embed.html).

Полная **OpenAPI** спецификация: Streaming API.

HLS Адрес для плеера: `http://FLUSSONIC-IP/STREAMNAME/index.m3u8`

Подробнее в статье Воспроизведение HLS. Для вставки на сайт используйте (embed.html) или любой сторонний плеер. Например, hls.js или clappr.

Полная **OpenAPI** спецификация: Streaming API.

DASH Поток доступен по адресу `http://FLUSSONIC-IP/STREAMNAME/index.mpd`

Подробнее в статье Воспроизведение DASH.

Полная **OpenAPI** спецификация: Streaming API.

MSE-LD Адрес для плеера: `ws://FLUSSONIC-IP/STREAMNAME/mse_ld`

HTML5 (MSE-LD) Поток, проигрываемый по HTML5, доступен по адресу: `http://FLUSSONIC-IP/STREAM`

Подробнее в статье HTML5 (MSE-LD) воспроизведение с низкой задержкой.

MSS Поток доступен по адресу: `http://FLUSSONIC-IP/STREAMNAME.isml/manifest`

Подробнее в статье Воспроизведение MSS.

Полная **OpenAPI** спецификация: Streaming API.

HTTP MPEG-TS Поток доступен по адресу: `http://FLUSSONIC-IP/STREAMNAME/mpegts`

HTTP MPEG-TS с относительным таймшифтом URL-адрес для проигрывания HTTP MPEG-TS с относительным таймшифтом:

`http://FLUSSONIC-IP:PORT/STREAM_NAME/timeshift_rel-3600.ts`

В этом примере записанный поток будет проигрываться с задержкой в 1 час (3600 секунд).

HTTP MPEG-TS с абсолютным таймшифтом URL-адрес для проигрывания HTTP MPEG-TS с абсолютным таймшифтом:

`http://FLUSSONIC-IP:PORT/STREAM_NAME/timeshift_abs-1643257722.ts`.

Фрагмент архива можно получить на полной скорости, но в режиме стриминга, через промежуток времени равный длине фрагмента.

RTMP Поток доступен по адресу:

- `rtmp://FLUSSONIC-IP/static/STREAMNAME`

RTSP Поток доступен по адресу:

- `rtsp://FLUSSONIC-IP/STREAMNAME`

Если у потока есть несколько аудио- и видеодорожек или дорожек с субтитрами, то вы можете выбрать какие дорожки проигрывать с помощью параметра 'filter.tracks'.

Примеры:

- `rtsp://FLUSSONIC-IP/STREAMNAME?filter.tracks=a2v1`
- `rtsp://FLUSSONIC-IP/vod/file?filter.tracks=a2v1` — VOD.
- `rtsp://FLUSSONIC-IP/STREAMNAME2 = rtsp://FLUSSONIC-IP/STREAMNAME1?filter.tra`

Можно выбрать одну дорожку:

- `rtsp://FLUSSONIC-IP/STREAMNAME?filter.tracks=a1` — выбрать только аудио.
- `rtsp://FLUSSONIC-IP/STREAMNAME?filter.tracks=v1` — выбрать только видео.

WebRTC Поток по протоколу WebRTC WHEP доступен по адресу:

- `http://FLUSSONIC-IP/STREAM_NAME/whep`

Подробнее о WebRTC плеере и организации проигрывания в статье WebRTC проигрывание.

Полная **OpenAPI** спецификация: Streaming API.

SHOUTcast Flussonic Media Server умеет отдавать SHOUTcast или ICEcast радиопоток.

Поток доступен по адресу: `http://FLUSSONIC-IP/STREAMNAME/shoutcast`

The complete **OpenAPI** specification: Streaming API.

SRT *Flussonic* поддерживает проигрывание SRT-потоков.

В общем случае адрес для проигрывания SRT-потока выглядит так:

`srt://FLUSSONIC-IP:SRT_PORT?streamid=#!::r=STREAM_NAME,m=request`

Подробнее о протоколе SRT, `streamid` и других поддерживаемых параметрах см. Использование протокола SRT.

Варианты URL для проигрывания SRT описаны на странице Воспроизведение SRT.

Выбор дорожек для проигрывания

Если у потока есть несколько аудио- и видеодорожек, а также дорожек с субтитрами, то вы можете указать, какие дорожки следует отдавать для проигрывания.

Для этого укажите номера дорожек, добавив параметр `?filter.tracks=[aN|vN|tN|lN]+`, за которым следует один или несколько номеров дорожек без пробела, в конце URL потока. Ниже указаны возможные значения:

- `aN` — N-ая аудиодорожка,
- `vN` — N-ая видеоодорожка,
- `tN` — N-ая дорожка с WebVTT субтитрами,
- `lN` — N-ая дорожка с DVB-субтитрами или телетекстом.

По умолчанию *Flussonic* выбирает первую аудио- и видеодорожку (a1v1). Если указать больше двух дорожек или указать дорожку в неверном формате, то используется значение по умолчанию (a1v1).

Примеры:

- `rtsp://FLUSSONIC-IP/STREAMNAME?filter.tracks=a2v1` — вторая аудиодорожка и первая видеодорожка для RTSP-потока.
- `http://FLUSSONIC-IP/STREAM_NAME/index.m3u8?filter.tracks=v2` — вторая видеодорожка для HLS-плейлиста.
- `http://FLUSSONIC-IP/STREAMNAME.isml/manifest?filter.tracks=v1t1t2t3` — первая видеодорожка и три дорожки с субтитрами для MSS-манифеста.

Запрет использования протоколов

По умолчанию разрешено проигрывание по всем протоколам, но вы можете запретить определенные протоколы (**Except**) или разрешить только определенные протоколы (**Only**) с помощью переключателя

рядом со ссылками для проигрывания на вкладке **Output**. Эта функция полезна не только для обеспечения безопасности, но и для снижения нагрузки на сервер, т.к. упаковка во все доступные протоколы может потреблять много ресурсов.

Эти настройки можно задать и через файл конфигурации. Например, для потока `channel_01` можно разрешить проигрывание по всем протоколам, кроме MPEG-TS и HLS (соответствует переключателю в положении **Except**).

```
stream channel_01 {
  protocols -mpegts -hls;
}
```

Для потока `channel_02` разрешить только проигрывание по DASH и запросы API, при этом запретив проигрывание по всем остальным протоколам (соответствует переключателю в положении **Only**):

```
stream channel_02 {
  protocols dash api;
}
```

Получение данных о проигрываемом потоке

Вы можете отправлять запросы API для получения информации о проигрываемом потоке. Полученные данные можно интегрировать в любую внешнюю систему, например, сайт, мониторинг, плеер или мобильное приложение.

Адрес для получения технической информации о проигрываемом потоке:

`http://FLUSSONIC-IP/STREAM_NAME/media_info.json`

Адрес для получения информации о статусе записи архива DVR потока:

`http://FLUSSONIC-IP/STREAM_NAME/recording_status.json`

Плеер предпросмотра во Flussonic UI

Вы можете проиграть поток прямо во *Flussonic UI* с помощью плеера предпросмотра. Этот плеер позволяет проигрывать потоки по протоколу HLS, MSE-LD или DASH. Кроме того, в нем можно проиграть архив DVR, если DVR включен для потока. Плеер предпросмотра использует специальную страницу **embed.html** (подробнее см. Вставка видео на сайт) с соответствующими параметрами.

Чтобы открыть плеер предпросмотра, перейдите в раздел **Media** > **Streams** и нажмите кнопку **Play** рядом с нужным потоком.

В открывшемся окне выберите нужную вкладку и начните проигрывание. Доступны следующие вкладки:

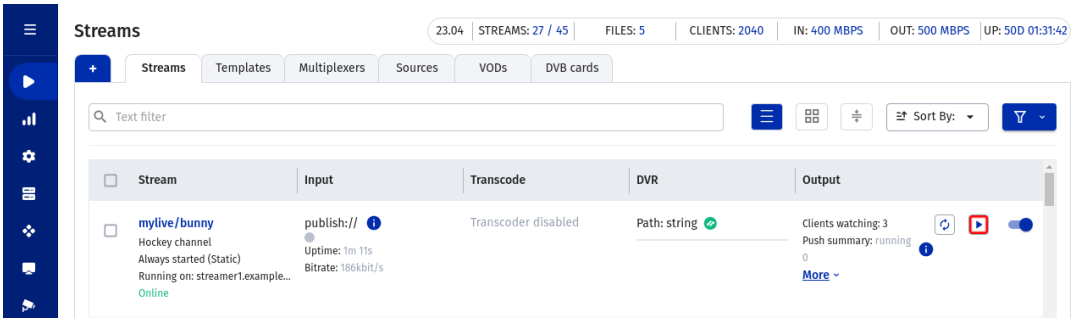


Figure 10.14: Кнопка плеера предпросмотра

- **HLS** — для проигрывания по HLS. Плеер будет использовать страницу `embed.html`.
- **MSE** — для проигрывания по HTML (MSE-LD) с низкой задержкой. Плеер будет использовать страницу `embed.html?realtime=true`.
- **DASH** — для проигрывания по DASH. Плеер будет использовать страницу `embed.html?proto=dash`.
- **DVR** — для проигрывания архива DVR (если DVR включен для потока). Плеер будет использовать страницу `embed.html?dvr=true`. Подробнее о настройках DVR плеера читайте в главе Просмотр записей архива из административного веб-интерфейса.

Внизу страницы плеера предпросмотра отображается HTML код для вставки соответствующего плеера на веб-страницу.

Чтобы закрыть окно плеера предпросмотра, нажмите **Esc**.

10.3.2 Отправка потока на другие серверы

Копирование потока на другие серверы (push)

Flussonic Media Server может принудительно копировать поток на другие серверы. Например, можно копировать поток в CDN или в социальные сети.

Чтобы использовать эту функцию, в настройках потока перейдите на вкладку **Output** и промотайте страницу вниз до раздела **Push live video to certain URLs**. Здесь вы можете добавить ссылки, на которые *Flussonic* будет отправлять поток.

Figure 10.15: Flussonic push options

Flussonic поддерживает отправку потоков по следующим протоколам:

RTMP Этот протокол обычно используется для публикации видео в социальных сетях. Примеры и особенности настройки описаны здесь.

HTTP MPEG-TS Этот протокол подойдет для отправки потоков в сторонние видеостриминговые сервисы. Ссылка для публикации должна быть предоставлена сервисом, в который вы осуществляете публикацию.

HLS По этому протоколу можно отправлять потоки в облако или в CDN, т.к. обычно CDN его поддерживают. Ниже приведен пример конфигурации для отправки потока в облако Amazon AWS.

Копирование потока в облако Amazon AWS

```
stream breakingnews {
  input publish://;
  segment_count 10;
  segment_duration 10;
  push hlss://[api-id].execute-api.[aws-region].amazonaws.com/[stage]/[
    folder-name];
}
```

Эта конфигурация позволит вам использовать Amazon AWS API Gateway в качестве сервисного прокси для Amazon S3, как описано в документации Amazon.

M4S Используйте этот протокол для отправки потоков на другой сервер Flussonic. Это потоковый протокол, который не создает задержки и нужен для того, чтобы передать данные с Flussonic на Flussonic для дальнейшей передачи по WebRTC/RTMP.

Ссылка для M4S выглядит следующим образом: `m4s://FLUSSONIC-IP:PORT/STREAM_NAME`. На принимающем *Flussonic*'е должен быть настроен соответствующий поток для приема отправляемых данных.

Push с 302 редиректом При публикации по m4s:// Flussonic поймет HTTP 302 и последует по указанному адресу. Это значит, что можно указать не только адрес Flussonic-сервера, но и ваш собственный бекенд для выбора точки публикации. Например, `m4s://example.com/router;`.

Управление отправкой потоков на другой сервер

Если для потока настроена публикация по адресу, который стал недоступен, то по умолчанию Flussonic будет бесконечно пытаться отправить поток по этому адресу.

Flussonic может следить за состоянием отправки потоков на другие серверы, и собирать статистику по попыткам отправки. Наглядное отображение статусов отправки в UI поможет вам принимать меры — приостанавливать offline-потоки или ограничивать попытки отправить их.

Статус отправки в виде индикатора показан на главной странице в списке **Streams** и в настройках потока на вкладке **Output** (**Push live video to certain URLs**). Причину остановки отправки потока можно посмотреть в логах.

Figure 10.16: Flussonic push options

10.4 Transcode

10.4.1 Транскодер

Транскодирование необходимо для того, чтобы:

- создать мультибитрейтный поток,
- изменить параметры видео — кодек и битрейт потока, размер картинки,
- наложить логотип.

Процесс транскодирования состоит из двух этапов:

- **Декодирование** — процесс получения необработанных ”сырых” данных из сжатых.
- **Кодирование** — процесс сжатия необработанных ”сырых” данных для их последующей передачи по сети. Кодирование определяет параметры видео, такие как разрешение, битрейт и тип сжатия.

Для кодирования и декодирования необходим ***кодек*** — алгоритм сжатия видео и аудио. Видеокодек влияет на размер файла и качество изображения. H.264 и AAC — наиболее часто используемые видео- и аудиокодеки для потокового вещания.

Помните, что разные протоколы поддерживают разные контейнеры и кодеки.

Транскодирование — процесс декодирования потока, преобразования некоторых параметров видео и/или аудио, а затем повторного кодирования потока для передачи через Интернет.

Транскодирование — одна из операций над медиа либо их комбинация:

- **Транскодирование** — это изменение видео- и/или аудиокодека (типа сжатия). Например, преобразование видео MPEG2 в H.264.
- **Транссайзинг** — изменения размера видеокадра, или разрешения видео. Например, уменьшение разрешения с 3840×2160 (4K) до 1920×1080p.
- **Трансрейтинг** — процесс понижения битрейта потока без изменения кодека или разрешения. Например, у видео 4K с битрейтом 45 Мбит/с можно снизить битрейт до 15 Мбит/с.

Ниже показан пример получения мультибитрейтного потока при транскодировании видео 4K с битрейтом 45 Мбит/с:

Схема 1. Пример транскодирования

В *Flussonic Media Server* есть встроенный транскодер. Он поддерживает транскодирование на графическом процессоре и с использованием центрального процессора.

Транскодер работает со всеми видами источников, которые *Flussonic* может захватывать. Протокол HLS поддерживается частично — некоторые источники могут не транскодироваться. Работоспособность вашего источника HLS с транскодером необходимо каждый раз проверять самостоятельно.

При транскодировании на NVENC транскодер может обрабатывать потоки с 10-битной глубиной цвета.

Транскодер бесшовно переключается между источниками без потери кадров, что обеспечивает ровное проигрывание результирующего потока без мигания или других артефактов на экране зрителя. Бесшовное переключение достигается за счёт сохранения разрешения видео исходного потока в результирующем потоке. Транскодер преобразует источники таким образом, чтобы разрешение видео в них соответствовало значению, указанному в параметре 'size'. Если параметр `size` не указан, то разрешение выходного потока будет соответствовать разрешению видео самого первого источника, полученного на вход транскодера.

Содержание:

Транскодирование является крайне ресурсоемким процессом (по CPU) и включает в себя следующие этапы:

1. Декодирование исходного потока. 2. Обработка и кодирование сырого потока в соответствии с заданными параметрами.

Рассчитывайте от 5 до 20 каналов на один сервер в зависимости от настроек.

Установка транскодера

Для транскодирования на GPU Nvidia NVENC никаких дополнительных пакетов устанавливать не требуется.

Для транскодирования на центральном процессоре (CPU) нужен установленный пакет `flussonic-transcod`. При настройках системы по умолчанию он устанавливается вместе с Flussonic Media Server как recommended-пакет. Если у вас отключена установка рекомендованных пакетов, установите пакет для транскодирования вручную:

```
apt-get -y install flussonic-transcoder
```

Пакет устанавливается из того же репозитория, что и пакет `flussonic`.

Настройка транскодера

У транскодера есть множество опций, которые можно разделить на следующие основные группы:

- Глобальные опции (применяются ко всем видеотрекам)
- Опции кодирования видео (применяются индивидуально к каждому видеотреку)
- Опции кодирования аудио (применяются ко всем аудиотрекам)

Все опции подробно описаны в разделе Опции транскодера.

Вы можете настроить транскодер одним из трех способов:

- Веб-интерфейс Flussonic (recommended)
- Файл конфигурации
- Flussonic API (см. 7Ctranscodertag/stream/operation/stream-save%7CtranscoderСправочник API)

Настройка транскодера через веб-интерфейс Веб-интерфейс Flussonic позволяет настроить транскодер для потока или шаблона.

Чтобы настроить транскодер через веб-интерфейс Flussonic:

В разделе **Media** > **Streams** или **Media** > **Templates** > нажмите на имя потока или шаблона, для которого хотите настроить транскодирование. Затем перейдите во вкладку **Transcoder** и нажмите **Enable transcoder**.

Используйте стрелки справа, чтобы раскрыть или свернуть ту или иную группу настроек.

Параметры кодирования аудио

- **Copy from input** — отметьте, чтобы получить на выходе аудио с теми же характеристиками, что и на входе.
- **Bitrate** — битрейт аудио дорожки.
- **Codec** (aac|opus|mp2a|pcma|ac3) — аудио-кодек (по умолчанию используется кодек AAC).
- **Sample rate** (bypass|0|8000|16000|32000|44100|48000)
- **Channels** — количество аудиоканалов в выходном потоке.
- **Volume** — громкость звука на выходе. Это может быть значение в Дб (с ”+” или ”-”), которое будет добавлено к громкости на входе, либо коэффициент, на который нужно умножить громкость на входе. Подробнее см. Как изменить уровень громкости звука?.
- **Split channels** — выберите эту опцию, чтобы каждый аудиотрек с несколькими каналами разбивался на несколько моно-треков.

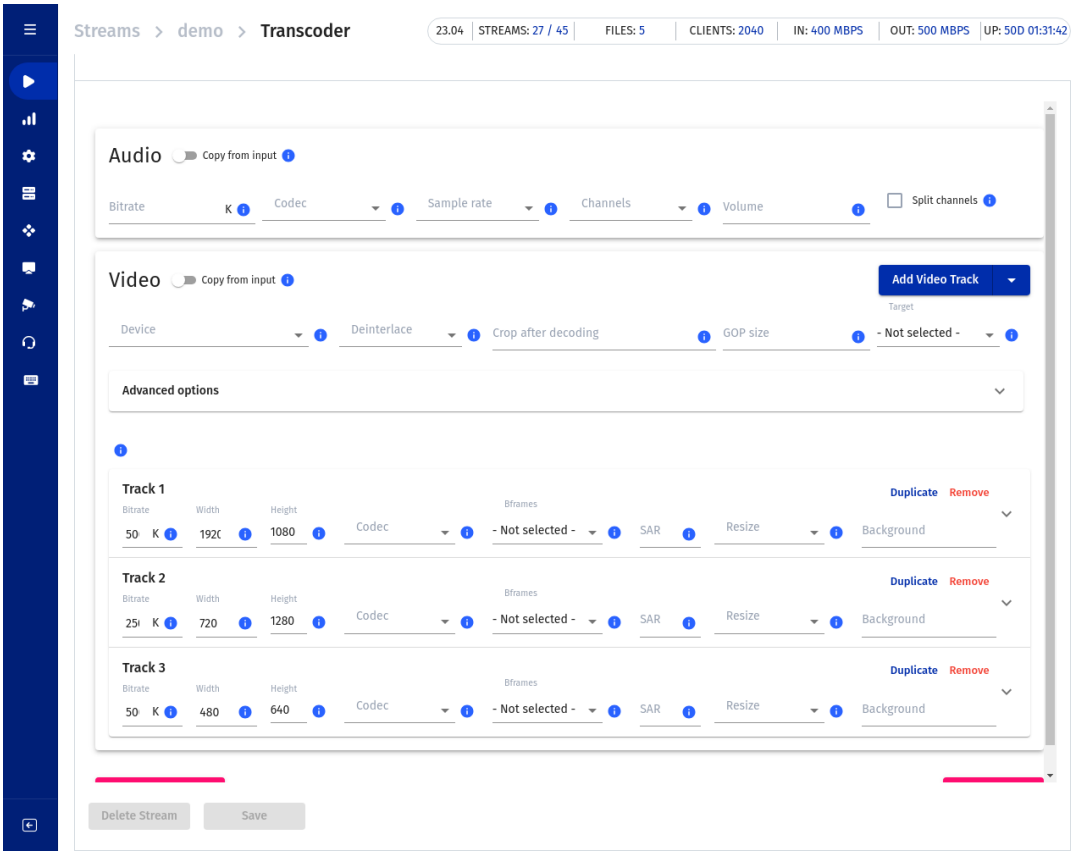


Figure 10.17: Flussonic transcoder

Глобальные настройки транскодера Глобальные настройки применяются ко всем выходным видеодорожкам.

Device — выбор вида транскодера. Для Flussonic Media Server позволяет включить аппаратное кодирование и выбрать модель и ID видеокарты NVENC, либо использовать ЦП. Аппаратное кодирование позволяет транскодировать значительно больше потоков на одном сервере. Для Flussonic Coder в этом поле вы выбираете устройство для транскодирования потока.

Чтобы автоматически распределять множество потоков между GPU, добавьте для каждого потока в конфигурационном файле опцию `deviceid=auto` в `transcoder`.

Deinterlace — устраняет чересстрочность. Читайте подробнее об этой опции здесь.

Для Nvidia эта опция представляет две опции в файле конфигурации (`deinterlace` и `deinterlace_rate`), которые используются вместе. Существуют следующие отношения между выбранным значением в поле **Deinterlace** и значениями этих опций в файле конфигурации:

|Deinterlace в UI|Опции в файле|Метод деинтерлейса Nvidia| |—-|—-|—| |off|`deinterlace=false, deinterl`
|on|`deinterlace=true, deinterlace_rate=frame`|adaptive| |on double rate|`deinterlace=true, d`
|adaptive|`deinterlace=adaptive, deinterlace_rate=frame`|adaptive| |adaptive double rate|`deinterlace=adaptive, deinterlace_rate=field`|adaptive|

Figure 10.18: transcoding options

Crop after decoding — большинства типов транскодера позволяет делать обрезку видео. Обрезка позволяет получить на выходе только часть площади изображения. Введите 4 числа, которые означают следующее: **Crop-X** и **Crop-Y** — координаты верхнего левого угла результирующего изображения, относительно исходного (т.е. если (0,0) — это левый верхний угол исходного изображения), **Crop-Width** — ширина изображения, **Crop-Height** — высота изображения.

GOP size — количество кадров в группе кадров GOP. Читайте подробнее об этой опции здесь.

Improve the transcoder performance by running it as part of Flussonic (use with caution). По умолчанию транскодер на сервере исполняется в отдельном от Flussonic процессе. Такое поведение более надежно обеспечивает бесперебойную работ Flussonic Media Server. Если вы включаете опцию **Improve the transcoder performance by running it as part of Flussonic** (`external=false` в файле), то транскодер будет выполняться в одном процессе с Flussonic Media Server. Это

Figure 10.19: transcoding options

ускоряет кодирование, особенно при кодировании аудио или при кодировании с использованием Nvidia. Однако ошибка транскодера может привести к остановке работы Flussonic Media Server.

При транскодировании нескольких потоков на Nvidia NVENC убедитесь, что опция **Improve the transcoder performance by running it as part of Flussonic** имеет одинаковое значение на всех потоках.

Параметры кодирования видео Добавить настройки видеодорожки в транскодер можно тремя способами:

- Нажмите кнопку **Add video track** и (опционально) выберите битрейт, чтобы появился диалог

настройки параметров видео.

- Чтобы получить на выходе видео с теми же характеристиками, что и на входе, отметьте **Copy from input**.
- Чтобы продублировать указанные вами настройки в новую видеодорожку, кликните **Duplicate**.

Figure 10.20: transcoding options

Кроме того, вы можете скопировать все настройки транскодера в другие потоки.

После того, как вы добавили видеодорожку, вы сможете редактировать ее настройки. Чтобы развернуть настройки кодирования дорожки, щелкните стрелку.

Все параметры находятся на одном экране:

Figure 10.21: transcoding options

- **Width** — ширина итогового видео в плеере на экране в пикселях.
- **Height** — высота итогового видео в плеере на экране в пикселях.
- **SAR (X:Y)** — соотношение ширины дисплейного представления к ширине пиксельного представления. Подробнее о SAR читайте здесь.
- **Resize** — стратегия изменения размеров (ресайза) видео до указанных размеров Height (и Width).
- **Background** — цвет фона — области в плеере, не заполненной видео после ресайза. Указывается только для стратегии 'fit'.
- **Bitrate** — видео-битрейт дорожки.

- **Codec** (H.264|H.265|MP2V|AV1) — видео-кодек. По умолчанию H.264.
- **Profile** (baseline|main|high) — профиль выходного видео в зависимости от кодека. Профиль позволяет предположить, может ли видео проигрываться на каком-либо устройстве.
- **Interlace** — используется для получения потока с чересстрочным кодированием из прогрессивного. Подробнее об опции `interlace` читайте здесь.
- **Preset** — влияет на качество и скорость доставки. Подробнее о пресетах читайте здесь.
- **B-frames** — значения 0|1|2|3|4 соответствуют последовательностям кадров: IP|IBP|IBBP|IBBBP|IBBBBP.
- **Open GOP** — разрешает транскодеру разбивать выходной поток на GOP с немного различающимся количеством кадров, но в районе заданного в опции **GOP size**. Эта настройка применима только к транскодированию на процессоре. Иногда это помогает уменьшить трафик.
- **Refs** — референсные кадры, использующиеся при межкадровой компрессии для ссылок на последующие кадры. Лучшее качество видео будет при большем количестве референсных кадров.
- **Level** — может использоваться для совместимости с устаревшими устройствами проигрывания.
- **Logo**
- **Extended** — если требуемых опций нет на экране, добавьте их вручную в разделе **Extended**:

Полный список параметров с актуальными описаниями см. в API reference.

Сохранение настроек Чтобы сохранить новые значения, нажмите **Save**.

Чтобы удалить все настройки и отключить транскодер для этого потока, нажмите **Disable transcoder**.

Копирование настроек транскодера в другие потоки Чтобы скопировать настройки в другие потоки:

- Перейдите на вкладку **Transcoder** потока, в котором вы уже настроили параметры транскодера.
- Нажмите **Copy settings**
- Перейдите на вкладку **Transcoder** потока, где вы хотите применить те же настройки, и нажмите **Enable and paste settings**. Если в потоке уже настроен транскодер, кнопка будет **Paste settings**.

Настройка транскодирования через файл

Опции транскодирования можно указать в конфигурационном файле Flussonic `/etc/flussonic/flussoni`

Чтобы включить и настроить транскодер через конфигурационный файл, используйте директиву `transcoder` с набором опций транскодирования.

Указывать опции транскодирования необходимо в следующем порядке:

- глобальные опции,
- опции видео (обязательные и необязательные) для каждого трека,
- опции аудио (обязательные и необязательные).

Пример настройки транскодера для потока `example`:

```
stream example {
  input fake://fake;
  transcoder vb=2048k size=1280x720 preset=slow ab=128k;
}
```

Пример настройки параметров мультибитрейтного потока:

```
vb=2048k preset=veryfast vb=700k size=720x576 preset=veryfast vb=300k size
   =320x240 preset=veryfast ab=128k
```

Опции транскодера для анаморфного видео

Транскодер поддерживает анаморфные видеопотоки путем учета соотношения сторон пикселя. Для настройки используйте опции `size` и `sar`:

Читайте подробное описание опций ‘size‘ и ‘sar‘ в списке опций транскодера на этой странице.

Кроме `size`, изменилось поведение следующих старых параметров: `aspect`, `force_original_aspect_r` и `crop`:

- Параметр `aspect` заменен на `sar`. Почти все типы транскодеров во Flussonic будут интерпретировать его как SAR (не DAR), за исключением Nvidia NVENC.
- Параметр `force_original_aspect_ratio` не требуется в большинстве случаев, и, если он требуется, проставляется автоматически.
- Параметр `crop` теперь поддерживается почти всеми типами транскодеров (не спутайте его со стратегией изменения размера ”crop”).

Настройки транскодера, которые вы настроили в более ранних веррсиях, останутся такими же и будут обрабатываться прежним методом. Транскодер станет обрабатывать параметры по-новому, только если вы указываете новые параметры — SAR или стратегию изменения размера (или оба) — и при этом не указываете устаревшие параметры (`aspect`, `force_original_aspect_ratio`).

Аппаратное ускорение

С помощью аппаратного транскодера можно серьезно увеличить количество транскодируемых потоков на одном сервере.

Flussonic Media Server поддерживает технологии Nvidia NVENC и Intel Quick Sync. Один видеопоток можно транскодировать с использованием только одного вида транскодера. Подробнее про установку и настройку аппаратного ускорения см. Аппаратное транскодирование на Nvidia NVENC и Intel Quick Sync Video.

Опции транскодера

Схему API для всех опций транскодера можно найти в справочнике API.

Глобальные опции:

hw `hw` — включает аппаратное кодирование. Указывается в настройках потока.

deinterlace `deinterlace` — конвертирует чересстрочное видео в прогрессивное.

В чересстрочном видео нечетные и четные строки кадра демонстрируются как отдельные поля. Сначала на экране отображаются нечетные строки, затем – четные. Два таких поля вместе составляют один видеокадр. Чересстрочное видео хорошо подходит для вещания, т.к. изображения могут демонстрироваться на экране с низкой пропускной способностью. Но у чересстрочного видео также есть и недостаток: при быстром движении видео может быть нечетким, т.к. в один момент времени захватывается лишь часть изображения, и по краям видеокадра могут быть заметны искажения.

В прогрессивном же видео нечетные и четные строки отображаются одновременно, то есть видеокадр отображается на экране целиком.

Деинтерлейсинг необходим для комфортного просмотра ТВ на ПК/мобильных устройствах. Указывается один раз и действует сразу на все видеопотоки.

В UI этой опции соответствует поле **Deinterlace**.

deinterlace_rate `deinterlace_rate` — для Nvidia NVENC можно удалять второе поле, получившееся после устранения чересстрочности, предотвращая тем самым повышенный битрейт.

- `deinterlace_rate=frame` — из последовательности полукадров `1a 1b 2a 2b 3a 3b`, получаются кадры `1a1b 2a2b 3a3b`. При этом fps остается прежним.
- `deinterlace_rate=field` — из полукадров `1a 1b 2a 2b 3a 3b` формируются `1a1b 1b2a 2a2b` fps увеличивается после транскодирования в два раза.

В случае использования Nvidia NVENC обе опции (`deinterlace` и `deinterlace_rate`) добавляются в файл конфигурации при выборе вами какого-либо значения в поле **Deinterlace**. Существуют следующее соответствие между выбранным значением в поле **Deinterlace** и значениями опций в файле конфигурации:

Deinterlace в UI	Опции в файле	Метод деинтерлейса N
off	`deinterlace=false, deinterlace_rate=frame`	weave
on	`deinterlace=true, deinterlace_rate=frame`	adaptive
on double rate	`deinterlace=true, deinterlace_rate=field`	adaptive
adaptive	`deinterlace=adaptive, deinterlace_rate=frame`	adaptive
adaptive double rate	`deinterlace=adaptive, deinterlace_rate=field`	adaptive

crop `crop` — позволяет обрезать видео.

Использование: `crop=x:y:width:height`, где:

- `x:y` — координаты левого верхнего угла выходного видео в пределах размеров входного видео,
- `width` — ширина выходного видео
- `height` — высота выходного видео.

gop `gop=150` — устанавливает количество кадров в одном GOP. Flussonic транскодирует поток, создавая каждый GOP точно такого размера, как указано в этой настройке.

Чтобы разрешить делать GOP не обязательно строго одного размера, используйте в дополнение к этой опции опцию `disable_cgop`.

fps `fps` — задает частоту кадров. Указывается отдельно для каждого видеопотока.

Опции видео

vb `vb` (video bitrate) — параметр, задающий битрейт видео дорожки. Задаётся в виде числового значения (1000k, 1500k, 2000k и т.д.). **Значение должно обязательно заканчиваться на _k_.** Каждое указание опции `vb` создает новую видео дорожку в выходном потоке.

Опция `vb=copy` сохраняет параметры оригинального потока, то есть просто копируется в исходящий поток.

При формировании мультибитрейтного потока мы не рекомендуем использовать одновременно `vb=copy` и численное значение `vb=NUMk` в параметрах транскодера. Иначе у ваших зрителей могут возникнуть проблемы с проигрыванием потока, например, зависание потока на конечном устройстве.

preset `preset` — предустановленный набор значений, обозначающих скорость кодирования, от которой зависит качество сжатия. Чем лучше качество сжатия, тем дольше по времени кодируется файл, и наоборот.

Это означает, что лучшего качества при кодировании можно достичь используя более медленный preset (`slow`), но кодирование займет больше времени. Используйте «медленные» пресеты, если для вас важнее качество, а не скорость.

Список поддерживаемых пресетов:

- `veryfast`
- `medium`
- `slow`

Пресет по умолчанию — `medium`.

size `size` — задает размеры видео на дисплее, где оно будет отображаться. Используется вместе со стратегией ресайза (crop, fit, scale) и цветом фона той части окна плеера, которую не заполнит видео.

Параметр `size` теперь означает размер окна воспроизведения на экране, которое Flussonic передает плееру, а не размер видео в пикселях. Ранее `size` интерпретировался как пиксельный размер, а размер окна воспроизведения зависел от SAR потока или от значения устаревшего параметра `aspect`.

logo `logo` — позволяет наложить логотип. Логотип добавляется до изменения размера изображения видео нашим транскодером. Это значит, что при значительном изменении размера выходного видео логотип может заметно исказиться.

Подробнее о наложении логотипа

alogo `alogo` — позволяет наложить логотип. Логотип добавляется после изменения размера изображения видео нашим транскодером. Это предотвращает растягивание логотипа, которое может произойти при добавлении логотипа при помощи опции `logo`. Для каждого выходного разрешения видео вам нужно подготовить и указать отдельный файл логотипа.

Подробнее о наложении логотипа

vcodec `vcodec` — позволяет задать видео кодек. По умолчанию используется H.264. Flussonic Media Server позволяет кодировать в H.265 ('hevc'), `mp2v` или AV1. Указывается отдельно для каждого видеопотока.

Кодек `mp2v` недоступен при использовании аппаратного кодирования.

refs `refs` — количество референсных кадров. Указывается отдельно для каждого видеопотока.

bframes `bframes` — позволяет отключить b-frames. Это может понадобиться, например, при вещании в RTSP. Указывается один раз и действует сразу на все видеопотоки.

sar `sar` — изменяет соотношение сторон видео. Применяется, чтобы из анаморфного видео получить не-анаморфное. Используется вместо устаревшего `aspect`, но с отличиями.

SAR в терминах Flussonic — это соотношение ширины дисплейного представления к ширине пиксельного представления. Ширина дисплейного представления — это количество пикселей на матрице отображающего дисплея, это то, что передается плееру для проигрывания. А ширина пиксельного (внутреннего) представления — это количество пикселей в оригинальной YUV.

Пиксельную ширину входного видео надо умножить на SAR (X:Y), чтобы получить дисплейную ширину. Этот параметр можно использовать (иногда вместе со стратегией изменения размера) если нужно получить видео для отображения на нестандартном дисплее. Фактически меняет параметры анаморфности.

В пользовательском интерфейсе `sar` представлен в расширенных опциях видео.

Старый параметр `aspect` обрабатывается как SAR. Для транскодера на Nvidia NVENC `aspect` интерпретируется как DAR (соотношение размеров окна плеера) и обрабатывается как в более ранних версиях Flussonic.

> Изменение соотношения сторон не поддерживается для аппаратного кодирования с использованием Intel QuickSync (`hw=qsv`).

Flussonic исходя из `sar` вычисляет разрешение выходного видео. Видео с внутренней пиксельной шириной 720 и sar=16:11 будет на выходе из Flussonic иметь дисплейную ширину 1048. Картинка такой ширины, в пикселях дисплея, будет при воспроизведении в плеерах.

force_original_aspect_ratio (не использовать) `force_original_aspect_ratio=true` — сохраняет соотношение сторон видео путем добавления черных полос. Опция полезна если вы хотите сохранить разрешение на выходе при переключении между разными источниками.

Пример:

```
vb=2048k size=1280x720 force_original_aspect_ratio=true
```

disable_cgop `disable_cgop=1` — разрешает транскодеру разбивать выходной поток на GOP с немного различающимся количеством кадров, но в районе заданного в опции `gop`. Эта настройка применима только к транскодированию на процессоре (а не на аппартном транскодере). Иногда это помогает немного уменьшить трафик.

interlace `interlace` — используется для получения потока с чересстрочным кодированием из прогрессивного.

Возможные значения опции: `interlace=tff|bff|tff_separated|bff_separated|mbaff|true`

Если опция отсутствует в конфигурации, то на выходе останется progressive. Если просто включить опцию без указания метода получения чересстрочного видео (`interlace=true`), то будет использован метод по умолчанию (свой для каждого вида транскодера). Можно указать и другой метод.

- `tff` — interlaced, top field first, interleaved field store. Используется с hw=qsv, nvenc.
- `bff` — interlaced, bottom field first, interleaved field store. Используется с hw=qsv, nvenc.
- `tff_separated` — interlaced, top field first, separated fields. Используется с hw=qsv.
- `bff_separated` — interlaced, top field first, separated fields. Используется с hw=qsv.
- `mbaff` — interlaced libx264 MBAFF method. Используется только с hw=cpu.
- `true` — включить чересстрочную развертку методом по умолчанию для используемого энкодера (метод `mbaff` используется по умолчанию для hw=cpu, метод `tff` — для hw=qsv, nvenc).

rc_method `rc_method` используется для создания выходного видео с постоянным битрейтом, подходящего для трансляции в телевизионные сети.

Опция принимает значения:

- `rc_method=cbr` — энкодер создаст поток, совместимый с DVB-C.
- `rc_method=vbr` — не кодировать поток для совместимости с DVB-C.

На данный момент использование этой опции потребляет много ресурсов (одно ядро ЦП для одного потока MPTS с CBR).

Опции аудио

ab `ab` — задает битрейт аудио. Указывается только один раз (даже если у вас мультибитрейтное видео). Значение должно обязательно заканчиваться на _k_.

acodec `acodec` — задает аудио кодек. Доступные значения: `aac`, `mp2a`, `opus`, `pcma`, `ac3`. По умолчанию все аудиопотоки пережимаются в AAC.

ar `ar` — задает sample rate, частоту дискретизации). Пример: `ar=44100`.

ac `ac` — задает количество аудио-каналов.

avol `avol` — громкость звука на выходе. Это может быть значение в Дб (с "+" или "-"), которое будет добавлено к громкости на входе, либо коэффициент, на который нужно умножить громкость на входе. Подробнее см. Как изменить уровень громкости звука?.

split_channels `split_channels` — если значение этой настройки равно `true`, каждый аудиотрек с несколькими каналами будет разбиваться на несколько моно-треков.

Прожиг текста, времени и субтитров

Настройки Настройки прожига текста, времени и субтитров можно указать через API или в файле конфигурации одним из следующих способов:

- Как глобальный параметр транскодера. Этот вариант соответствует параметру API 7Ctranscoder%7Cgloba get/response%7Ctranscoder%7Cglobal%7Cburntranscoder.global.burn (в конфигурационном файле — задается перед всеми видео дорожками vb). Текст, время или субтитры, прожигаемые таким образом, будут добавлены во все дорожки, которые выдает на выход транскодер.
- Как настройку транскодирования конкретного потока. Этот вариант соответствует параметру API 7Ctranscoder%7Cvideo%7Cburntag/stream/operation/stream-get/response%7Ctranscoder%7Cvideo%7C (в конфигурационном файле — задается после соответствующего параметра vb). Текст, время или субтитры, прожигаемые таким образом, будут добавлены только в соответствующую дорожку. Настройка потока имеет приоритет над глобальной настройкой.

Общий синтаксис для опции burn:

`burn=<filter>@<text:pos:x:y>@font:<ttf:size:color:alpha>@box:<border:color:alpha`

Здесь:

- filter — time|sub|text
- text — дорожка с субтитрами (например, t1) — для `sub`, какой-либо текст (для `text`), %T или %F или их комбинация (для `time`).

- text:pos:x:y — pos — расположение (буквенное обозначение расположения — см. ниже), x:y — смещение вправо или влево (x) и вверх или вниз (y) к центру. Смещение не может быть отрицательным числом.
- font:
- box:

Краткий вариант опции burn:

- `burn=time` выведет время в дефолтном формате
- `burn=sub` выведет WebVTT субтитры из дорожки t1
- `burn=text` выведет пустую строку (позднее можно установить текст при помощи API)

Правила:

- Первая группа (`<filter>`) является обязательной, остальные группы не обязательные (text, font, box).
- Первый параметр в каждой группе является обязательным (text, font, box).
- Порядок параметров должен соблюдаться.
- Отсутствующие параметры будут заменены на значения по умолчанию: size — 16, color — `black` для `box`, `white` для `font`, border — 6, alpha — 0.8, ttf — FiraCode-Regular.ttf.
- Можно одновременно использовать несколько видов опции burn (например, `burn=text` и `burn=time`), но нельзя использовать один и тот же тип дважды для одной дорожки.

Ниже более подробное описание с примерами.

burn=time `burn=time` — добавляет время. При желании вы можете указать смещение времени относительно времени сервера Flussonic, по умолчанию смещение = 0.

Дополнительно можно настроить отображение времени — шрифт (font) и расположение на экране (box) — см. Настройки отображения.

Примеры настроек:

- `burn=time@offset+3` — выведет время в дефолтном формате `YYYY-MM-DD HH:MM:SS` в зоне +3 часа от времени сервера Flussonic.
- `burn=time@%Toffset-3:tr@box` — выведет время в формате `HH:MM:SS` в зоне -3 часа в темном боксе в правом верхнемм углу (tr = top right)

- `burn="time@%F -- %Toffset-2:c@font:FiraCode-Regular.ttf:26:green:0.8@box:yel` — выведет время в формате YYYY-MM-DD -- HH:MM:SS в зоне -2 часа в центре кадра с дополнительными настройками шрифта и бокса.
- `burn=time@%F:cb:0:200@font:default@box` — время в формате YYYY-MM-DD по центру снизу со смещением на 200 вверх.

burn=sub `burn=sub` — записывает субтитры в поток (dvb_teletext, dvb_subtitles или closed captions).

Вначале необходимо извлечь субтитры в текстовый вид, чтобы передать их на вход транскодеру. Например, в случае closed captions для этого используется опция `cc.extract`.

Дополнительно можно настроить отображение субтитров — шрифт (font) и расположение на экране (box) — см. Настройки отображения.

Пример опции:

```
burn="sub@t1:cb:10:10@font:Arial-Regular.ttf:30:white:1.0"
```

Здесь:

- `sub` указывает, что следует взять субтитры из входного потока (dvb_teletext, dvb_subtitles или closed captions) и "прожечь" субтитры в выходной поток.
- `t1` — номер дорожки WebVTT субтитров.
- `cb` (central bottom = внизу по центру) — расположение субтитров.
- `10:10` — сдвиг по горизонтали и по вертикали к центру относительно указанного буквами расположения.
- `font:FONT_NAME.ttf` — шрифт. См. настройки отображения шрифтов ниже после примера.
- `30` — размер шрифта.
- `white` — цвет шрифта.
- `1.0` — прозрачность текста (используйте значения от 0.1 до 1.0, 0.0 — полностью прозрачно, 1.0 — полностью непрозрачно).

Пример прожига для потока, который содержит closed captions:

```
stream example {
  input udp://239.0.0.1:1234 cc.extract;
  transcoder vb=3000k burn="sub@t1:cb:0:80@font:default:35:white:1.0"
    vcodec=h264 open_gop=false preset=veryfast size=1920x1080:scale:#000000
    vb=1800k burn="sub@t1:cb:0:80@font:default:25:white:1.0" vcodec=h264
    open_gop=false preset=veryfast size=-1x720:scale:#000000 ab=128k;
}
```

В примере мы извлекаем closed captions с помощью опции `cc.extract`. Вам могут потребоваться другие опции, а не эта.

> В зависимости от того, какой вид субтитров содержится в вашем входном потоке, используйте соответствующие опции для извлечения субтитров в текстовый формат.

burn=text `burn=text` — записывает указанную текстовую строку.

Дополнительно настроить отображение текста — шрифт (font) и расположение на экране (box) — см. Настройки отображения.

Пример опции для добавления текста в конкретную дорожку:

```
transcoder vb=3500k burn=text@Hello:tr@box:green ab=64k;
```

Шрифт

- Flussonic поддерживает шрифты .ttf
- Flussonic ищет указанный в параметрах файл шрифта в подкаталоге `font` каталога `/etc/flussonic/`. Т.е. вы можете поместить файл сюда: `/etc/flussonic/font/SomeFont.ttf`
- Если файл шрифта, указанный в параметрах, отсутствует в `/etc/flussonic/font/`, автоматически будет применен дефолтный шрифт `FiraCode-Regular.ttf`, который входит состав Flussonic.
- Если шрифт у вас лежит в другой директории, можно указать в качестве параметра полный путь к файлу шрифта. Например, укажем путь к одному из системных шрифтов:

```
font:/usr/share/fonts/truetype/freefont/FONT_NAME.ttf:50:white:1.0
```

- Можно явно указать шрифт по умолчанию: `font:default:30:white:1.0`. Будет использован `FiraCode-Regular.ttf`, однако если вы скопируете в директорию со шрифтами `/etc/flussonic/` файл шрифта с именем `default`, то будет использоваться он.

Примеры:

`font:default:50` — дефолтный шрифт с размером 50

`font:default:24@box` — дефолтный шрифт с размером 24 в боксе с размерами по умолчанию

`font:default:26:blue` — указан размер и цвет

`font:default:26:blue:0.9` — указаны размер, цвет, прозрачность

Расположение Дополнительно можно указать место на экране, где будут отображаться данные.

- tl — top left — левый верхний угол
- tr — top right — правый верхний угол
- bl — bottom left — левый нижний угол
- br — bottom rigth — правый нижний угол
- c — center — посередине
- ct — center top — вверху по центру
- cb — center bottom — внизу по центру

Вместе с этими обозначениями можно указать смещение по горизонтали и вертикали от указанного расположения:

- `cb:10:200` — текст будет расположен по центру внизу кадра со смещением x=10 (вправо) и y=200 (вверх)
- Смещения по умолчанию равны 10. Смещения могут быть положительными числами или 0.

Пример:

`burn=time@%F:cb:0:200@font:default@box` — время в формате YYYY-MM-DD по центру снизу со смещением на 200 вверх.

Для обработки и отображения шрифтов Flussonic использует библиотеку libfreetype, которая включена в набор библиотек, предоставляемый пакетом flussonic-transcoder-base. Для рендеринга текста в CPU и Nvenc транскодерах используется фильтр ffmpeg drawtext

10.4.2 Транскодирование

Со спутника видео передается либо в кодеке MPEG-2, либо в H.264 (он же AVC или MPEG-4 part10). Как правило, для простоты MPEG-4 part 10 сокращают до MPEG-4, но тут важно не спутать с MPEG-4 part 2, который совершенно никак не совместим и не похож на H.264 и использовался в старых IP камерах.

Аудио передается в MPEG audio layer 2 (сокращенное mp2), либо в ac3 (a/52).

Причём важно понимать, что сегодня H264, как правило, сжимается с intra-refresh, т.е. в видео потоке нет опорных кадров (IDR или keyframe). Такой метод сжатия позволяет сгладить скачки битрейта.

В результате ни один из передаваемых со спутника вариантов аудио или видео не проигрывается на айфоне. В браузере проигрывается только H264.

При передаче через интернет, как правило, можно смело сжимать видео из mpeg2 в h264 с трехкратным снижением трафика.

При передаче HD каналов через интернет сегодня приходится сжимать поток в несколько разных качеств: от HD с максимальным качеством до стандартного SD для компенсации перегруженных каналов.

В итоге видео со спутника для предоставления качественного OTT сервиса надо транскодировать в другие кодеки и качества.

Важно не путать транскодирование с перепаковкой. Транскодирование — крайне ресурсоёмкая операция, включающая в себя:

- распаковку потока до кодированного видео/аудио
- декодирование до сырого видео/аудио
- изменение размеров и прочих параметров
- кодирование обратно
- упаковка в транспорт для потока

Упаковка и распаковка относительно легкие операции, стриминговый сервер может обрабатывать до 1000 каналов на одном компьютере. Транскодировать на одном компьютере можно от 1 до 30 каналов в зависимости от размера и мощности компьютера.

Для транскодирования можно использовать специализированные выделенные устройства, центральный процессор или видеоплату: внешнюю или встроенную в процессор.

Специализированные устройства мы рассматривать не будем, потому что в своей массе это либо компьютер с какой-то программой, либо крайне дорогостоящее и очень специализированное оборудование, или же либо попросту необоснованно дорогое устройство, реализуемое исключительно за счёт маркетинговых усилий компании производителя и не позволяющее достигнуть значимых результатов.

H.264

Для обработки видео на CPU существует несколько разных программ, но по большому счёту на сегодняшний день существует лишь две библиотеки, которые имеет смысл использовать для сжатия в кодек H.264 на CPU: это бесплатная libx264 и платная MainConcept. Всё остальное либо хуже, либо сильно хуже, причём как по выходному результату, так и по использованию ресурсов.

Практика работы с MainConcept в этой статье рассматриваться не будет, будет упомянута только libx264

Кодек H.264 является стандартом де-факто на сегодняшний день для видео, потому что он поддерживается во всех современных устройствах, за исключением разве что некоторых устройств от Google.

Альтернатив ему практически нет. Сегодня появился и развивается H.265, у него уже есть большая поддержка, но пока что работа с ним — это инвестиции в будущее.

Кодеки от Google: VP8 и VP9 являются больше желанием гугла перетянуть одеяло на себя, нежели чем-то реально полезным. Результирующее качество хуже, поддержки аппаратного декодирования нет, а следовательно растет цена устройства.

При кодировании видео надо понимать, что приходится балансировать между такими параметрами:

- задержка внутри энкодера в кадрах,
- использование CPU (сколько миллисекунд требуется на сжатие одного кадра),
- выходное качество картинки (насколько пиксельная и какие цвета),
- выходной битрейт.

Для всех видов эфира абсолютно критичным является использование CPU. Если настройки энкодера требуют полной загрузки CPU или больше, то видео не будет успевать кодироваться в реальном времени и, следовательно, потоковость видео пропадет.

Для VOD такого жесткого ограничения нет и фильм длиной в час вполне можно кодировать три часа, если хочется понизить битрейт. При этом для эфирного видео обычно все-таки стараются использовать не всю мощность процессора, что бы обрабатывать на одном компьютере не 4 канала, а 10.

Что касается задержки внутри энкодера, то она критична для видеоконференций, но совершенно некритична для IPTV. Даже 5 секунд задержки при вещании телевидения не меняют качество сервиса.

У битрейта и качества связь достаточно четкая: чем больше информации о картинке мы передаем, тем лучше она будет отображаться. Повысить качество картинки, снизив битрейт, как правило, можно за счёт выбора более результативных инструментов компрессии, которые требуют большей задержки и большего количества тактов.

Понимание этой сложной взаимосвязи нужно для того, что бы лучше воспринимать заверения о том, что «наш энкодер самый лучший энкодер в мире». Сравнивать приходится минимум по 4-м параметрам, но в итоге всё сводится к тому: сколько денег стоит разово и в месяц транскодирование одного канала с желаемым качеством и выходным битрейтом.

Flussonic Media Server для транскодирования

Отдельным пакетом к Flussonic Media Server идет транскодер.

Flussonic Media Server может декодировать видео из UDP/HTTP MPEG-TS, RTMP источников и кодировать его в несколько качеств и размеров.

Эта возможность становится нужна, когда возникает необходимость показывать видео не только на приставках, но и на планшетах: там выбор доступных кодеков существенно меньше, чем на приставке.

Важно отметить, что для того, что бы видео игралось на айфоне, надо даже H264 со спутника транскодировать, потому что как правило на спутнике для плавного битрейта используется intra-refresh режим кодирования, создающий видео, которое не играется на айфоне.

Flussonic Media Server удобнее чем VLC или другие варианты для организации транскодирования, потому что управляется одним конфигурационным файлом и автоматически следит за состоянием транскодирования. VLC же требует написания большого количества мониторинговых скриптов для отслеживания состояния транскодирования.

Следующая важная возможность Flussonic Media Server для транскодирования — автоматическая перебалансировка потоков при падении одного из серверов. Если один из 20 транскодеров ночью сломается, то остальные транскодеры можно настроить на автоматический захват потоков для транскодирования, причём стример сам заберет потоки с резервных транскодеров.

10.4.3 Flussonic Coder

Flussonic Coder — программно-аппаратное решение для транскодирования видео и аудио, которое имеет преимущества перед другими способами транскодирования с использованием Flussonic Media Server:

- позволяет крупным компаниям комплексно и предсказуемо закрывать потребности клиентов,
- позволяет унифицировать процесс техподдержки,
- помогает интеграторам защищать проекты,
- сохраняет доступ к абонентскому устройству.

Flussonic Coder представляет собой составную часть кластера Flussonic, предназначенного для обработки, передачи и записи видео. Он поддерживает множество форматов, кодеков и протоколов для видеопотоков в любой точке кластера.

Flussonic Coder — это сервер с нашей специальной операционной системой Linux, несколькими модулями NVIDIA Jetson и установленным программным обеспечением для транскодирования. Он поставляется вместе с прошивкой и не требует установки дополнительных драйверов. Один модуль NVIDIA Jetson может транскодировать 6 потоков Full HD в 3 профиля качества или 12 потоков SD в 3 профиля качества.

Полученные видеопотоки существуют во Flussonic в виде последовательности элементарных кадров. На входе видео демультиплексируется на отдельные кадры, а на выходе — мультиплексируется и снова упаковывается для передачи по одному из современных протоколов видеостриминга.

В этой статье:

- Настройка Flussonic Coder
- Настройка потока для использования Flussonic Coder

Настройка Flussonic Coder

На Flussonic Coder установлен Flussonic Media Server, так что настраивать Flussonic Coder можно через веб-интерфейс Flussonic Media Server. Чтобы посмотреть настройки Flussonic Coder, перейдите на страницу **Chassis**. На этой странице представлены четыре раздела, которые описаны ниже.

System information Здесь вы можете:

- посмотреть версию Flussonic Coder,
- посмотреть версию прошивки и проверить наличие новой версии,

- обновить прошивку до конкретной версии или до последней версии,
- перезапустить Flussonic Coder, нажав **Restart Chassis**.

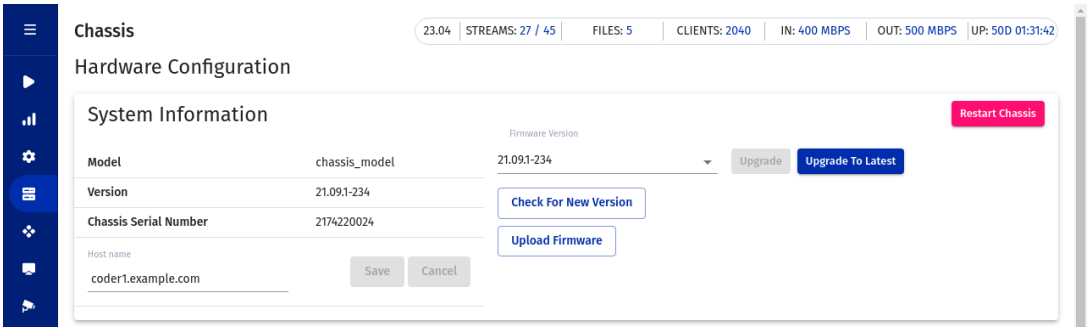


Figure 10.22: Flussonic Coder System

Network configuration Здесь вы можете изменить шлюз по умолчанию. Для этого выберите необходимый интерфейс шлюза из выпадающего списка в поле *Default gateway interface* и нажмите **Save**, чтобы сохранить настройки.

Шлюз по умолчанию используется для проверки лицензии и обновлений, а также для отправки ответов на запросы (API, HLS и т.д.), поступающие на сетевые интерфейсы Flussonic Coder. Изменять шлюз по умолчанию для сетевой интерфейса необходимо с осторожностью и силами опытных сетевых инженеров, чтобы не потерять управление Flussonic Coder.

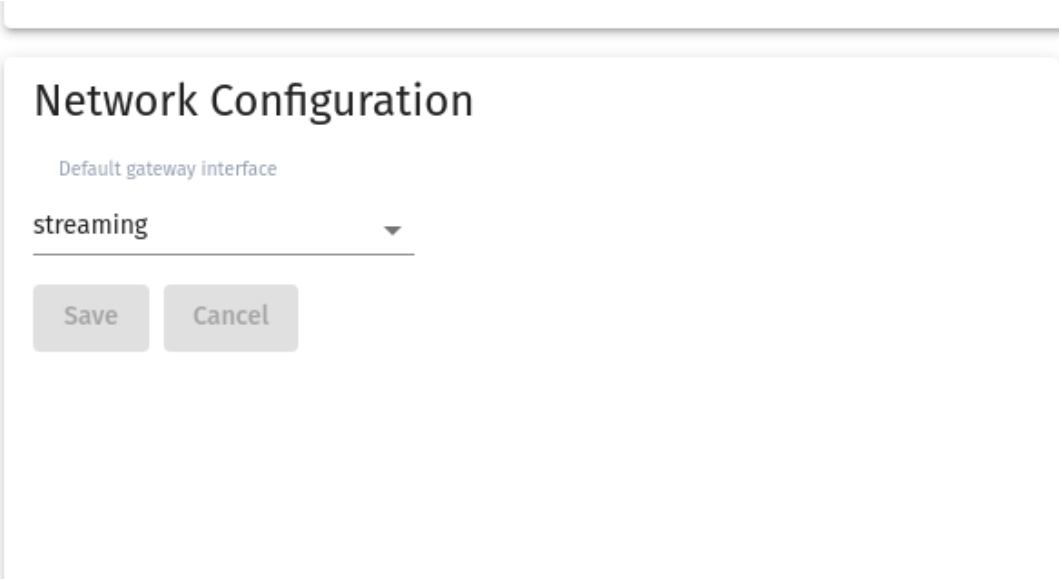


Figure 10.23: Flussonic Coder Network

Time configuration Здесь вы можете проверить текущее системное время на устройстве и увидеть статус синхронизации с NTP-сервером. Также в этом разделе можно добавить URL-адреса NTP-серверов.

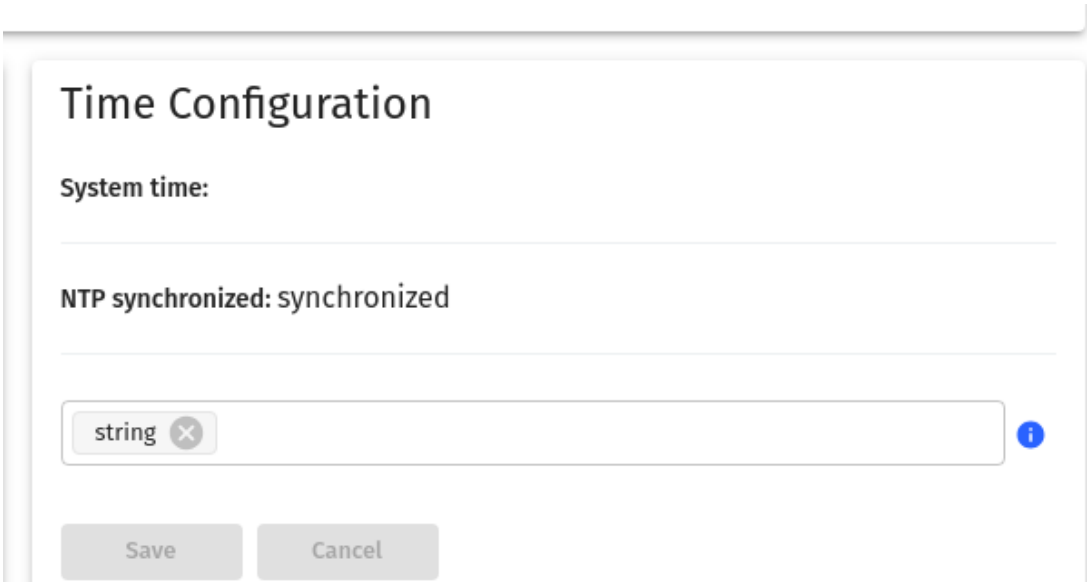


Figure 10.24: Flussonic Coder Time

Interfaces Этот раздел содержит информацию о DNS и список сетевых интерфейсов, которые использует Flussonic Coder. Здесь вы можете:

- указать IP-адрес сервера DNS,
- выбрать тип интерфейса (static или DHCP),
- указать IP-адрес интерфейса, маску сети или шлюз,
- включить или отключить интерфейс.

Hardware Modules Monitor В этом разделе показана информация о модулях NVIDIA Jetson, которые использует Flussonic Coder для транскодирования.

Здесь вы можете:

- Отслеживать такие данные модуля, как:

- статус работы (*Status*), - число транскодируемых каналов (*Channels*), - температура (*Temperature*), - потребляемая мощность (*Power*), - серийный номер (*Serial Number*).

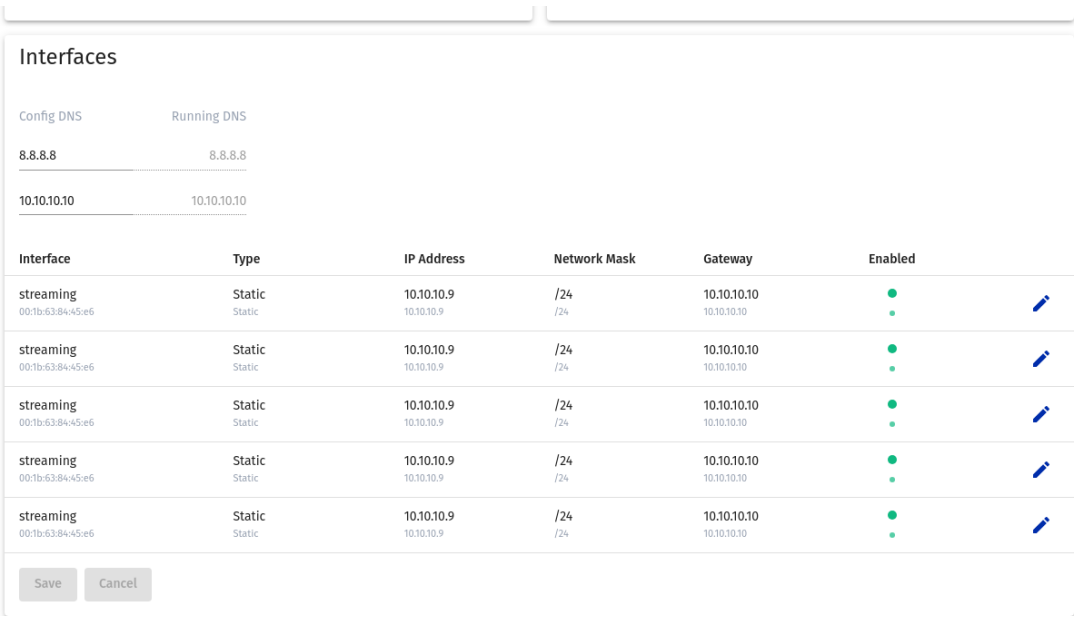


Figure 10.25: Flussonic Coder Network

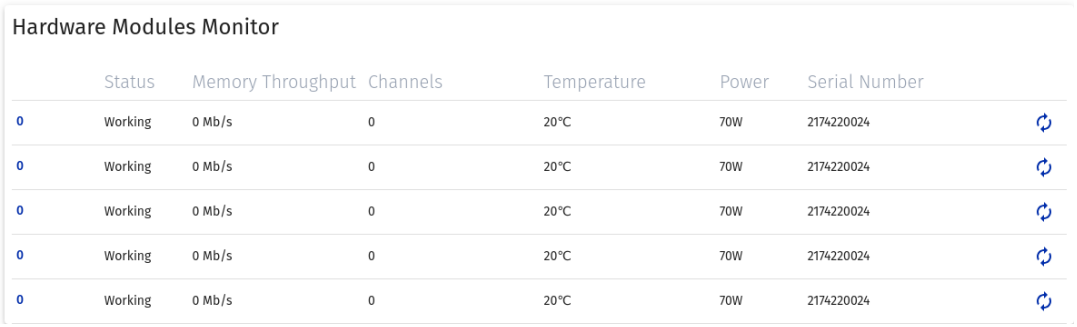


Figure 10.26: Flussonic Coder Monitor

- Перезагрузить модуль при необходимости.

Обратите внимание, что расположение модулей NV02 в слотах шасси строго определяется архитектурой устройства. Модули должны быть вставлены в шасси, начиная с левого слота: например, если у вас есть один модуль, то он должен быть вставлен в первый слот слева, два модуля в первый и второй слот и т. д. Нельзя менять модули местами, перемещать или удалять их.

Если вы заметили проблемы в порядке нумерации или не все модули доступны, проверьте, надежно ли они закреплены в слотах и в правильном ли порядке. Это может быть связано с отхождением контактов при транспортировке. Обязательно сообщите о проблеме в службу поддержки support@flussonic.com.

Настройка потока для использования Flussonic Coder

Чтобы настроить поток на использование Flussonic Coder для транскодирования, используйте опцию `hw=coder` в директиве `transcoder` в конфигурации потока. Читайте больше о настройках транскодера на странице Транскодер.

Flussonic Coder поддерживает метод CUDA yadif для деинтерлейсинга видео, что позволяет лучше обрабатывать динамические сцены. Чтобы использовать этот метод, добавьте опцию `deinterlace=yadif` в конфигурационный файл:

```
transcoder deviceid=0 hw=coder vb=10000k size=1920x1080 deinterlace=yadif
    ab=192k
```

10.4.4 Транскодирование отдельных аудиодорожек

В некоторых ситуациях бывает необходимо транскодировать входные аудиодорожки по-отдельности с разными параметрами транскодирования. Например, когда видео принимается со спутника, одна аудиодорожка может быть закодирована с помощью кодека MP2A, а другая – с помощью AC3. Аудиодорожка в AC3 имеет достаточно хорошее качество и не нуждается в транскодировании, тогда как аудиодорожку в MP2A нужно транскодировать для проигрывания в браузерах. Кроме того, иногда может быть необходимо из одной входной аудиодорожки создать несколько выходных с разными параметрами (например, с разным битрейтом).

Для транскодирования отдельных аудиодорожек используйте опцию `atrack` в конфигурации транскодера. Эта опция позволяет указать порядковый номер входной аудиодорожки как целое число или строку в формате `a<N>`. Например, `atrack=1` или `atrack=a1` означает первую входную аудиодорожку.

Все опции аудио, указанные в конфигурации перед первым вхождением опции `atrack`, по умолчанию применяются ко всем аудиодорожкам. Опции, указанные после `atrack`, применяются к конкретной аудиодорожке. Если после `atrack` не указано никаких опций, то у выходной аудиодорожки будут параметры, указанные для всех аудиодорожек.

Пример для транскодирования трех входных аудиодорожек с разными параметрами:

```
stream sample {
  input fake://fake;
  transcoder vb=1000k ab=copy acodec=aac atrack=1 ab=copy atrack=2 ab=64k
    atrack=3;
}
```

Первая и третья входные аудиодорожки будут транскодированы с оригинальным битрейтом, а вторая – с битрейтом 64k.

Пример для создания двух аудиодорожек из одной входной аудиодорожки:

```
stream fake {
  input fake://fake;
  transcoder vb=copy ab=64k acodec=ac3 atrack=1 ab=64k acodec=opus atrack
    =1;
}
```

В этом примере транскодер создаст две аудиодорожки из первой входной аудиодорожки. Настройки первой выходной дорожки: `ab=64k`, `acodec=opus`. Настройки второй выходной дорожки: `ab=64k` `acodec=ac3` (так как эти опции применяются ко всем аудиодорожкам).

10.5 DVR

10.5.1 Запись видеопотоков (Digital Video Recording, DVR)

Flussonic Media Server позволяет записывать видеопотоки и проигрывать их. Эту функциональность мы называем DVR (digital video recording).

DVR записывает потоки после транскодера и до DRM, т.е. на диск записывается уже обработанное, но ещё не зашифрованное видео. Для записи оригинального, нетранскодированного видео, нужно использовать дублирование потоков (протокол `copy`) и записывать оригинальный поток, а транскодировать уже во втором.

Подсистема архива представляет из себя очень сильно развитую, надежную и высокоэффективную технологию, в которой заложены следующие особенности:

- Единый формат данных для разных протоколов проигрывания, позволяющий один раз скачать и раздавать в разных протоколах
- Многоуровневое индексирование, позволяющее эффективно оперировать архивами глубиной в год с самого запуска, не блокируясь на старте
- Параллелизированный доступ к дискам, позволяющий защитить весь сервер от перегрузки одного устройства
- Эффективное управление RAM кешем данных
- Интегрированное управление самими данными и их индексами, снимающее необходимость администрирования дополнительной БД
- Прецизионное сохранение таймстемпов кадров, необходимое для интеграции с внешней аналитикой

Более подробно про особенности архива:

Запись архива

- Расширяемая запись на локальные диски и в облачные хранилища по протоколу S3. NFS тоже поддерживается, хоть и категорически не рекомендуется.
- Автоматическое удаление старого архива с точным гранулярным сохранением нужных эпизодов

Кластеризация

- Дублирование архива на соседний сервер
- Прозрачное кеширование архива на ретрансляторе

Проигрывание

- Немедленно доступное проигрывание через различные протоколы и через веб-интерфейс: HLS, MPEG-TS, RTSP, RTMP, DASH
- Экспорт архивных записей в MP4 файл
- Выгрузка timelapse
- Отложенный просмотр в другом часовом поясе
- Интеграция с IPTV middleware для просмотра записанных передач (Catchup TV)
- DVR API
- Скриншоты из видео и сохранение их в архиве

10.5.2 Проигрывание архива

Архивные записи можно посмотреть с помощью административного веб-интерфейса либо встроив наш DVR плеер на веб-страницу.

Аналог плеера, который вы видите в веб-интерфейсе, можно встроить на ваш сайт с помощью специального адреса embed.html с параметром `dvr`.

Кроме того, к архивным записям можно получить доступ по разным протоколам с помощью специальных URL.

Доступ к DVR архиву по специальным URL

URL-адреса для проигрывания DVR Доступ к архиву по URL может осуществляться в режиме потока и в режиме файла. Файл отличается от потока тем, что он конечен. То есть, при просмотре файла плеер будет показывать перемотку, а доступный для просмотра участок видео будет ограничен началом и концом. При просмотре потока перемотка не отображается, у потока нет конца, и он может продолжаться длительное время.

Это отличие видно и по URL. Например, файловый URL может заканчиваться на "index-1345345345354-3600.m3u8" (определены границы: начало 1345345345354 и через 3600 секунд окончание), а потоковый URL может заканчиваться на "timeshift_abs-1345345345354.ts" (определено только начало).

URL зависит от протокола, который вы используете для передачи видео, записанного в архиве.

Электронный телегид (Electronic Programme Guide, EPG) DVR можно использовать вместе с EPG. Современный подход к предоставлению архива телепередач — записывать весь эфир и потом давать просматривать прошедшие (или отматывать назад текущие) передачи, используя расписание телепередач — EPG, Electronic Program Guide, он же электронный телегид.

Информация о записанных передачах и об их времени хранится в Middleware, а Flussonic Media Server предоставляет доступ к своему архиву как к бесконечной ленте (с удобной навигацией).

Есть два режима:

- просмотр уже записанной передачи;
- просмотр передачи, которая сейчас идет.

Если передача уже прошла и закончилась, то middleware на основании EPG формирует ссылку для просмотра из архива. Пользователь получает возможность посмотреть записанную передачу, как обычный файл. Например, если передача началась в 18:15 по Москве (14:15 UTC) 27 августа и длилась час, то middleware должен при выборе передачи в списке прошедших сформировать URL вида:

```
http://FLUSSONIC-IP:PORT/STREAM_NAME/index-1409148900-3600.m3u8
```

Если передача сейчас всё ещё идет, то middleware может сформировать специальный URL к архиву, позволяющий отматывать назад прямой эфир на начало передачи. Данная функциональность, к сожалению, поддерживается далеко не на всех устройствах и приставках, но тем не менее она существует. URL для такой незакончившейся передачи будет выглядеть так:

```
http://FLUSSONIC-IP:PORT/STREAM_NAME/index-1409148900-now.m3u8
```

Подробней о EPG можно прочитать в статье про работу с IPTV middleware.

Просмотр записей архива из административного веб-интерфейса

Вы можете просмотреть содержимое архива видеорегистратора через веб-интерфейс *Flussonic UI*. Для этого:

- Перейдите в настройки потока, нажав на название видеопотока на вкладке *Media*.
- На открывшейся странице откройте вкладку **DVR**.

На странице вы увидите DVR-плеер:

1. Текущее время проигрывания.

2. Пуск/пауза.

3. Центрирование таймлайна на ползунке проигрывания.

4. Приближение или отдаление таймлайна.

5. Перемещение вперёд/назад по таймлайну.

6. Регулирование громкости проигрывания.

7. Регулирование скорости проигрывания.

8. Ползунок проигрывания.

9. Таймлайн.

10. Указатель начальной точки, с которой должен начинаться фрагмент.

11. Указатель конечной точки, в которой должен заканчиваться фрагмент.

12. Сделать скриншот записи.

13. Начальное и конечное время фрагмента.

14. Скачать фрагмент.

15. Календарь.

16. Покадровый просмотр

17. Перейти к следующему кадру или на 5 кадров вперед

18. Перейти к предыдущему кадру или на 5 кадров назад

Вы также можете открыть плеер в новой вкладке или новом окне, используя URL вида:

```
http://FLUSSONIC-IP:PORT/STREAM_NAME/embed.html?dvr=true
```

Это плеер embed.html, использующийся для встраивания видео на веб-страницы.

Проиграть архив можно также в плеере предпросмотра прямо во *Flussonic UI*.

Навигация На таймлайне (9) есть несколько зон, отмеченных разными цветами: **красный цвет** означает, что записи на этот период нет, **зелёный** означает, что запись доступна, **голубой** цвет означает текущий час, а **серый** — будущее время.

Вы можете кликнуть по любому месту на таймлайне или передвинуть ползунок проигрывания (8), чтобы начать воспроизведение видеопотока в плеере с выбранного места. Также вы можете использовать кнопки для навигации по таймлайну:

- знак прицела (3) для центрирования таймлайна на ползунке проигрывания.
- - и + (4) для того, чтобы приблизить или отдалить представленный на таймлайне промежуток времени для более точного выбора времени.
- <, > (5) для перемещения вперёд/назад по таймлайну, чтобы передвинуть показанный на таймлайне промежуток времени на более раннее или позднее время (при этом саму запись вы не перематываете).
- календарь (15) для выбора даты для просмотра/экспорта записи в этот день (если запись существует и дата не выходит за пределы определённой вами глубины архива).

Вы также можете найти в архиве конкретный момент с помощью покадрового просмотра. Это может быть полезно, например, чтобы при просмотре записи с камеры наблюдения найти лицо конкретного человека или номер машины. Для этого:

- Установите ползунок проигрывания (8) в зеленой зоне, там, где вы ожидаете найти нужный момент.
- Появится кнопка **Seek per frame** (Покадровый просмотр) (16). Нажмите ее.
- Используйте кнопки перехода к следующему кадру или на 5 кадров вперед (17), а также кнопки перехода к предыдущему кадру или на 5 кадров назад (18), чтобы найти нужный момент.

Экспорт записей DVR Вы можете скачать один или несколько фрагментов записи архива для экспорта в виде видеофайла с расширением MP4. Для этого:

- Переместите указатели границ (10 и 11) на желаемые точки или введите точное время начала и конца фрагмента (13).
- Закрепите их с помощью знака замка (13).
- Скачайте фрагмент, нажав на кнопку скачивания (14).

Подробнее см. экспорт DVR.

10.5.3 Работа с DVR через API

Flussonic предоставляет набор методов для работы с DVR через API. Это позволяет автоматизировать работу с DVR в рамках вашей инфраструктуры. Эта статья посвящена обзору методов API и примерам их использования для управления DVR и работы с ним.

Обзор API-методов для работы с DVR

Flussonic позволяет получать данные о записанном потоке, настраивать запись архива и т.д. Часть команд доступна только администратору, а другая часть доступна также и пользователям.

Администратор может изменять настройки архива или сохранить его в виде файла на диск. Пользователи могут запрашивать информацию о потоке, JPEG-скриншоты и т.п.

Ниже вы можете увидеть команды для работы с DVR через API для администраторов и пользователей.

Команды, доступные только администратору

- Настроить DVR для потока
- Защита участков DVR от удаления
- Фрагменты записей DVR
- Экспорт фрагмента записи архива в виде MP4-файла

Команды, доступные пользователям

- Получение JPEG-скриншотов из архива по времени UTC
- Генерация JPEG-скриншотов по запросу
- Получение видео-скриншотов
- Скачивание фрагмента записи архива в виде файла MP4 или MPEG-TS

Команды, доступные только администратору

Запросы, доступные только администратору, должны быть авторизованы. Администратору необходимо передать basic-токен или bearer-токен при отправке запроса. В примерах ниже мы будем использовать простую авторизацию с указанием имени пользователя и пароля для входа на сервер *Flussonic*, разделённых двоеточием, вида `username:password`.

Настройка DVR для потока Для настройки DVR для потока используйте запрос `Flussonic-API: PUT /st` и передайте необходимую конфигурацию в формате JSON:

```
curl -u username:password -X PUT "http://FLUSSONIC-IP/streamer/api/v3/
    streams/STREAM_NAME" \
> -H "Content-Type: application/json" \
> --data '{"dvr": {"root":"/storage","expiration":3600}}'
```

, где:

- `/storage` — директория, где будут храниться записи DVR,
- `3600` — глубина архива, или период хранения записей, после которого они будут удалены из директории.

Чтобы изменить настройки DVR для потока, используйте такой же запрос:

```
curl -u username:password -X PUT "http://FLUSSONIC-IP/streamer/api/v3/
    streams/STREAM_NAME" \
> -H "Content-Type: application/json" \
> --data '{"dvr": {"root":"/storage","expiration":172800}}'
```

, где:

- `/storage` — директория, где будут храниться записи DVR,
- `172800` — глубина архива, или период хранения записей, после которого они удаляются из директории.

Защита участков DVR от удаления В *Flussonic* реализован механизм защиты определенных частей архива от автоматического удаления. Для защиты записей используются ***эпизоды** — *набор метаданных об участке архива. Информация об эпизодах хранится в* Flussonic Central *и запрашивается при очистке архива.* Flussonic Media Server* не удаляет участки архива с эпизодами до тех пор, пока:

- На диске не становится критически мало места в 7Cstorage-limittag/dvr/operation/dvr-get/response%7Cstorage-limitбайтах или 7Cdisk-usage-limittag/dvr/operation/dvr-get/response%7Cdisk-usage-limitпроцентах.
- Глубина записи эпизода не превышает своего предела 7Cepisodes-expirationtag/dvr/operation/dvr-get/response%7Cepisodes-expirationepisodes_expiration.

Если вы удалите поток с эпизодами или внешняя база данных с эпизодами станет недоступна, *Flussonic* продолжит хранить участки DVR с эпизодами до окончания периода 7Cepisodes-expirationtag/dvr/operation/dvr-get/response%7Cepisodes-expirationepisode_expiration, чтобы защитить данные от внезапного удаления.

Чтобы использовать приведенные ниже вызовы API, установите и настройте *Flussonic Central*.

> Вы можете использовать для работы с эпизодами собственный конфигурационный бэкенд вместо *Flussonic Central*, см. Управление эпизодами.

Чтобы создать или обновить эпизод, отправьте в *Flussonic Central* запрос `Central-API: PUT streamer/ap`

```
curl -u username:password --request PUT 'http://127.0.0.1:9019/streamer/api
    /v3/episodes/1' \
-H "Content-Type: application/json" \
--data '{"episode_id": 1,"media": "test-879c595839","opened_at":
    1686808712,"updated_at": 1686808712,"closed_at": 1686823112}'
```

Когда надобность в хранении защищенного участка отпадет, удалите эпизод `Central-API: DELETE strea`

```
curl -u username:password --request DELETE 'http://127.0.0.1:9019/streamer/
    api/v3/episodes/1' \
```

> Обратите внимание, что логин и пароль, которые нужно использовать для авторизации API-запросов к *Flussonic Central*, отличаются от логина и пароля администратора *Flussonic*. Логин и пароль *Central* указаны в файле `/etc/central/central.conf`. Токен авторизации — это строка `username:password`, закодированная в base64.

Вы также можете просматривать список имеющихся эпизодов и информацию о конкретном эпизоде. Соответствующие запросы отражены в схеме API.

Фрагменты записей DVR *Flussonic* может вернуть список временных интервалов фрагментов записей и информацию о времени начала фрагмента и продолжительности фрагмента в каждом из них.

Чтобы получить список временных интервалов фрагментов записей DVR, используйте запрос `Flussonic-API: GET streamer/api/v3/streams/{name}/dvr/ranges`, где `{name}` — имя потока:

```
curl -u username:password http://FLUSSONIC-IP/streamer/api/v3/streams/
    STREAM_NAME/dvr/ranges
```

> Время в интервалах возвращается в формате Unix-времени.

Если вам необходимо удалить фрагмент архива с определенным интервалом вручную, используйте запрос `Flussonic-API: DELETE streamer/api/v3/streams/{name}/dvr/ranges`, где `{name}` — имя потока:

```
curl -u username:password -X DELETE http://FLUSSONIC-IP/streamer/api/v3/
    streams/STREAM_NAME/dvr/ranges \
> -H 'Content-Type: application/json' \
```

```
> --data '{"from":1675326686, "duration":7200}'
```

, где:

- `1675326686` — время начала фрагмента архива в формате Unix-времени,
- `7200` — продолжительности фрагмента в секундах.

Экспорт фрагмента записи архива в виде MP4-файла Администратор может экспортировать фрагменты архива в виде MP4-файлов и загружать их по ссылке на удаленный компьютер или в хранилище Amazon S3, а также хранить их на диске сервера.

Особенности экспорта в MP4-файл Основная проблема при экспорте архива в файл любого формата заключается в том, что необходимо указать размер в заголовке файла. Существуют различные подходы к решению этой задачи, например, некоторые программы создают временный файл на диске, а некоторые делают два прохода по архиву (на первом проходе находят все нужные фрагменты, на втором формируют файл).

Flussonic Media Server выполняет экспорт в fragmented MP4 за один проход архива без формирования временных файлов благодаря особенностям формата fMP4. Заголовок fMP4-файла не содержит информации о кадрах, поэтому его можно сформировать мгновенно. Сами данные разбиваются на небольшие независимые фрагменты каждый со своим заголовком, где уже содержится информация о кадрах.

Таким образом, загрузка экспортируемого фрагмента начинается сразу, даже если файл еще не сформирован или даже не записан. А если экспорт прервется, например из-за разрыва связи, уже скачанную часть файла все равно можно будет посмотреть.

> Есть и другие способы доступа к архиву, кроме скачивания. Попробуйте использовать проигрывание DVR вместо экспорта.

Экспорт фрагмента записи архива в виде MP4-файла по ссылке Чтобы скачать фрагмент архива в виде MP4-файла, используйте запрос `Flussonic-API: POST streamer/api/v3/streams/{n` где `{name}` — имя потока.

Обязательные параметры указываются в строке запроса:

- `{from}` — время начала фрагмента записи архива в формате Unix-времени.
- `{duration}` — продолжительность фрагмента записи в секундах.
- `{path}` — путь до MP4-файла на диске сервера или хранилище Amazon S3.

Экспорт фрагмента записи архива в виде MP4-файла на диск сервера Чтобы экспортировать фрагмент архива в виде MP4-файла и сохранить его на диске сервера, используйте команду ниже:

```
curl -u username:password -X POST "http://FLUSSONIC-IP/streamer/api/v3/
    streams/STREAM_NAME/dvr/export?from=1675159800&duration=4200&path=/home/
    example/file.mp4"
```

Здесь:

- `from=1675159800` — время начала фрагмента записи архива в формате Unix-времени.
- `duration=4200` — продолжительность фрагмента записи в секундах.
- `path=/home/example/file.mp4` — путь до MP4-файла на диске сервера.

Указанный фрагмент архива будет сохранен в виде MP4-файла `file.mp4` в каталоге `/home/example`.

Будьте осторожны при указании имени файла, чтобы не затереть существующие.

Экспорт фрагмента записи архива в виде MP4-файла в Amazon S3 Чтобы экспортировать фрагмент записи архива в виде MP4-файла и сохранить его в хранилище Amazon S3, используйте следующую команду:

При формировании ссылки для экспорта фрагмента архива в хранилище Amazon S3 вам необходимо *URL-кодирование*. **URL-кодирование** преобразовывает символы параметров в строке запроса таким образом, чтобы они корректно передавались через Интернет.

```
curl -u username:password -X POST "http://FLUSSONIC-IP/streamer/api/v3/
    streams/STREAM_NAME/dvr/export?from%3D1675159800%26duration%3D4200%26
    path%3Ds3%3A%2F%2FAWS_ACCESS_ID%3AAWS_SECRET_KEY%40s3.amazonaws.com%2
    Fbucket%2Fpath%2Fto%2Ffile.mp4"
```

Здесь:

- `from=1675159800` — время начала фрагмента записи архива в формате Unix-времени.
- `duration=4200` — продолжительность фрагмента записи в секундах.
- `path=s3://AWS_ACCESS_ID:AWS_SECRET_KEY@s3.amazonaws.com/bucket/path/to/file.` — путь до MP4-файла в хранилище Amazon S3 с данными вашей учётной записи: идентификатором ключа доступа `AWS_ACCESS_ID` и секретным ключом доступа `AWS_SECRET_KEY`.

Можно также сохранять и метаданные MP4-файла, добавив `meta=true` в строку запроса:

```
curl -u username:password -X POST "http://FLUSSONIC-IP/streamer/api/v3/
   streams/STREAM_NAME/dvr/export?from%3D1675159800%26duration%3D4200%26
   path%3Ds3%3A%2F%2FAWS_ACCESS_ID%3AAWS_SECRET_KEY%40s3.amazonaws.com%2
   Fbucket%2Fpath%2Fto%2Ffile.mp4%26meta%3Dtrue"
```

Метаданные будут записаны в атом `udta.meta.ilst.data`.

Команды, доступные пользователям

При вызове команд, доступных пользователям, необходима проверка прав доступа. Это необходимо, чтобы обеспечить безопасный доступ клиентов к данным. Для этого вы можете использовать авторизацию по токену. В примерах ниже мы не будем использовать авторизацию при запросах к API.

Получение JPEG-скриншотов из архива по времени UTC Когда вы включили скриншоты для DVR или по URL, *Flussonic* начинает создавать и писать скриншоты на диск, а вы можете получать эти скриншоты по специальным URL. Эти URL содержат время того момента, в который был получен скриншот.

Flussonic может находить JPEG-скриншоты с указанием приблизительного момента времени. Например, вы можете узнать из датчика движения, что в районе определённого времени записаны необходимые вам события, или вы можете знать необходимое время, посмотрев его на таймлайне в плеере.

Если по указанному вами моменту времени скриншот не был найден, то *Flussonic* найдёт ближайший к указанному вами момент времени, когда был сделан скриншот. *Flussonic* исправляет время на точное, если вы указали неточное в своем запросе. Он вернёт часть URL, который может быть использован для получения необходимого скриншота.

Например, запросим скриншот от 31.01.2023 13:28:11. Именно в этот момент времени скриншот записан не был, но есть скриншот в момент времени, близкий к указанному. *Flussonic* возвращает `Location` с указанием момента времени, в который был найден скриншот (в нашем примере это 31.01.2023 13:27:07). Это время которое мы можем использовать далее для доступа к JPEG-скриншотом.

Чтобы запросить JPEG-скриншот потока из DVR, используйте запрос `Streaming-API: GET /{name}/{fro`
где:

- `{name}` — имя потока,
- `{from}` — момент времени в формате Unix-времени, после которого *Flussonic* ищет записанный JPEG-скриншот или ближайший ключевой кадр для генерации JPEG-скриншота.

```
curl -v 'http://FLUSSONIC-IP/STREAM_NAME/1675171691.jpg'
...
```

```
< HTTP/1.1 302 Found
< Connection: keep-alive
< Date: Tue, 31 Jan 2023 13:28:11 GMT
...
< Location: /STREAM_NAME/1675171627.jpg
```

Генерация JPEG-скриншотов по запросу *Flussonic* может создавать JPEG-скриншоты на лету. При этом экономятся ресурсы диска. Нагрузка на процессор будет ощутимая, поэтому защищайте сервер с помощью авторизации доступа. А лучше используйте видео-скриншоты, если нет необходимости именно в JPEG.

Чтобы получить JPEG-скриншот, используйте запрос `Streaming-API: GET /{name}/{from}-preview.j`
где:

- `{name}` — имя потока,
- `{from}` — момент времени в формате Unix-времени, после которого *Flussonic* ищет записанный JPEG-скриншот или ближайший ключевой кадр для генерации JPEG-скриншота.

```
curl -v 'http://192.168.2.3:80/STREAM_NAME/1675179644-preview.jpg'
...
< HTTP/1.1 200 OK
...
< Content-Length: 5738
< Content-Type: image/jpeg
< Last-Modified: Wed, 02-May-2018 07:00:40 GMT
< X-Thumbnail-Utc: 1525244440
...here goes jpeg...
```

Получение MP4 видео-скриншотов Мы рекомендуем использовать видео-скриншоты вместо JPEG-скриншотов. Видео-скриншоты архива получают почти так же, как JPEG-скриншоты. *Flussonic* исправляет URL, если по запрошенному URL нет подходящего кадра для создания скриншота.

Чтобы получить MP4 видео-скриншот из записанного в архив DVR потока, используйте запрос `Streaming-API: GET /{name}/{from}-preview.mp4`, где:

- `{name}` — имя потока,
- `{from}` — момент времени в формате Unix-времени, после которого *Flussonic* ищет записанный MP4-скриншот или ближайший ключевой кадр для возвращения MP4-скриншота.

```
curl -v 'http://FLUSSONIC-IP/STREAM_NAME/1675252800-preview.mp4'
...
< HTTP/1.1 302 Found
< Location: /STREAM_NAME/1675252803-preview.mp4
```

Flussonic вернёт MP4-файл, состоящий из одного кадра.

Скачивание фрагмента записи архива в виде файла MP4 или MPEG-TS *Flussonic* позволяет скачать фрагмент записи архива DVR в формате файла MP4 или MPEG-TS на свой компьютер.

Чтобы скачать фрагмент архива в виде MP4-файла, используйте запрос `Streaming-API: GET {name}/ar` где:

- `{name}` — имя потока,
- `{from}` — время начала фрагмента записи в формате Unix-времени,
- `{duration}` — продолжительность фрагмента записи в секундах.

Ниже показаны примеры URL для загрузки фрагмента архива потока длиной в один час в:

- MP4:

```
http://FLUSSONIC-IP/STREAM_NAME/archive-1525186456-3600.mp4
```

- MPEG-TS:

```
http://FLUSSONIC-IP/STREAM_NAME/archive-1525186456-3600.ts
```

Вставьте подобную ссылку в браузер и начнётся загрузка файла на компьютер.

Если вам необходимо скачать фрагмент архива с определёнными дорожками, укажите их в параметре ‘filter.tracks‘ в строке запроса:

```
http://FLUSSONIC-IP/STREAMNAME/archive-1525186456-4200.mp4?filter.tracks=
    v1a1
```

10.5.4 Субтитры

Flussonic Media Server распознает DVB субтитры, читает телетекст и Closed Captions из MPEG-TS и SDI и передает по протоколам HLS и DASH.

Для *DASH* возможные выходные форматы субтитров — WebVTT и TTML, для *HLS* — только WebVTT.

Для *MSS* Flussonic передает TTML-субтитры — выходные MSS потоки будут иметь субтитры в формате TTML если они были во входящем потоке.

Субтитры SCTE-27 Flussonic просто ”пропускает” через себя без изменений. Дополнительные настройки для этого не требуются. Если во входном потоке были субтитры в таком формате, то дорожка с ними будет присутствовать и в выходном MPEG-TS потоке.

Flussonic также может отправлять телетекст с потоками, переданными на карту SDI.

Содержание:

- Распознавание DVB субтитров и конвертация в WebVTT
- Передача телетекста в HLS и DASH
- Передача скрытых субтитров в HLS и DASH
- Передача субтитров в MSS
- Передача телетекста и субтитров в MPTS/SPTS
- Передача телетекста (Teletext B) из MPEG-TS в VBI аналогового видео

10.6 DVB

10.6.1 Цифровое телевидение

Цифровое телевидение на замену аналоговому сигналу развивалось паралелльно развитию интернета и длительное время имело характеристики, недостижимые на обычном подключении к интернету. В связи с этим в DVB (digital video broadcasting) системах сильно развились решения, которые были необходимы в своё время, не очень нужны в интернете, но всё ещё актуальны, потому что они повсеместно используются.

В этой статье будут рассмотрены особенности DVB (включая распространение по IP сетям, т.е. IPTV) в контексте отличия от OTT доставки и UGC сервисов типа youtube или twitch.

В мире есть несколько разных наборов стандартов, описывающих цифровое телевидение, это DVB, ATSC, ISDB-T.

Все они на сегодняшний день технически описывают как именно упаковать телевизионный поток в MPEG-TS с соблюдением норм TR 101 290

Где живет DVB?

DVB используется в 4 случаях:

- Спутниковое ТВ, DVBS (Satellite)
- Кабельное ТВ, DVBC (Cable)
- Эфирное ТВ, DVBT (Terrestial)
- Кабельное ТВ по IP сетям, IPTV

IPTV часто не включают в DVB, но по своему смыслу, технической, кадровой организации это в чистом виде DVB.

Все эти среды характеризуются тем, что связь в них однонаправленная, широковещательная. Т.е. всем абонентам готовится один и тот же набор данных и вещается из источника. Количество потребителей вообще никак не влияет на сам источник, он и не знает сколько их. Для интернета, где количество клиентов создает нелинейную нагрузку на сервера, это всё кажется очень непривычным.

DVB в IP

Почему IPTV вне всяких сомнений относится к DVB? Потому что суть услуги ровно такая же: однонаправленная низкоинтерактивная раздача телеканалов по заранее спроектированной собственной сети доставки.

Приставки и телевизоры те же, что для DVB-C, тот же набор сервисов и т.п.

Некоторое отличие есть только с HbbTV. Это технология, позволяющая добавить немного онлайн-интерактивности, но в 20-х годах очень тяжело внедрять инновационную концепцию веб-сайта уровня ранних 90-х, так что технология всё ещё перспективная.

интерактивность DVB

За такое прекрасное инженерное достижение (полностью защитить себя от скачков нагрузки) приходится платить жуткой неинтерактивностью сервиса. Спутниковые ТВ сервисы вынуждены просить клиентов не выключать надолго приставку, чтобы она не пропустила сеанс приема ключей, а монтажник такого сервиса дозванивается в офис и просит прям сейчас отправить для новой приставки ключи в космос, чтобы она включилась. Монтажникам нетфликса этого не понять.

Сложности с интерактивным сервисом приводят к потерям денег, когда человек не может прям сейчас подключиться к спортивному пакету каналов и вместе с друзьями посмотреть финал чемпионата мира. В погоне за такими сервисами традицонные DVB сервисы сначала переехали на IP сети, а потом обзавелись т.н. Middleware, т.е. по сути веб-сайтами, на которых вставлены плееры.

Всё это для того, чтобы облегчить расставание подписчика с деньгами, когда ему прямо сейчас пригорело.

Инженерные особенности DVB

Однонаправленные среды доставки DVB, как то спутник, кабель и эфир задают следующие технические особенности:

- Постоянная скорость передачи данных. Например спутниковый передатчик на нужной частоте передает 48 мегабит и это значит ровно 48 миллионов бит в секунду: не больше и не меньше. Если на передатчике ПО решило заняться сборкой мусора на пару секунд, у потребителей будет потеря сигнала. Для инженеров это означает необходимость создавать системы мягкого реального времени (soft real time) с жесточайшим требованием по стабильности битрейта.
- Пуш модель всех сервисов. Хочется передать витрину контента или расписание передач? Это будут не страницы на веб-сайте, а специальным образом подготовленные данные, которые регулярно отправляются всем-всем даже если они не нужны потребителям. Это означает, что под все сервисы канал ограниченный, а значит их сложно наращивать.
- Исторически сложившаяся зарегламентированность всех протоколов и сервисов. В то время, как на онлайн кинотеатре могут несколько раз в день менять что угодно, DVB оказался зарегламентирован комитетами и государственными регуляторами на уровне протоколов. Это значит, что в целом работать будет любой телевизор, но перечень всех возможных сервисов крайне ограничен и не менялся годами. По сути в DVB есть только витрина контента в виде перечня телеканалов и расписание передач на этих телеканалах в довольно ограниченном виде. HBBTV, попытка нарисовать хоть какие-то кнопки поверх обычного ТВ, это опоздавшая попытка добавить интерактивности и разнообразия в стандарты, забронзовевшие пару десятилетий назад.

Для доставки телевидения по DVB используется формат упаковки MPEG-TS. Называть его протоколом

неправильно, потому что в этом стандарте нет описания взаимодействия клиента и сервера, да и в большинстве случаев нет этого взаимодействия.

MPEG-TS очень хорошо подходит под задачи DVB, потому что рассчитан на использование вне IP, когда нет никаких границ пакетов, адресов, портов. Просто поток байт, в котором надо как-то найти границы пакетов и начать показывать видео предельно быстро.

Практика использования MPEG-TS такая, что картинка начнет показываться в течении пары секунд.

MPEG-TS

В контексте DVB про MPEG-TS необходимо знать следующие темы:

- Что такое пиды (PID), CC
- Таблицы PAT, PMT, NIT - всё это нужно для организации списка телеканалов
- Таблицы EIT для передачи расписания EPG
- CBR кодирование контента, которое очень условно CBR, но уж так называют
- PCR, на который сегодня обращают сильно больше внимания, чем это на самом деле нужно

PID, CC MPEG-TS проектировался под упаковку в один толстый физический канал множества параллельных потоков данных.

В IP сетях параллельные потоки данных разделяются IP адресами и портами на обоих концах соединения. В MPEG-TS это разделение делается указанием 13-битного номера канала внутри общего потока в начале каждого пакета. Все пакеты одного размера: 188 байт, с 4-х байтным заголовков. В этих 4 байтах первый байт отведен под фиксированное число 0x47, которое используется чтобы найти в потоке байт границы пакетов (3-х байт 0x47 идущих через 187 байт точно достаточно для синхронизации). Оставшиеся 3 байта используются под PID (program identifier), Continuity Counter и прочие флажки, которые чуть менее важны, а некоторые вообще уже забыты.

Continuity Counter - это дешевый 4-битовый способ понять, что пакеты не пропущены. DVB не очень подразумевает перестановку пакетов местами (в отличие от RTP), так что проследить, что пара пакетов потерялась, CC очень помогает. Отличить потерю одного пакета от 17 уже сильно сложнее.

На выходе из Flussonic Media Server при нормальных условиях не может быть CC ошибок, потому что сам сервер никуда ничего не теряет. Могут быть или рестарты стрима, или потери пакетов дальше.

PAT, PMT и т.п. Все эти абревиатуры означают доведенный до государственного стандарта формат передачи списка телеканалов, доступных на текущей частоте и соседних. Протокол стандартизован, его все приставки и телевизоры умеют читать, никакого развития и изменений тут нет уже десятилетия.

О том, как стандарты от комитетов могут не успевать за нуждами рынка, красноречиво говорит ситуация с LCN. Чтобы строго зафиксировать порядок телеканалов у клиентов, есть какие-то способы указать порядок телеканалов, но он не стал общим стандартом, в итоге для разных телевизоров, надо указывать по-разному.

В выделенных каналах (пидах) передаются тщательно упакованные в битовые структуры записи о телеканалах. Особой расширяемостью вся эта конструкция не отличается, всё что моложе 10 лет уже обросло всякими расширениями сбоку, которые присутствуют там несколько чужеродно. Но в целом конструкция работает и даже позволяет добавлять новые кодеки.

Flussonic сам формирует все эти таблицы с нуля, никакого пропускания с входа на выход нет.

EIT EIT - это способ передать Electronic Program Guide в отдельном пиде.

Ограниченность канала метаданных прослеживается даже в дизайне этой возможности. Текущее расписание на сегодня передается сильно чаще, чем на следующие дни: чего торопиться, если до выхода передачи ещё много часов, а текущее расписание приставка должна получить быстро.

Система подготовки EPG бывает такой нетривиальной, что зачастую выносится в отдельные программы, но в Flussonic есть встроенный генератор EIT из XMLTV.

CBR кодирование С этим искуственным термином есть очень много придумок и непониманий. Во-первых, CBR в бытовом понимании есть только у звука. Это mpeg2 audio позволяет кодировать каждый фрейм в одинаковое фиксированное количество байт. У видео CBR существует разве что у тех, кто гоняет по 300-3000 мегабит на поток в сыром кодеке или с раздельной компрессией кадров. H264 это всегда VBR, хотя конечно сделать выморочную ситуацию с сжатием каждого кадра в свой размер можно.

Если взять VBR поток и посмотреть на часовые файлы, то они будут примерно одинакового размера. Это тоже в каком-то смысле CBR, просто с большим окном. В DVB это окно по стандарту — 1 секунда. Т.е. каждые 25 (или сколько у вас fps) кадров идущие подряд, не превышают в сумме нужное количество байт.

Энкодеров, которые попадают ровно в нужный битрейт нет, но если открыть DVB Analyzer, то можно увидеть красивые графики с идеально ровным битрейтом. Как же получаются красивые графики идеального CBR битрейта в анализаторе?

Вся хитрость в скрытом стафинге. В MPEGTS есть добивка до нужного трафика нуль пакетами, идущими на строго выделенном порту, а есть стафинг в H264 NAL юнитах и вот его уже анализаторы не показывают.

Так что всё что нужно от энкодера для DVB — подобраться к нужной границе снизу максимально

близко, но ни в коем случае её не превышать. Печаль в том, что даже такие гарантии для энкодеров сегодня считаются в своей массе избыточными и очень мало, кто проектирует транскодеры с расчетом на эти жесткие требования DVB.

Учитывая, что Flussonic Media Server поддерживает работу с разными энкодерами, мы можем транскодировать видео с необходимым для DVB качеством.

PCR Священная корова телевизионных инженеров, мифы о котором сильно обгоняют реальность. В большинстве статей рассказывается что-то очень интересное про фазовую подстройку частоты, что наверное должно немедленно всё объяснить программисту, у которого в руках API для декодирования, показывания картинки и таймер.

PCR - это прошитое в поток время. Просто референсное время, на которое должен ориентироваться плеер. Пришел кадр, у него есть PTS (Presentation TimeStamp), когда показывать? Да когда нужный PCR покажется в потоке, тогда и показывай. Фазовые подстройки - это всё про старые приставки, у которых собственные часы гуляли так, что рассчитывать на PCR было очень разумно.

Вокруг PCR accuracy и jitter существуют устойчивые мифы, что для его формирования нужны какие-то прецизионные часы, системы реального времени и на «обычном сервере» его не проставить. Это красивые образы уровня «железное надежнее софтверного» и не имеет никакой связи с реальностью. Формирование потока не требует систем жесткого реального времени, просто нужен софт, который учитывает сразу несколько требований из стандарта DVB и позволяет при выкидывании любого пида сохранить равномерность остальных.

PCR accuracy - это просто замер, который говорит о том, насколько не совпадает линейность роста номера пакетов и роста PCR.

Flussonic полностью выбрасывает все входящие метки PCR (и в большинстве случаев их игнорирует, потому что они не особо нужны) и правильно с нулевым джиттером проставляет их на выходе.

10.6.2 TR 101 290

Развитие цифрового телевидения (не абстрактного термина, а конкретного набора спецификаций и продуктов, объединяющихся словом DVB) попало на ту эпоху, когда государственные регуляторы успевали за развитием рынка и успевали создавать стандарты, которым все игроки должны были следовать, чтобы вообще работать на рынке.

Для того, чтобы выработать общие правила оказания услуг и получить общие измеримые показатели качества в DVB разработали набор *инструментальных* проверок, показывающих корректность формирования потока байт в MPEG-TS и этот документ называется ETSI TR 101 290

В TR101290 в явном довольно однозначном виде описано, как конкретно надо проверять поток байт от получателя, чтобы считать телевизионный поток пригодным для *качественного* анализа. Анализ качества картинки - это уже делается другими способами, там ключевые слова PSNR, SSIM, VMAF. Документ принят регуляторами в различных странах как обязательный к исполнению и является хорошим примером того, как работа регулятора улучшает общее состояние рынка, задавая понятные правила в виде однозначно трактуемого документа.

Сами проверки настолько вычислительно несложны, что без каких либо затруднений возможен постоянный мониторинг сотен и тысяч каналов. Это отличается от качественной проверки, так например VMAF практически никто не гоняет постоянно, а тестируют лишь выборочно, считанные минуты за час.

Аналогом этого документа мог бы быть документ, по которому написан mediastreamvalidator от Apple для протокола HLS или расширенный валидатор XML+MP4 для DASH.

Та часть TR101290, которая описывает сами байты, состоит из трех глав, отсортированных по критичности, называемых приоритетами. Кроме этой части есть ещё описание проверок в различных средах, например DVB-T/T2, DVB-C/S

Применимость норматива

Сам по себе документ важен только для цифрового телевидения (DVB) и прежде всего для сред без IP, т.е. кабель/эфир/спутник. Там он очень осмысленнен и важен.

Однако он не является неотъемлемой частью самого MPEG-TS, поскольку в том же HLS его требования совершенно бессмысленны. Так, например, в HLS не используется вообще PCR, что для обычного телевизионщика является крамолой.

1 приоритет

Пункт **5.2.1 First priority: necessary for de-codability (basic monitoring)** описывает самые грубые проблемы, которые делают невозможным распаковку транспортного потока. Это:

- `TS_sync_loss`, `Sync_byte_error`: отсутствие возможности синхронизации по байту 0x47

- PAT_error, PAT_error_2 на пиде 0 отсутствует PAT или регулярно теряется (информация со списком телеканалов в потоке)
- PMT_error, PMT_error_2 на перечисленных в PAT пидах недостаточно часто повторяются таблицы PMT (описание каждого телеканала)
- PID_error какие-то пиды заявлены, но на них нет пакетов в течении желаемого времени (тут стандарт плавает, оставляя выбор за пользователем)
- Continuity_count_error CC ошибки, что на самом деле является индикатором потерь, дублирования или перестановки пакетов. Чаще всего конечно потери.

На практике последний пункт вызывает непонимания при общении людей с опытом из телевидения и людей с опытом из программирования.

Для последних фраза «CC ошибка» прежде всего звучит как «код генерирует невалидный поток байт», а на самом деле оно вполне может быть «из-за микроберстов теряются пакеты и на получателе не хватает данных».

2 приоритет

Следующая глава *5.2.2 Second priority: recommended for continuous or periodic monitoring* описывает корректность генерации данных той программой, которая формировала поток байт. В DVB мире часто используется понятие ремультиплексинга, когда транспортные потоки не полностью распаковываются до кадров и обратно, а лишь частично переписываются, сохраняя неизмененной часть данных. В процессе этих перепаковок могут быть допущены ошибки, которые тоже отслеживаются этой частью документа. Так же любая из ошибок ниже (и выше) может появиться из-за того, что биты поменялись при доставке и получатель увидел не то, что отправлял источник.

- Transport_error какая-то программа слева по тракту поставила индикатор поломанного потока. Например этим можно пользоваться чтобы специально поднять алерт только у мониторинга, не разламывая проигрывание на телевизорах
- CRC_error одну из служебных таблиц поправили, а забыли пересчитать контрольную сумму
- PCR_error слишком большое расстояние между соседними метками времени потока
- PCR_repetition_error слишком редкая маркировка времени потока, надо чаще
- PCR_discontinuity_indicator_error время потока скакнуло, а индикатора разрыва/переключен между источниками не стояло
- PCR_accuracy_error самая загадочная ошибка для тех, кто думает, что PCR связан с реальным временем. Неоднородность маркировки времени потока
- PTS_error слишком редко ставят PTS. Это ошибка из тех времен, когда в MPEGTS вообще можно было не ставить таймстемпы кадров PTS/DTS. В том же HLS без этих меток ничего не будет играть, зато можно не ставить PCR.

- `CAT_error` проблема с дешифровкой потока

PCR

PCR, он же время потока - это один самых устойчивых мифов, особенно на фоне пугающего требования в 500 наносекунд точности. Ему приписывают связь с абсолютно точным временем, распускаются мифы о том, что компьютер не может выдать PCR, а волшебный железный мультиплексор сделанный на очень дорогих израильских (почему бы и нет) FPGA картах - сможет, потому что нужно точное время.

PCR - это просто номер пакета, пересчитанный по линейной формуле: `PCR = PCR_initial + (N*188*8)`

Вот и вся магия, берите бесплатно. Вся хитрость в том, что эта формула имеет смысл только в том случае, когда поток подготовлен с CBR требованиями, т.е. за каждую секунду приходит одно и то же количество байт. Для HLS эти требования отсутствуют и поэтому в HLS вообще нет никакого смысла говорить о PCR.

Наш медиа сервер Flussonic генерирует MPEG-TS поток самостоятельно, все метки времени проставляет сам, упаковывает в CBR и выдает нулевой джиттер PCR.

3 приоритет

- `NIT_error`, `NIT_actual_error`, `NIT_other_error`, `SI_repetition_error`, `TDT_error`, `RST_error`, `SDT_other_error` проверки на частоту и корректность информационных пакетов (таблиц), в которых рассказывается, что это вообще за транспортный поток и что в нём можно посмотреть
- `Unreferenced_PID` - жалоба на то, что пид есть, но он ничейный, ни в какой таблице он не перечислен. Такое бывает, когда подмешивают левые пиды и с другой стороны их забирают, т.е. когда заменяют инструментальное оформление структуры потока на голосовое.
- `Buffer_error`, `Empty_buffer_error` - ошибки HRD буфера попавшие в 3-й приоритет, хотя скорее им место во 2-м.
- `EIT_error`, `EIT_actual_error`, `EIT_other_error`, `EIT_PF_error` - всё что касается расписания, чтобы оно приходило вовремя и актуальное

HRD Buffer error

Непросто найти детали об этой простой штуке, так что рассказываем здесь.

Кадр с каким-то PTS (указанным в PES заголовке кадр) начинает передаваться получателю и тот накапливает его в буфер. Буфер растет с каждым пришедшим пакетом. Как только время потока PCR превышает указанный PTS, фрейм выкидывается из буфера целиком уменьшая его наполнение. Получается пилообразный график.

Он не должен выходить за верхнюю границу (определенную) и не опустошаться «ниже нуля», т.е. кадр должен прийти полностью к тому моменту, когда наступит указанное время потока. За это как раз отвечает транскодер, который должен достаточно ужать кадр.

Опустошение буфера одно из немногих, которое действительно влияет на разрушение картинки, ведь телевизору надо или подходить формально и декодировать разломанный кадр, или ждать, пока он прийдет полностью создавая дерганую картинку.

10.7 NDI

10.7.1 Захват NDI источника

Flussonic поддерживает захват видео по протоколу NDI.

Большинство программ предложит вам урл `ndi://Server_name (Source 1)`, вам надо превратить его в `ndi://Server_name/Source 1`

NDI кодек не позволяет показывать видео в браузере, для получения HLS или отправки на другой сервер, потребуется включение транскодера.

Для работы с NDI потребуется установить дополнительный пакет `ndi-bridge`:

```
apt install -y ndi-bridge
```

10.8 RTMP

10.8.1 Протокол RTMP

RTMP (Real Time Messaging Protocol) — проприетарный и закрытый протокол передачи видео поверх TCP. Главным плюсом RTMP стала его реализация в Adobe Flash Player, который на короткий период был главным способом показать потоковое видео в браузере и на мобильном телефоне.

Широчайшее распространение в течении короткого промежутка времени стало предпосылкой к тому, чтобы этот протокол не исчез, а остался и сегодня в качестве contribution средства. Никто сегодня не смотрит видео по RTMP, но по нему публикуют видео в социальные сети.

Применимость и ограничения

RTMP терпимо справляется с отправкой одной видео дорожки среднего качества и одной аудиодорожки среднего качества по хорошему, желательно проводному интернет соединению. Сегодня этот протокол встречается, например, в OBS - бесплатной программе для вещания с ноутбука в соцсети.

Когда RTMP не подойдет?

- Несколько языков по этому протоколу стандартными методами передать нельзя. Нужна специальная доработка клиента и сервера.
- Мультибитрейтная доставка по нему практически невозможна, это прерогатива HLS.
- Плохой канал станет непреодолимым препятствием из-за проблемы Head of line blocking, т.е. достаточно одного потерянного пакета и соединение приходит в негодность, потеря многих секунд видео.
- Качественные современные кодеки (AV1, Opus) доступны только через расширения, ещё не получившие массового рапространения.
- Нет стандартного способа передачи субтитров, как и прочих метаданных.

Технические детали RTMP

В RTMP заложено очень много идей для RPC (remote procedure call). Его авторы планировали сделать что-то типа Corba (но без схем, т.е. с соответствующей перспективой поддержки): вызовы методов, ссылки на объекты, отдельные каналы вызовов методов на объектах.

Дизайнились, но не получили распространения Shared Objects - способ синхронизации данных между подключенными клиентами. Весь RTMP вырос из сервера приложений, который должен был обеспечивать серверную логику для подключенных флеш клиентов. Видео и аудио были не первичны.

Это хорошо видно по списку ограничений:

- не больше одной видео дорожки
- не большое одной аудио дорожки
- фиксированный выбор кодеков, из которых буквально три сохраняют актуальность на сегодня (H.264, AAC, PCMA)

Итеративный дизайн RTMP привел к тому, что в нём существует много способ делать одно и то же и ещё больше ситуативных недокументированных решений, работающих в конкретной связке клиента и сервера. Плюс к этому в него заложены скрытые механизмы конкурентной борьбы.

Судебные иски Adobe В 2012-м году Adobe опубликовали спецификацию на RTMP, которая начиналась со слов «читающий эту спецификацию обязан ей следовать байт в байт и ничего не придумывать от себя», после чего сразу побежали судиться с альтернативным реализациями на основании того, что они нарушают эту спецификацию.

Не мудрено, ведь если флеш плеер подключался к серверу, написанному по этой спецификации, то он не мог проигрывать H264 (хитрым образом проверяя мусорную часть первого пакета). Правда так торопились, что вообще ничего нельзя было сделать по этой спецификации, но судебные дела были.

Аналогичная судьба постигла утилиту rtmpdump, авторы которой провели реверсинжиниринг протокола RTMPE, который заявлялся как передовая защита от копирования данных (вместо DRM). Под RTMPE уходили немалые маркетинговые бюджеты, а выяснилось, что вся безопасность основана на обфусцированном ключе, который лежит открытым текстом в каждой копии флеш плеера. Т.е. банальный TLS (и работающий по нему RTMPS) несравненно более защищен от копирования, потому что RTMPE не только не защищает от копирования клиентом, он не защищает и от прослушивания трафика.

Замедленный хендшейк RTMP На старте по протоколу необходимо дважды обменяться пакетами, в которых нет никаких данных, только мусор, который особенным образом интерпретируется в различных клиентах и серверах. Это добавляет на старте минимум 2 ненужных RTT, без которых можно обойтись.

Нарезание видео на чанки Один из артефактов идеи о сервере приложений с протоколом для RPC - нарезание видео на чанки. Это планировалось делать для того, чтобы в середине длинного кейфрейма можно было воткнуть посылку какого-нибудь вызова метода, но на практике это сегодня никому не нужно, ведь никому кроме подключенного флеш плеера это не нужно.

Особенно удачным дизайн решением было сделать дефолтный размер чанка чуть больше MTU, чем обеспечить фрагментирование пакетов и увеличить packet rate на передаче видео.

FLV в основе В основе RTMP лежит формат FLV сравнимый с ним по удачности дизайна. Это означает фиксированный набор поддерживаемых кодеков, миллисекундная точность таймстемпов и невозможность указания больше одной видео и аудио дорожки.

Миллисекундная точность указания таймстемпов недостаточна для перепаковки в mpegts или fmp4. Так, например, для убирания заикания на айфоне, приходилось перевычислять DTS аудиокадров, исходя из их реальной длительности.

Подробнее о борьбе с ограничениями кодеков ниже.

Неопределенность с URL в RTMP При дизайне этого протокола получилось так, что в нём нет классических URL.

Дело в том, что когда указывается URL: `rtmp://server/live/tvchannels/cnn`, то он будет в плеере разбит на несколько частей:

- имя сервера `server`
- application name, например `live`
- имя потока `tvchannels/cnn`

Тут возникает неоднозначность, ведь может быть правильное разбиение `live/tvchannels` и `cnn`?

Клиент угадать не может, у разных CDN-ов сложились разные дефолты. Выходов несколько:

- сокращать количество сегментов до 2-х, тогда можно угадать, где какая часть
- указывать разделитель в пути: `rtmp://server/live/tvchannels///cnn`, например тройной слеш.

Это техническое решение имеет свои причины, ведь RTMP делался для сервера приложений, вот так эти приложения и именовались.

Сегодня такое решение практического смысла не имеет, но продолжает вносить сумбур.

К этому добавляется то, что иногда можно встретить такие имена приложений: `live?key1=value1`. Если в имени потока тоже есть query string, то как их собирать в URL не очень понятно, и получаются такие монстры: `rtmp://server/live?key1=value1///cnn?key2=value2`

RTMFP, RTMPT Мимолетным явлением мелькнули RTMFP (купили компанию с её технологиями и хотели воткнуть UDP в Flash Player, но рынок не был готов, впереди было десятилетие HLS) и RTMPT — low latency вещание по HTTP. Второй был неплохой идеей, если бы не был таким монстром, который регулярным поллингом должен быть скачивать чанки несколько раз в секунду с сервера.

Enhanced RTMP

Закрепившаяся у телевизионщиков практика публиковать видео на сервера по RTMP (и чего им HTTP MPEG-TS не подошел? В нём ведь абсолютно всё лучше) привела к неприятной ситуации: протокол не позволяет покупать новое железо и внедрять новые кодеки H265 и AV1. Аналогично и со звуком, а без Opus довольно странно строить сервис.

Первая итерация добавления H265 довольно просто воспользовалась невостребованным номером кодека и этого хватило, чтобы клиент и сервер смогли объясниться.

Однако YouTube принял решение о внедрении более расширенной спецификации Enhanced RTMP в которой возможны дальнейшие добавления новых кодеков. Этот протокол ещё побудет с нами.

Реализация RTMP в Flussonic

Настройка приема и передачи потоков по RTMP в *Flussonic* выполняется стандартными способами. Важно учитывать приведенные выше особенности URL.

Также для использования RTMP/Enhanced RTMP и RTMPS необходимо задать порты во *Flussonic*.

Flussonic поддерживает:

- Проигрывание по RTMP.
- Захват по RTMP
- Прием публикации по RTMP. Пример настройки см. на странице Публикация из OBS Studio в Flussonic Media Server.
- Отправку по RTMP. Настройка осуществляется стандартным образом с учетом особенностей формирования URL, описанных ниже. Примеры настройки для популярных соцсетей приведены на странице Публикация в социальные сети.

Задание портов для RTMP/Enhanced RTMP и RTMPS

Порты для RTMP/Enhanced RTMP и RTMPS можно задать в разделе **Listeners** на вкладке **Config** -> **Settings**.

> Для использования RTMPS необходимо получить SSL-сертификат.

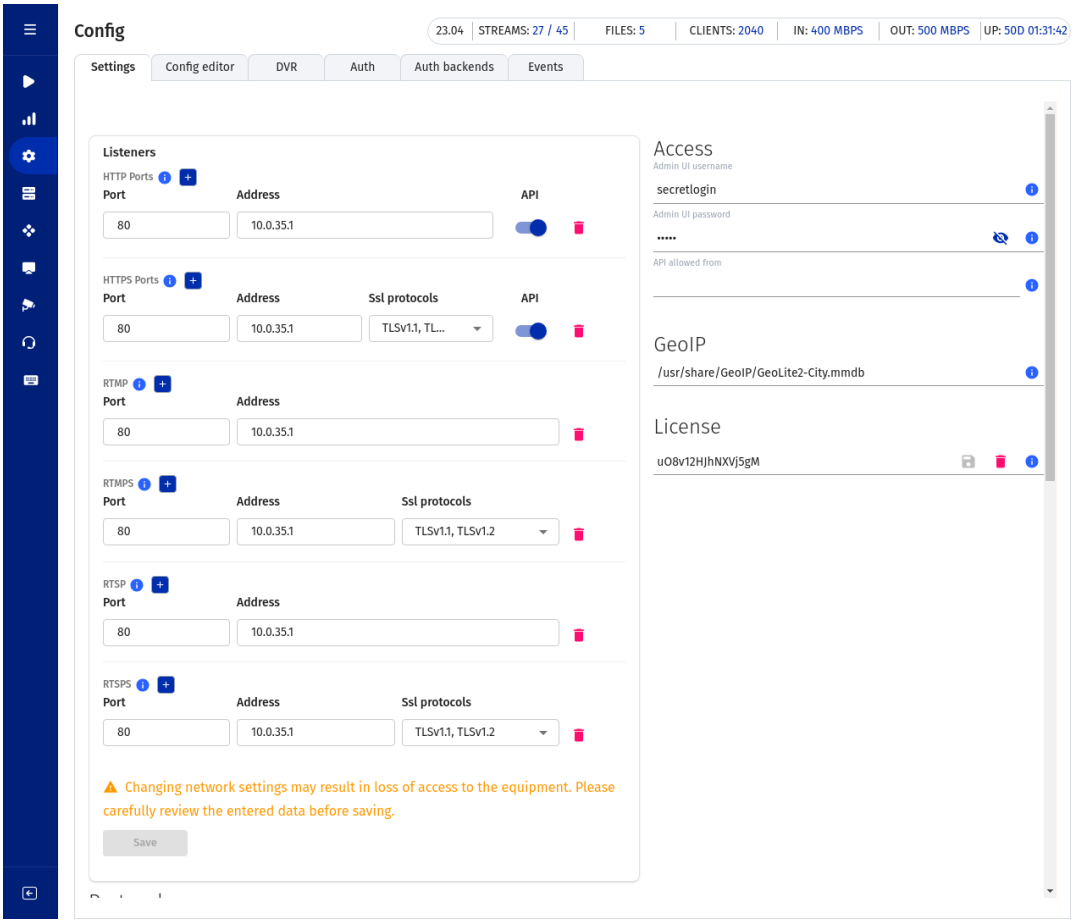


Figure 10.27: Listeners

10.8.2 Публикация по RTMP во Flussonic

Flussonic поддерживает прием публикации видео по протоколу RTMP. Публикация потоков по протоколу RTMP широко применяется при вещании живого видео через Интернет, поскольку RTMP гарантирует доставку контента, позволяя при этом получить низкую задержку. Подробнее об RTMP см. на странице Использование протокола RTMP.

Настройка публикации в *Flussonic* осуществляется стандартным образом, как описано в разделе Публикация видео на сервер. Пример настройки см. на странице Публикация из OBS Studio в Flussonic Media Server.

Ниже приведены URL, которые можно использовать для публикации из стороннего ПО во *Flussonic*. Подробнее об особенностях формирования URL при использовании протокола RTMP читайте здесь.

Для публикации по RTMP или RTMPS необходимо задать порты.

Порядок настройки

Чтобы настроить публикацию во *Flussonic Media Server* по протоколу RTMP:

- Создайте поток с источником `input publish://`, как описано здесь.
- Настройте порт для RTMP.
- Начните публиковать поток во *Flussonic*, например как описано здесь.

Публикация в статический поток

Для публикации по RTMP в статический поток можно использовать такие URL:

- `rtmp://FLUSSONIC-IP/stream_name`. В этом случае имя приложения должно быть указано как часть имени потока во *Flussonic*. Например, можете указать название потока в *Flussonic* `client15/published1`, а в стороннем приложении:
- `rtmp://FLUSSONIC-IP/static/stream_name`. В этом случае в стороннем приложении можете указать имя приложения `static`, а *Flussonic* распознает это зарезервированное слово как имя приложения и отбросит его.:

!!! note Если вы явно настроите во *Flussonic* поток с составным именем `static/stream_name`, `static` будет считаться частью имени потока.

Публикация в динамический поток

При публикации по RTMP в динамический поток используется следующая логика:

- Сервер склеивает имя приложения с путем публикации. Так пары `rtmp://FLUSSONIC-IP/template/chat-15` и `rtmp://FLUSSONIC-IP/template`, `my/chat-15` превратятся в имя публикуемого потока `template/my/chat-15`.
- Ищется первый префикс публикации, который подходит под это имя. Например, для приведенных выше URL может быть выбран префикс публикации с именем `template`.
- Дальше во всех интерфейсах авторизации и т.п. имя потока будет полное: `template/my/chat-15`.

Имейте в виду, что при публикации по RTMP по динамическому имени нельзя использовать шаблон с зарезервированным названием `static`. Если вы используете в URL сегмент `static`, этот сегмент будет считаться именем приложения и будет отбрасываться при создании потока во Flussonic. Например:

`rtmp://FLUSSONIC-IP/static/test`

В этом случае во *Flussonic* создастся поток `test`. Если вы не хотите, чтобы часть `static` отбрасывалась, вместо использования шаблона явно настройте поток для публикации со статическим именем `static/test`.

Транскодирование звука

Если публикуемый RTMP поток содержит аудио в PCMU, то вы можете транскодировать его в AAC либо указать, что аудио транскодировать не следует:

- `output_audio=(keep|add_aac|aac)`. Опция указывает, как транскодировать аудио. Можно получить результирующее аудио только в AAC (`aac`), в AAC+PCMU (`add_aac`) или оставить исходный кодек (`keep`). По умолчанию используется `keep`.

```
template chats {
  prefix chats;
  input publish://;
  output_audio aac;
}
```

10.8.3 Захват потока по RTMP

Flussonic поддерживает захват видео по протоколу RTMP. Подробнее об RTMP см. на странице Использование протокола RTMP.

URL для захвата по RTMP в общем случае имеет вид:

```
rtmp://FLUSSONIC-IP/application/stream_name
```

Протокол требует, чтобы в URL было не меньше двух сегментов. Первый сегмент, по умолчанию, используется для указания имени RTMP приложения (`application`). Подробнее об особенностях формирования URL при использовании протокола RTMP читайте здесь.

Если название RTMP приложения на сервере состоит больше чем из одного сегмента, то в адресе надо указать два слэша для явного разделения на `application` и `stream name`.

10.9 HLS

10.9.1 Воспроизведение HLS

Flussonic Media Server поддерживает раздачу видео по протоколу HLS. Многие из доступных возможностей нестандартны для HLS, но мы поддерживаем их для вашего удобства.

Поддерживаемые кодеки: H264, H265, MPEG2 video, AAC, MP3, MPEG2 audio, AC-3.

Flussonic Media Server позволяет получать по HLS прямой эфир, видео по запросу, видео из архива (catch up и timeshift).

Если входной поток содержал DVB субтитры или телетекст, то они будут переданы в выходной поток, проигрываемый по HLS, если настроить Flussonic для этого. Если поток пишется в архив, то и субтитры сохраняются в архиве.

Содержание:

Структура протокола HLS

Принцип работы протокола HLS заключается в разбиении всего потока или файла на медиа-сегменты одинаковой длины. Сегменты представляют собой файлы с расширением `.ts`, которые последовательно скачиваются по HTTP. Кроме того, HLS создает файл индекса, который содержит ссылки на файлы медиа-сегментов и сохраняется с расширением `.m3u8`.

Вы можете использовать для проигрывания HLS разные URL, в зависимости от нужного сценария, но в любом случае, по каждому из этих URL вы получите плейлист (или манифест) одного из двух следующих типов.

Медиа-плейлист Медиа-плейлист — это плейлист, в котором все строки обозначают отдельные медиа-сегменты. Каждый медиа-сегмент указан с помощью URI файла с расширением `.ts`. Плейлист дополняется новыми URI во время проигрывания. Такой плейлист используется для проигрывания простых потоков или файлов с одной видеодорожкой и одной аудиодорожкой.

Пример медиа-плейлиста:

```
#EXTM3U
#EXT-X-TARGETDURATION:7
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:38530
#EXT-X-PROGRAM-DATE-TIME:2022-03-22T08:06:37Z
#EXTINF:5.000,
2022/03/22/08/06/37-05000.ts
#EXTINF:5.000,
2022/03/22/08/06/42-05000.ts
#EXTINF:5.000,
2022/03/22/08/06/47-05000.ts
#EXTINF:5.000,
2022/03/22/08/06/52-05000.ts
```

Мастер-плейлист Мастер-плейлист — это плейлист, в котором все строки обозначают отдельные медиа-плейлисты. Каждый медиа-плейлист указан с помощью URI файла с расширением `.m3u8`. Мастер-плейлист используется для проигрывания мультибитрейтных потоков или файлов. Он позволяет клиенту динамически переключаться между потоками или файлами с разным качеством.

Пример мастер-плейлиста:

```
#EXTM3U
#EXT-X-STREAM-INF:CLOSED-CAPTIONS=NONE,RESOLUTION=640x480,FRAME-RATE
    =25.000,CODECS="avc1.4d001f,mp4a.40.2",AVERAGE-BANDWIDTH=500000,
    BANDWIDTH=630000
tracks-v3a1/mono.m3u8
#EXT-X-STREAM-INF:CLOSED-CAPTIONS=NONE,RESOLUTION=768x576,FRAME-RATE
    =25.000,CODECS="avc1.4d001f,mp4a.40.2",AVERAGE-BANDWIDTH=610000,
    BANDWIDTH=770000
tracks-v2a1/mono.m3u8
#EXT-X-STREAM-INF:CLOSED-CAPTIONS=NONE,RESOLUTION=960x720,FRAME-RATE
    =25.000,CODECS="avc1.4d001f,mp4a.40.2",AVERAGE-BANDWIDTH=650000,
    BANDWIDTH=820000
tracks-v1a1/mono.m3u8
```

Стандартное воспроизведение HLS

Если у вас есть простой live поток или файл (одно видео, один звук), то URL для воспроизведения через HLS очень простой:

```
http://FLUSSONIC-IP/STREAM_NAME/index.m3u8
```

где `FLUSSONIC-IP` — это пример адреса + порта вашего Flussonic Media Server.

Flussonic Media Server также принимает `playlist.m3u8` в конце URL для обратной совместимости с другими серверами.

При воспроизведении мультиязычного или мультибитрейтного контента есть ряд особенностей формирования URL, описанных ниже на этой странице.

Воспроизведение по HLS fMP4 для H.265

Фрагментированный MP4 (fMP4) дает важные преимущества. Прежде всего, это единственный способ **воспроизвести видео в HEVC через HLS**. Кроме того, контейнер MP4 поддерживается любым плеером, в отличие от с MPEG. Формат fMP4 также может использоваться DASH, так что только манифест будет отличаться от HLS, а кодирование MP4 будет выполняться один раз для обоих протоколов.

По следующему URL можно воспроизвести HLS с кодированием в MP4-фрагменты:

```
http://FLUSSONIC-IP/STREAMNAME/index.fmp4.m3u8
```

где `FLUSSONIC-IP` — это пример адреса + порта вашего Flussonic Media Server.

Воспроизведение по Low-Latency HLS

Flussonic поддерживает воспроизведение по Apple Low-Latency HLS (LL-HLS).

Мультиязычный HLS

Если вы хотите воспроизвести ваш мультиязычный поток на iPhone, вам нужно использовать тот же `http://192.168.2.3:80/STREAMNAME/index.m3u8`

Но если вы хотите посмотреть мультиязычный поток используя VLC или приставку, нужно включать video.m3u8.

URL для плеера:

```
http://FLUSSONIC-IP/STREAM_NAME/video.m3u8
```

Это связано с тем, что, согласно требованиям Apple HLS, для каждого отдельного языка нужно указывать отдельный плейлист с audio only вариантом. В MPEG-TS другой механизм: рядом с видео укладываются все аудиодорожки, и плеер сам выбирает, что будет проигрывать. Чтобы видео можно было посмотреть на iPhone, оно должно соответствовать требованиям Apple. Но VLC и приставки, в нарушение стандарта HLS, ожидают старый вариант MPEG-TS, преобразованный в HLS. Поэтому нужно включать `video.m3u8`.

Добавление «Audio only» для Apple

Apple требует, чтобы у всех ваших потоков был вариант без видео, только звук.

Они считают, что если пользователь смотрит видео по 3G и, оказавшись в зоне неуверенного приема, лучше у него будет только звук, чем буферизация.

Вы можете включить эту опцию в Flussonic Media Server следующим образом:

```
stream example {
  input file://vod/bunny.mp4;
  add_audio_only;
}
```

> Такая настройка может сделать ваш `index.m3u8` адрес невоспроизводимым в VLC или на приставке. В этом случае используйте `video.m3u8` (см. выше).

Отдельные плейлисты для STB-приставок

Когда у вас есть мультибитрейтный мультиязычный контент и вы хотите проиграть его на приставке, которая не поддерживает мультибитрейтные HLS-плейлисты, вы можете запросить с Flussonic Media Server отдельные плейлисты с одним видео и всеми звуковыми дорожками, как с опцией `mono`:

```
http://FLUSSONIC-IP/STREAM_NAME/mono.m3u8
```

Этот плейлист не мультибитрейтный (не вариантный), в нем URL до сегментов, в которых первая видео дорожка и все доступные звуковые дорожки.

Если вы хотите доставлять мультиязычные мультибитрейтные потоки на приставки, не понимающие стандарт Apple для мультиязычности, используйте `video.m3u8`:

```
http://FLUSSONIC-IP/STREAM_NAME/video.m3u8
```

Это мультибитрейтный плейлист, который отдает список плейлистов с разными качествами: `video1.m3u8`, `video2.m3u8` и т.д.

Отдельные плейлисты для аудио HLS-поток может содержать несколько аудиодорожек. Некоторые Smart TV (например, Samsung TV) и браузеры, поддерживающие стандарт *MSE (Media Source Extensions)*, не умеют переключаться между таковыми. Несколько аудиодорожек используется, например, для адаптирования контента для нескольких стран и народов. В результате плееры проигрывают только одну дорожку аудио, или же поток может вовсе не воспроизводиться.

Flussonic может отдать на выход плейлист с отдельным аудио. Используйте `separate_audio=true` в строке запроса HLS-плейлиста для плеера, чтобы получить отдельный плейлист с аудио:

```
http://FLUSSONIC-IP/STREAM_NAME/index.m3u8?separate_audio=true
```

Вот как будет выглядеть плейлист:

```
#EXTM3U
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="aac",NAME="eng a1",DEFAULT=YES,AUTOSELECT
    =YES,LANGUAGE="eng",URI="tracks-a1/mono.m3u8"
#EXT-X-STREAM-INF:AUDIO="aac",CLOSED-CAPTIONS=NONE,RESOLUTION=320x240,FRAME
    -RATE=25.000,CODECS="avc1.420015,mp4a.40.2",AVERAGE-BANDWIDTH=240000,
    BANDWIDTH=310000
tracks-v1/mono.m3u8
```

Эта опция работает live-трансляций, VOD и DVR.

Например, для DVR плейлист можно запросить по следующей ссылке:

```
http://FLUSSONIC-IP/STREAM_NAME/archive-1643098810-30.m3u8?separate_audio=
    true
```

Ниже приведён пример такого плейлиста для DVR:

```
#EXTM3U
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="aac",NAME="eng a1",DEFAULT=YES,AUTOSELECT
    =YES,LANGUAGE="eng",URI="tracks-a1/index-1643098810-30.m3u8"
#EXT-X-STREAM-INF:AUDIO="aac",CLOSED-CAPTIONS=NONE,RESOLUTION=320x240,FRAME
    -RATE=25.000,CODECS="avc1.420015,mp4a.40.2",AVERAGE-BANDWIDTH=240000,
    BANDWIDTH=300000
tracks-v1/index-1643098810-30.m3u8
```

Проигрывание DVR по HLS

DVR catchup playback Когда ваш поток уже записан на сервере нашим DVR, вы можете воспроизвести видео через HLS, используя время начала и длительность передачи (например, из EPG).

URL будет:

```
http://FLUSSONIC-IP/STREAM_NAME/archive-1659507585-3600.m3u8
```

Этот плейлист имеет тип **VOD** (static). Он будет отдавать список сегментов начиная с UTC 1659507585 (3 августа 2022г, 06:19:45 GMT) и на один час вперед. Это будет завершенный плейлист, он не будет меняться со временем.

Вы можете получить плейлист типа **EVENT** (append-only), если конец периода находится в настоящем или будущем. Для этого добавьте в URL параметр `event=true`. Такой плейлист будет отдавать список сегментов начиная с UTC 1659507585 (3 августа 2022г, 06:19:45 GMT) вплоть до настоящего момента. К плейлисту со временем будут добавляться новые сегменты, а все предыдущие сегменты сохранятся. Если вы вызовете тот же самый URL спустя некоторое время, когда указанный конец периода окажется в прошлом – плейлист уже будет иметь тип **VOD**.

URL `mono` даст список сегментов, содержащих все дорожки, в MPEG-TS:

```
http://FLUSSONIC-IP/STREAM_NAME/mono-1659507585-3600.m3u8
```

Более конкретный `videoN` плейлист даст список сегментов с N видео дорожкой и всеми звуковыми дорожками:

```
http://FLUSSONIC-IP/STREAM_NAME/video1-1659507585-3600.m3u8
```

и variant видео плейлист со списком `videoN` плейлистов, который можно использовать в старых видеоприставках и VLC:

```
http://FLUSSONIC-IP/STREAM_NAME/video-1659507585-3600.m3u8
```

> Не используйте URL-адрес `video` для браузеров и современных клиентов. В большинстве случаев рекомендуем использовать стандартный URL-адрес `archive` для получения плейлиста для видео с несколькими битрейтами или несколькими языками.

HLS Event playlist Вы можете запросить HLS Event плейлист по ссылке:

```
http://FLUSSONIC-IP/STREAM_NAME/index-1659507585-now.m3u8
```

Event плейлисты используются для того, чтобы дать клиенту возможность перемотатывать видео в рамках идущего мероприятия, например, вебинара, концерта или текущей ТВ программы. Такой плейлист будет отдавать список сегментов начиная с UTC 1659507585 (3 августа 2022г, 06:19:45 GMT) вплоть до настоящего момента. К плейлисту со временем будут добавляться новые сегменты, а все предыдущие сегменты сохранятся.

Обратите внимание, что плеер начнет проигрывание с прямого эфира, для доступа к архиву нужно перемотать в плеере.

Перемотка плейлиста Есть специальный плейлист **"rewind-N.m3u8"** с большим «скользящим» окном, позволяющий перематывать и ставить на паузу HLS потоки на долгие часы.

```
http://FLUSSONIC-IP/STREAM_NAME/rewind-7200.m3u8
```

7200 – длина HLS плейлиста в секундах. Это означает, что ваши клиенты могут поставить эфир на паузу на 2 часа или перемотать на начало футбольного матча без обращения по специальными архивным ссылкам.

DVR timeshift playback Если у вас есть поток, записываемый на диск, но вы не настроили отложенный поток, вы можете проиграть видео со сдвигом по времени с помощью правильно построенного HLS адреса.

Относительный таймшифт:

- Проигрывание на `ago` секунд назад: `http://FLUSSONIC-IP:PORT/STREAM_NAME/timeshift_re`

Абсолютный таймшифт:

- Проигрывание с момента `from` по UTC:`http://FLUSSONIC-IP:PORT/STREAM_NAME/timeshift_`

DVR fMP4 по HLS Для проигрывания DVR в формате fMP4 по протоколу HLS поддерживаются такие же форматы ссылок, как и для обычного HLS:

- DVR catchup: `http://FLUSSONIC-IP:PORT/STREAM_NAME/archive-{from}-{depth}.fmp4.m`
- DVR window: `http://FLUSSONIC-IP:PORT/STREAM_NAME/index-{from}-{duration}.fmp4.`
- Event playlist: `http://FLUSSONIC-IP:PORT/STREAM_NAME/archive-{from}-now.fmp4.m3u8`
- Rewind: `http://FLUSSONIC-IP:PORT/STREAM_NAME/rewind-{ago}.fmp4.m3u8`
- Абсолютный DVR timeshift: `http://FLUSSONIC-IP:PORT/STREAM_NAME/timeshift_abs-{from`
- Относительный DVR timeshift: `http://FLUSSONIC-IP:PORT/STREAM_NAME/timeshift_rel-{ag`

Подробнее о том, зачем нужно проигрывание fMP4 по HLS, см. выше.

Воспроизведение отдельных дорожек

Если у потока есть несколько аудио- и видеодорожек, то вы можете указать, какие именно дорожки следует проигрывать.

Для этого укажите номера дорожек в параметре ‘filter.tracks‘, добавив его в конец URL потока.

Примеры:

- Выбрать первую аудио- и вторую видеодорожки:

```
http://FLUSSONIC-IP/STREAM_NAME/index.m3u8?filter.tracks=v2a1
```

- Выбрать только видео:

```
http://FLUSSONIC-IP/STREAM_NAME/index.m3u8?filter.tracks=v1
```

- Выбрать Проигрывание отрывка длиной 3600 секунд из DVR архива, начиная со времени UTC 1362504585:

```
http://FLUSSONIC-IP/STREAM_NAME/archive-1362504585-3600.m3u8?filter.tracks=
    v2a1
```

Сортировка треков в мультибитрейтном плейлисте

Для мультибитрейтного потока все треки перечисляются в мастер-плейлисте. По умолчанию видеотреки сортируются по возрастанию битрейта, т.е. проигрывание начинается с видеотрека с самым низким битрейтом.

Если вы хотите изменить порядок треков и начать проигрывание с другого трека, вы можете использовать для этого параметр 'filter.tracks'. Укажите в этом параметре номера видео- и аудиотреков в нужном порядке, и в результате треки в плейлисте будут отсортированы соответственно. Например:

`http://FLUSSONIC-IP/STREAM_NAME/index.m3u8?filter.tracks=v3v2v1a2a1`

В этом случае треками по умолчанию будут `v3` и `a2`.

Добавление скриншотов в плейлист HLS

В плейлист HLS можно добавить скриншоты как специальные теги, которые сможет прочитать плеер. Это можно сделать как для потока с включенным DVR, так и для VOD-файла.

Чтобы добавить скриншоты в плейлист, добавьте в URL потока или VOD-файла опцию `?thumbnails=`.

Пример для окна DVR потока:

`http://flussonic:80/ort/index-1644304617-60.m3u8?thumbnails=50`

Пример для VOD-файла:

`http://flussonic:80/vod/bunny.mp4/index.m3u8?thumbnails=100`

Указанное значение определяет, сколько ссылок на скриншоты будет добавлено в плейлист, чтобы покрыть продолжительность окна DVR или VOD-файла соответственно. Плеер добавит на полосу проигрывания скриншоты через равные интервалы времени. Продолжительность интервала между скриншотами равна всей продолжительности окна DVR или VOD-файла, разделенной на это значение.

Если указать слишком большое число, плеер будет использовать дополнительные ресурсы, что может привести к зависанию плеера или браузера. Уменьшив этот параметр, можно ограничить количество скриншотов и, таким образом, уменьшить расход ресурсов.

Для работы этой опции в настройках потока или VOD-файла необходимо указать параметры `thumbnails enabled=ondemand` и `size` (размер скриншотов). Например: `thumbnails enabled=onde`
Можно указать несколько размеров через пробел, например, `size=320x250 size=640x480`.
В этом случае в плейлист будет включено несколько треков со скриншотами. Каждый скриншот в соответствующем треке будет пропорционально уменьшен, чтобы уместиться в указанный размер.

Больше информации можно найти в схеме Streaming API.

10.9.2 Воспроизведение LL-HLS

Flussonic поддерживает воспроизведение по Apple Low-Latency HLS (LL-HLS) — это потоковый протокол, который основан на HLS и преодолевает его высокую задержку.

LL-HLS поддерживает те же кодеки, что и HLS (H.264, AAC, MP3), а также HEVC (H.265) и AV1. Контейнер может быть MPEG-TS или fMP4 (фрагментированный MP4). *Flussonic* использует fMP4 для упаковки потоков для доставки LL-HLS. Упаковка в fMP4 производится по стандарту CMAF.

Перед использованием HLS с низкой задержкой помните, что нагрузка на сеть и сервер будет увеличиваться, поскольку этот протокол делит сегменты HLS на ещё более мелкие сегменты (также называемые чанками (chunks)).

LL-HLS URL

Чтобы проигрывать поток по протоколу Apple Low-Latency HLS, откройте в плеере ссылку **CMAF** из настроек потока на вкладке **Output**. Полная поддержка LL-HLS есть в плеере THEOplayer.

CMAF — стандарт, который используется для создания MP4-фрагментов, совместимых со спецификацией на Low-Latency HLS.*

URL имеет вид:

```
http://FLUSSONIC-IP/example_stream/index.ll.m3u8
```

Проиграть LL-HLS так же можно с помощью embed.html плеера

```
http://FLUSSONIC-IP/example_stream/embed.html?realtime=true&proto=ll-hls
```

Если в потоке есть дорожки, которые не нужны в выходном LL-HLS потоке, используйте параметр `filter.tracks` в URL. С его помощью можно отфильтровать только те треки, которые необходимы на конечном устройстве (например, чтобы не доставлять 360p на телевизоры или 4K на мобильные устройства). Подробнее о фильтрации дорожек читайте в разделе Выбор дорожек для проигрывания.

Необязательные настройки LL-HLS

LL-HLS включен во *Flussonic* по умолчанию и **не требует никаких дополнительных настроек**. Мы задали параметры проигрывания LL-HLS таким образом, чтобы добиться оптимального соотношения задержки и потребления ресурсов. Тем не менее, настройки можно менять, если при значениях по умолчанию возникают проблемы (например, долго стартует видео, сервер перегружен при небольшом количестве зрителей, частая буферизация, задержка выше ожидаемой и т.п.). Обратитесь в нашу службу поддержки, чтобы получить помощь в подборе наилучших параметров LL-HLS для ваших задач.

В файле конфигурации можно задать дополнительные параметры (**если необходимо**):

Figure 10.28: Output protocols

- 7Cbody%7Csegment-durationtag/stream/operation/stream-save%7Cbody%7Csegment-durationsegment_du
  — длительность сегмента в HLS-потоке.
- 7Cbody%7Csegment-counttag/stream/operation/stream-save%7Cbody%7Csegment-countsegment_count
  — количество сегментов в HLS-плейлисте.
- 7Cbody%7Cchunk-durationtag/stream/operation/stream-save%7Cbody%7Cchunk-durationchunk_duration
  — длительность CMAF-чанка, или частичного сегмента HLS, сегмента HLS. По умолчанию
  значение равно `200` .

Пример:

```
stream example_stream {
  input fake://fake;
  segment_duration 4;
  segment_count 4;
  chunk_duration 500;
}
```

Параметры `segment_duration` и `segment_count` можно также изменить в UI на вкладке **Output** в профиле потока.

10.10 MPEG-TS

10.10.1 Мультиплексор

Flussonic Media Server предоставляет все необходимые инструменты для организации промышленного мультиплексора, позволяющего создавать многопрограммные MPEG-TS потоки (MPTS), пригодные для передачи в спутниковую или кабельную сеть. Он не требует специального оборудования, поэтому вы сможете использовать для мультиплексирования обычный сервер (однако для захвата и отправки сигнала могут потребоваться специальные платы).

Многопрограммные потоки нужны потому, что пропускная способность канала в таких сетях ограничена, и нужно по одному физическому каналу (кабелю/каналу до спутника) передать как можно больше данных — программ. Для этого упаковывают в один поток несколько программ.

Мультиплексор не вносит никакие изменения в кодеки, разрешение и другие параметры видео, т.е. не подразумевает транскодирование, а просто передает несколько входных сигналов в один выходной. Мультиплексор *Flussonic* может принимать на вход любые потоки, которые настроены в конфигурации *Flussonic Media Server*, и поддерживает все распространенные стандарты, применяемые в сфере спутникового и кабельного телевидения.

Кому нужен мультиплексор

- Операторам спутникового телевидения, которые **принимают** телепередачи от телеканалов и **передают** их **на спутник**. Для этого *Flussonic* предоставляет средства решения следующих задач:

Принять сигнал от телеканала. Как правило потоки от телеканалов приходят в Flussonic* по SDI, HDMI или ASI, как описано на следующих страницах этого раздела.

* Подготовить MPTS для передачи в DVB-T/C/S.

* Обеспечить мониторинг работоспособности системы и информирование операторов о сбоях, например, по SNMP.

Выполнять авторизацию каналов для оплаты подъема сигнала на спутник. Например, Flussonic* поддерживает авторизацию с помощью бэкенда.

- Организациям, которые хотят **принять** сигнал от контент-провайдера, например, спутникового оператора, и **передать** его **в собственную кабельную сеть**, например, в больнице, коттеджном поселке, отеле и т.п. Для этого *Flussonic* предоставляет средства решения следующих задач:

* Перед отправкой MPTS в кабельную сеть получить данные от своего контент-провайдера: это может быть сигнал со спутника, UDP-мультикаст, потоки из интернета, например публикуемые по SRT, и даже сигнал с вышки наземного вещания.

* Подготовить MPTS для передачи в DVB-T/C/S. В частности, можно добавить в MPTS программу передач (EPG), сформировать DVB-C-совместимый поток с постоянным битрейтом (CBR), управлять служебными таблицами, например, чтобы принимать и отправлять телетекст и скрытые субтитры.

* Подать на телевизор разнообразные программы (в номер отеля, по жилому дому, по коттеджному поселку) по DVB-C (coax).

Важным преимуществом является то, что *Flussonic* не только поддерживает конвертацию из одной физической среды в другую (DVB-S -> DVB-C), а формирует новый MPTS, в котором вы можете отфильтровать необходимые передачи, поменять их порядок или дополнить своими программами, например, добавить каналы из интернет-источников, видео с камер видеонаблюдения, собственный телеканал с фильмами VOD (при условии соблюдения авторских прав), собственную телепередачу и т.п.

* Добавить в MPTS свой канал с фильмами или полезной информацией.

* Обеспечить мониторинг работоспособности системы и информирование операторов о сбоях, например, по SNMP.

* Добавить в MPTS видео с камер видеонаблюдения или домофона.

* Выполнять авторизацию зрителей в соответствии с подпиской, отключать за неуплату, предоставлять дополнительный контент за дополнительную плату. В разделе Дескремблирование рассказано о том, как устроена авторизация доступа абонентов к кабельному и спутниковому ТВ.

Также *Flussonic* поддерживает различные системы DRM, авторизацию сессий проигрывания с помощью бэкенда, интеграцию с Middleware.

Поддерживаемые стандарты

MPTS потоки можно принимать и отправлять по стандартам ASI, DVB, ATSC и ISDB, а также в виде UDP-мультикаста. Стандарты SDI и HDMI используются для передачи несжатых данных (raw data) одной программы. Во всех случаях, кроме мультикаста, для приема и отправки потоков по этим стандартам требуется специальная плата захвата. Настройка подключения различных плат захвата к *Flussonic Media Server* описана на следующих страницах этого раздела.

По стандартам DVB, ATSC и ISDB (в зависимости от региона) сигнал обычно приходит со спутника на головную станцию, где установлен *Flussonic*. Стандарты SDI, ASI и HDMI применяются в профессиональном оборудовании при передаче сигнала между устройствами на объекте оператора связи (в пределах помещения или здания), в том числе такой сигнал может прийти с головной станции на сервер захвата *Flussonic*. Таким образом вы можете использовать *Flussonic* на всех узлах инфраструктуры, и в большинстве случаев вам не понадобится никакое другое стороннее ПО и оборудование.

Подробнее о передаче данных в сетях оператора связи читайте на странице Цифровое телевизионное вещание.

См. также информацию об эмуляции raw-источников на странице Копирование потоков.

10.11 RTSP

10.11.1 RTSP

RTSP — это протокол, который позволяет соединить две точки и установить направленный поток аудио/видео с ультранизкой задержкой между ними.

Сегодня он в основном используется в IP-камерах по историческим причинам. Это хорошо продуманный, высоко расширяемый протокол, который имеет достаточно функций и возможностей для использования сегодня и не устарел за свои 30 лет существования. Мы сходим с ума по ультра-нулевой-минимальной задержке в настоящее время, поэтому этот протокол все еще актуален.

RTSP можно сравнить с SIP и WebRTC. SIP очень похож на RTSP, но используется для управления двунаправленным (или более сложной топологией) потоком и используется в IP-телефонии или системах видеоконференций. WebRTC похож на SIP, но текстовый протокол сигнализации заменен неопределенным HTTP-методом обмена информацией. Однако расширения WHEP-/WHIP для WebRTC являются прямой заменой RTSP.

По сравнению с MPEG-TS, RTSP сосредоточен на доставке аудио/видео через IP, в то время как MPEG-TS сосредоточен на предоставлении телевизионных услуг на однонаправленных медиа.

Flussonic поддерживает:

Инициатор	Направление	Описание
Flussonic	Внутреннее	прием видео от IP-камер через RT
ffmpeg	Внутреннее	прием публикации от ffmpeg через RTSP (мы не видели, чтобы это делало ка
внешняя VMS	Внешнее	воспроизведение внешним клиентам через RTSP (обычно д
Flussonic	Внешнее	трансляция на другой сервер (обычно flusson

Иногда можно встретить RTSP/2.0, но это упоминается, но не встречается на практике. Все используют разные варианты RTSP/1.0.

Существуют два разных несовместимых варианта RTSP в дикой природе:

- RTSP IP-камеры, которая передает видео и аудио в разных потоках UDP/чередующихся TCP. Обычно юникаст.
- RTSP IPTV, которая передает видео и аудио внутри MPEG-TS, который инкапсулируется в пакеты RTP, отправляемые через мультикаст.

Flussonic имеет полную поддержку первого варианта и ограниченную поддержку второго (только прием).

RTSP, SDP, RTP and RTCP

Слово **RTSP** подразумевает использование следующих стандартов:

- RTSP как текстовый протокол сигнализации. Он выглядит так же, как HTTP и очень похож на него.
- SDP — это формат для одного текстового сообщения, которое передается от источника видео к пункту назначения видео и рассказывает о содержании и способах его получения.
- RTP — это бинарный протокол для передачи видео, аудио, метаданных через UDP или TCP в том же сокете, что и RTSP.
- RTCP — это бинарный двунаправленный протокол для обмена управляющими сообщениями, связанными с передаваемым RTP.

RTSP В то время как HTTP может обходиться только двумя глаголами (GET и POST), он расширяется списком бизнес-уровневых глаголов для WebDAV, таких как MKCOL, MOVE.

RTSP имеет дюжину хорошо известных глаголов, которые используются для логического уровня. Наиболее общий список включает: `OPTIONS/GET_PARAMETER`, `DESCRIBE`, `ANNOUNCE`, `SETUP`, `PLAY`, `RECORD`, `TEARDOWN`.

Инициатор соединения отправляет запросы и получает ответы. Термины вроде `клиент` или `сервер` здесь не очень удобны, поскольку не совсем ясно, кто является клиентом, а кто сервером, когда соединяются два сервера.

Например, когда Flussonic читает видео с RTSP-камеры, он пытается отправить следующие запросы:

```
> OPTIONS rtsp://192.168.1.100/h264 RTSP/1.0
> Www-Authenticate: Basic c2VjcmV0
> CSeq: 1
>
< RTSP/1.0 200 OK
< CSeq: 1
<
> DESCRIBE rtsp://192.168.1.100/h264 RTSP/1.0
> Www-Authenticate: Basic c2VjcmV0
> CSeq: 2
>
< RTSP/1.0 200 OK
< CSeq: 2
< Content-Type: application/sdp
< Content-Length: 370
<
.... here goes SDP
....
> SETUP rtsp://192.168.1.100/h264/trackID=1 RTSP/1.0
> Www-Authenticate: Basic c2VjcmV0
> CSeq: 3
>
< RTSP/1.0 200 OK
< CSeq: 3
<
```

```
> PLAY rtsp://192.168.1.100/h264/trackID=1 RTSP/1.0
> Www-Authenticate: Basic c2VjcmV0
> CSeq: 4
>
< RTSP/1.0 200 OK
< CSeq: 4
<
> GET_PARAMETER rtsp://192.168.1.100/h264 RTSP/1.0
> Www-Authenticate: Basic c2VjcmV0
> CSeq: 5
>
< RTSP/1.0 200 OK
< CSeq: 5
<
```

Это очень ограниченный пример, просто чтобы дать вам краткое представление о том, что здесь происходит.

Обратите внимание на то, что надо точно знать полный RTSP урл вместе с путем: `rtsp://192.168.1.100/h` Без пути на сервере (`/h264`) ничего проиграть не получится.

Задача поиска RTSP путей решается протоколом Onvif

Вы можете увидеть странный вызов `GET_PARAMETER` после `PLAY`. Никто на самом деле не собирается получать какие-либо параметры, это просто пустой вызов keepalive, чтобы сообщить, что инициатор все еще жив и все еще хочет отправлять/получать видео.

Это выглядит довольно безумно, когда вы используете тот же сокет для отправки видео, но обычно код организован таким образом, что если вы отправляете мегабиты видео в секунду, отправляете пакеты RTCP, но не отправляете `GET_PARAMETER`, соединение будет прервано.

SDP Протокол описания сеанса, как вы можете догадаться, обычно не описывает сам сеанс. Он описывает медиа и иногда может описывать способ получения/отправки этого медиа.

Пример SDP:

```
v=0
o=- 1273580251173374 1273580251173374 IN IP4 axis-00408ca51334.local
s=Media Presentation
e=NONE
c=IN IP4 0.0.0.0
b=AS:50000
t=0 0
a=control:*
a=range:npt=0.000000-
m=video 0 RTP/AVP 96
b=AS:50000
a=framerate:30.0
a=control:trackID=1
a=rtpmap:96 H264/90000
a=fmtp:96 packetization-mode=1; profile-level-id=420029; sprop-parameter-
```

```
    sets=Z0IAKeNQFAe2AtwEBAaQeJEV,aM48gA==
```

Этот тривиальный SDP содержит достаточно информации для настройки декодера (`sprop-parameter-sets`
и установления воспроизведения (поле `a=control:`).

Да, этот самоочевидный и читаемый человеком текст порождает вопрос: почему бы просто не использовать JSON? Потому что он был стандартизирован за годы до популяризации JSON, и это большая удача, что он не XML.

Как и во многих других протоколах, множество полей в SDP бесполезны и ничего не изменяют, но когда вы пишете широко используемое программное обеспечение, очень трудно угадать, какие именно поля бесполезны.

RTP Протокол реального времени (RTP) является основным протоколом, используемым для доставки аудио и видео. Он бинарный и довольно простой.

Каждый пакет содержит:

- идентификатор трека (все аудио и видео треки передаются отдельно)
- идентификатор последовательности для контроля потерь пакетов, переупорядочивания, повторной передачи и т.д.
- временной код этого пакета
- необязательные расширения

RTP может быть настолько удобным, что его даже используют для передачи MPEG-TS (который также имеет счетчик непрерывности и временные метки) в IP-сетях. Это используется для повторной передачи потерянных пакетов.

RTP ориентирован на использование UDP с максимальным контролем доставки на стороне пользователя. Поэтому пакеты RTP обычно ограничиваются размером около 1400 байт, что позволяет передавать их без фрагментации. Этот размер слишком велик для аудиокадров и слишком мал для видео.

Вся спецификация для упаковки аудио/видео внутри RTP предлагает какой-то вид фрагментации и агрегации.

Идея фрагментации проста: если вы потеряете 1400 байт посреди 30КБ кадра, можно восстановить эту потерю или даже восстановить эти данные. Если вы потеряете 30КБ UDP-пакет, его почти невозможно восстановить.

RTP использует 2 байта для размера одного пакета, поэтому невозможно сделать пакет больше 64КБ. Видеопотоки высокого разрешения имеют ключевые кадры, большие этого лимита, поэтому фрагментация обязательна.

Агрегация нескольких аудио пакетов в одном кадре не так широко используется, потому что это вносит дополнительную задержку, и поэтому это не так интересно. Мы готовы тратить 80% трафика, только чтобы не увеличивать задержку в 5 раз.

RTCP Протокол управления реальным временем (RTCP) — это секретная примесь RTSP. Этот протокол немного напоминает RTP (но они различны и не могут быть разобраны одним и тем же кодом) и позволяет двунаправленный обмен статистикой в реальном времени:

- Преобразование между временным кодом RTP и абсолютным временем NTP. Да, RTSP предполагает, что у вас есть абсолютные временные метки каждого кадра, и это чрезвычайно круто.
- Обмен количеством байтов и пакетов, которые были отправлены и получены. Некоторые источники RTSP могут изменять свой битрейт, если они видят растущую джиттерность или потерю пакетов на клиенте. Точно так же, как это делает WebRTC сегодня.
- Запросы на повторную передачу пакетов.
- Другие крутые вещи, связанные с доставкой видео.

Этот протокол разделяет функции между RTSP, SIP и WebRTC. WebRTC использует наиболее сложный поднабор функций RTCP.

RTSP в IP-камерах обычно использует только пакеты типов Sender Report (код 200) и Receiver Report (код 201). Оба являются обязательными, обычно ничего не будет работать, если их игнорировать, другие типы встречаются очень редко.

Стандарты

Flussonic ориентируется на следующие спецификации для работы с RTSP:

Стандарт	Назначение
RFC2326	Базовый документ RTSP
RFC2327, RFC3266, RFC4566	SDP
RTC1889, RTC3550	RTP
RFC3984,RFC6184	H264 в RTP
RFC3016, RFC6416	AAC в RTP
RFC2035, RFC2435	JPEG в RTP
RFC7798	HEVC в RTP
RFC7587	Opus в RTP
RFC1890, RFC3551	PCMA в RTP

RFC7826 RTSP/2.0 спецификация указана как замена для RTSP/1.0, но на практике не используется.

Чтение DVR через RTSP

Воспроизведение DVR через RTSP возможно. Хотя мы не рекомендуем использовать протоколы кадров для DVR (dash гораздо лучше подходит для этого), это может быть строгим требованием для интеграции видеонаблюдения, которое использует только RTSP.

Flussonic полностью поддерживает воспроизведение DVR через RTSP.

Это реализовано с использованием следующих функций:

- В SDP есть поле `range`, которое используется для передачи списка непрерывных периодов записи.
- Заголовок `Range: clock=20230919T084734Z-20230919T084914Z` в методе PLAY.
- `GET_PARAMETER` с `position` в теле вернет текущую временную метку воспроизведения.

Onvif

Onvif — это протокол обнаружения HTTP XML (и немного UDP), который может помочь настроить IP-камеру и найти URL-адреса RTSP.

Читайте больше об этом в статье о протоколе Onvif.

10.11.2 Onvif

Onvif - это HTTP XML протокол между камерой и клиентом, позволяющий узнать RTSP урлы на камере и настроить эту камеру.

Рядом с Onvif существует ещё WS-Discovery механизм, позволяющий найти камеру в локальной сети, но он не является неотъемлемой частью onvif. Вместо него можно использовать и другие проприетарные, зачастую плохо описанные механизмы.

Onvif так же распространяется на более сложные VMS системы и на видеоаналитику, но на практике эти применения onvif существенно более редкие, менее стандартизованные и реже используются. И ещё реже реализованы по стандартам.

Как работать с Onvif

Для начала работы нужно знать, на каком порту устройства есть Onvif сервер. В силу того, что большинство реализаций Onvif написаны на библиотеке gSOAP (она GPL), её проще запустить отдельным демоном на отдельном порту, чем встраивать в основной веб-сервер, поэтому обычная ситуация, когда камера показывает свой веб-интерфейс на 80-м порту, а машинные настройки через какой-нибудь 8899.

В протоколах для поиска камер типа WD-Discovery обычно порт для onvif явно указывается.

Сам по себе Onvif является SOAP протоколом и чуть ли не единственным примером SOAP протокола в интернете, за пределами энтерпрайзного мира.

Его WSDL (т.е. жестко структурированное описание протокола) доступен на сайте onvif вместе с текстовыми описаниями.

В теории использование такого жесткого протокола как SOAP должно если не гарантировать порядок в данных, то хотя бы гарантировать порядок в их оформлении. На практике всё сильно хуже, потому что библиотек для работы с SOAP очень мало, протокол очень сложный и в итоге всё заканчивается тем, что в живом коде на камере просто лежат шаблоны, в которые подставляются данные без проверки, генерируя невалидный XML.

Как проиграть видео по Onvif?

Очень частый вопрос, на который два ответа.

Первый, формальный: никак. Onvif это текстовый протокол для настройки камеры, видео по нему играть нельзя.

Второй, по существу: воспользоваться Onvif Profile S Specification и получить урлы для проигрывания видео с камеры по RTSP.

Обычно когда противопоставляют «проигрывание по RTSP» и «проигрывание по Onvif» подразумевается, что в первом случае никакого onvif нет и нужно где-то узнать, по каким урлам отвечает камера.
Во втором можно выяснить урл по стандартизованному протоколу.

Как настроить камеру по Onvif?

Onvif предлагает широкие возможности по настройке камер, но важно знать, что с каждой моделью камер надо будет на месте разбираться с тем, как именно интерпретируется стандарт.

Например, onvif разрешает завести несколько профилей (качеств, потоков, дорожек и т.п.), позволяет их создавать, удалять, давать свои имена. Практически все камеры позволят поменять битрейт профиля, но вот удалить профили или переименовать их как вам удобнее смогут уже единицы.

Это значит, что в вашем коде по настройке разных камер нельзя будет удалить все профили и пересоздать их с нуля по единому стандарту. Вместо этого нужно будет аккуратно для каждого типа камер описывать, как именно на эту конкретную модель заливать данные.

Как получить события движения по Onvif?

Onvif позволяет подключиться к камере и получать с неё события с детектора движения.

В теории должен поддерживаться long polling, т.е. если прям в момент подключения нет никакого движения, то камера должна засыпать и ждать до 60 секунд: вдруг какое-то движение будет. Сам по себе этот механизм очень хороший, но в многих камерах ничего такого нет, она сразу вернет отсутствие движения и попросит переподключиться.

На стороне клиента нужно такое обрабатывать и снижать частоту обращений.

Onvif и политика

Onvif является торговой маркой соответствующей организации, которая подчиняется американским законам. В 2019-м году Hikvision и Dahua исключались из этой организации и им было запрещено заявлять у себя поддержку Onvif. Если честно, то немного непонятно: а кто же тогда должен был остаться в этой организации, но им виднее.

Будущее этого стандарта, авторы которого пытаются забанить более 80% мирового рынка можно помыслить себе самостоятельно.

Альтернативы Onvif

PSIA На фоне огромной сложности SOAP и наличия ровно одной доступной реализации в виде gSOAP появился и никуда не внедрился стандарт PSIA. Прекрасная идея, которая сводится к договоренностям на HTTP и RTSP урлы и существенно более простому обмену. Но, не взлетело, в камерах мало используется.

GBT28181 В Китае распространен GB/T 28181. Статус его не до конца понятен, но скорее всего увеличение количества региональных стандартов - это норма, которая нас ожидает впереди.

ISAPI Аналог PSIA, но от hikvision и реально используется. ISAPI позволяет делать всё, что могут камеры Hikvision и выглядит наиболее продвинутым. Проблема с ним одна: неясно, будет ли его когда-либо поддерживать Dahua и остальные.

10.12 WebRTC

10.12.1 Использование протокола WebRTC

WebRTC (Web Real-Time Communications) — это P2P (peer-to-peer) протокол общения между двумя клиентами, регламентирующий двунаправленную безопасную передачу данных в реальном времени.

Давайте разобьём определение на части. *P2P (peer-to-peer)* значит, что два агента (клиента) взаимодействуют друг с другом напрямую без каких-либо посредников. *Двунаправленный* указывает на функционирование в двух (противоположных) направлениях. *Безопасный* значит, что доступ к информации осуществляется через шифрование одним или несколькими протоколами безопасности. *Коммуникация в реальном времени* говорит о том, что обмен данными происходит без задержек либо с минимальными задержками.

У WebRTC есть множество преимуществ, например:

- сверхнизкая задержка (менее *одной* секунды),
- обязательное шифрование по протоколам DTLS и SRTP,
- стандарт с открытым исходным кодом,
- не требует установки программного обеспечения, плагинов и т.д.
- поддержка адаптивного потокового вещания (ABR).

Как работает WebRTC

Протокол WebRTC используется для обмена данными между браузерами (без установки плагинов или иных расширений) или другими поддерживающими его приложениями по типу ”один-к-одному”, или ”клиент-клиент”. Например, для связи двух браузеров по протоколу WebRTC необходимо зайти на один и тот же сайт в интернете.

Однако используя *Flussonic* как *сигнальный сервер* между клиентами, вы можете с легкостью организовать и связь ”один-ко-многим”. ***Сигнальный сервер*** — посредник, который позволяет передавать данные о соединении от одного клиента к другому. Следовательно, соединение из ”клиент-клиент” превращается в ”клиент-сервер-клиент”. С помощью него возможно перейти от ”один-к-одному” к ”один-ко-многим”, предоставляя бóльшую функциональность.

Чтобы начать коммуникацию между клиентами, или агентами, по WebRTC, необходимо последовательное успешное выполнение четырёх шагов:

Подробнее об этих шагах читайте в официальном руководстве по созданию приложений, реализующих протокол WebRTC.

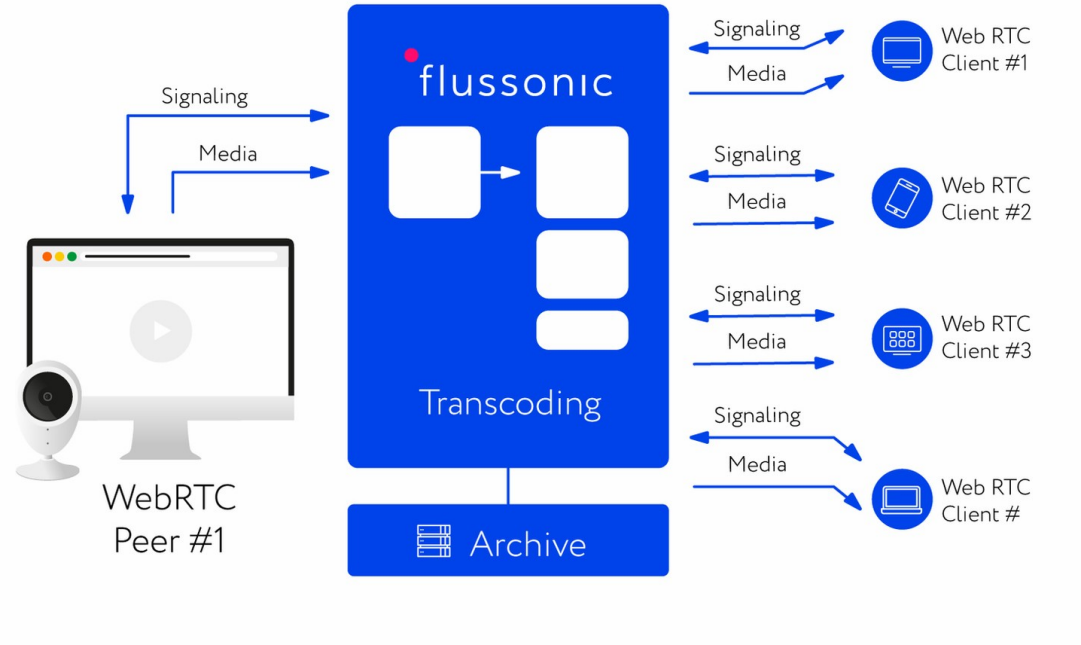


Figure 10.29: Flussonic как сигнальный сервер WebRTC

Figure 10.30

Публикация и проигрывание по WebRTC. Стандарты WHIP и WHEP

Долгое время WebRTC не использовался в области вещания и проигрывания потоков, так как у него нет стандартного протокола *сигнализации* и он слишком сложен для внедрения в инструменты и приложения для вещания. Для решения этой проблемы были разработаны стандарты **WHIP (WebRTC-HTTP ingest protocol)** и **WHEP (WebRTC-HTTP egress protocol)**. WHIP используется для публикации на Flussonic, а WHEP — для проигрывания с Flussonic.

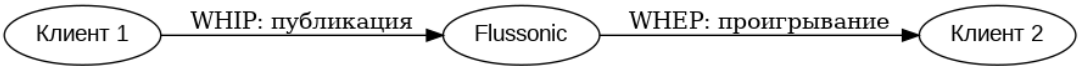


Figure 10.31

Эти протоколы предоставляют простой и независимый от медиа-сервера способ проигрывания потоков по WebRTC, который легко интегрировать в существующие инструменты вещания. Процесс согласования WebRTC в WHIP и WHEP сводится к отправке запроса HTTP POST с сообщением SDP и получению от медиа-сервера ответа 200/202 для получения ответного сообщения SDP. При этом WHIP и WHEP сохраняют все преимущества протокола WebRTC.

Flussonic оценивает состояние WHIP и WHEP сессий по полезным данным (payload) следующим образом:

- Если в течение 30 секунд *Flussonic* не получает хотя бы одного ICE-запроса на привязку или связывание (binding request) либо сообщения RTCP RR (Receiver Report), то WHEP-сессия (сессия проигрывания) завершается.
- Если в течение трёх секунд *Flussonic* не получает хотя бы одного ICE-запроса на привязку или связывание (binding request) либо RTP или RTCP SR (Sender Request) сообщения, то WHIP-сессия (сессия публикации) завершается.

Подробнее о процессе обмена данными для установления соединения между сервером (в нашем случае Flussonic) и клиентским приложением читайте в разделах **Overview** следующих стандартов:

- WHIP,
- WHEP.

Читайте также:

- Публикация по WebRTC о настройке публикации по WHIP.
- Проигрывание по WebRTC о настройке проигрывания по WHEP.

10.12.2 Проигрывание по WebRTC

Вы можете проиграть по WebRTC любой поток, например, полученный по WHIP как описано в разделе Публикация по WebRTC, или любой другой, настроенный на вашем сервере Flussonic. Для проигрывания можно использовать наш плеер `embed.html`, любой плеер с поддержкой WebRTC или приложение вашей собственной разработки.

Проигрывание WebRTC-потоков осуществляется про стандарту WHEP. О том, что такое WebRTC, WHIP и WHEP, читайте в главе Использование протокола WebRTC.

Ссылки для проигрывания потоков по WebRTC

Для проигрывания потока по WebRTC можно использовать наш плеер embed.html, который вы можете открыть в браузере по ссылке ниже:

```
http://FLUSSONIC-IP/STREAM_NAME/embed.html?proto=webrtc
```

где:

- `FLUSSONIC-IP` — IP-адрес вашего сервера *Flussonic*,
- `STREAM_NAME` — имя вашего WebRTC-потока.

Либо воспользуйтесь другим плеером с поддержкой WebRTC или приложением собственной разработки и укажите URL вида:

`http://FLUSSONIC-IP:PORT/STREAM_NAME/whep`

См. справочник Streaming API.

На клиенте нужно исполнить код для проигрывания видео по этому URL. Подробнее см. Рекомендации по созданию клиентского приложения.

Балансировка нагрузки при проигрывании по WHEP

Благодаря тому, что WHEP основан на HTTP POST-запросах, вы можете применять балансировщик нагрузки для распределения запросов на проигрывание между серверами в кластере. Балансировщик будет перенаправлять POST-запросы на серверы в кластере, используя HTTP код перенаправления 307.

Адаптивное потоковое вещание

Чтобы при проигрывании подстраивать битрейт потока под пропускную способность канала пользователя, можно настроить адаптивное потоковое вещание, то есть проигрывание с адаптивным битрейтом ABR (Adaptive Bit Rate). Таким образом каждый ваш пользователь сможет получить видео максимально возможного качества с учетом своего канала.

10.12.3 Адаптивное потоковое вещание по WebRTC

При трансляции генерируемого пользователями контента вы, вероятно, столкнетесь с необходимостью дать каждому зрителю максимальное качество видео, которое может позволить его сетевое соединения. Для этого в протоколе WebRTC предусмотрена возможность *адаптивного потокового вещания*, которое успешно реализовано в *Flussonic*.

Адаптивное потоковое вещание основано на технологии ***ABR (Adaptive Bitrate, адаптивный битрейт)***, предназначенной для эффективной доставки видео на большое количество разных устройств. Для реализации ABR из одного и того же источника с помощью транскодера генерируется несколько видео- и аудиодорожек с различными битрейтами и разрешениями — мультибитрейтный поток (MBR). Между сервером и каждым клиентским устройством устанавливается канал, в который отправляется одна из видео- и одна из аудиодорожек, причем в режиме ABR сервер сам выбирает, какую дорожку передавать в зависимости от текущей скорости сети у пользователя. При необходимости в клиентском приложении может быть предусмотрен ручной выбор дорожки.

Flussonic выбирает дорожку с приемлемым для пользователя битрейтом и качеством на основании следующих показателей, которые получает от браузера пользователя:

- Индикатор потерь пакетов — *NACK* (Negative ACKnowledgement). Это количество потерянных пакетов за единицу времени.
- Индикатор скорости передачи пакетов — *REMB* (Receiver Estimated Maximum Bitrate) или *TWCC* (Transport-wide Congestion Control). Это время доставки пакета от сервера до клиентского устройства. Подробнее об этих показателях см. Использование REMB или TWCC для ABR ниже.

Настройка ABR

Режим ABR включен для WebRTC по умолчанию. Это означает, что плееры будут работать в режиме автоматического переключения `auto`, пока зритель вручную не изменит качество видео. Чтобы включить режим `auto` вновь, необходимо выбрать его в настройках плеера.

Для того чтобы задать доступные в ABR дорожки, задайте параметры транскодера, например:

```
stream webrtc-abr {
  input fake://;
  webrtc_abr;
  transcoder vb=1000k size=1920x1080 bf=0 vb=300 size=320x240 bf=0 ab=64k
    acodec=opus;
}
```

Использование REMB или TWCC для ABR

Проигрывая потоки по WebRTC, *Flussonic* использует для отправки видео и аудио кадров протокол RTP. Для этого протокола доступны два механизма измерения пропускной способности. *Flus-*

sonic может использовать какой-либо из этих механизмов в алгоритме ABR для принятия решения о переключении битрейта на более высокое значение.

REMB Можно использовать механизм, основанный на сообщении **REMB (Receiver Estimated Maximum Bitrate)**, получаемом от клиента. Битрейт отправленного видео не может превышать битрейт, указанный в сообщении REMB. Однако если значение REMB растет, *Flussonic* может переключиться на трек с более высоким битрейтом. Подробнее о REMB.

Этот механизм довольно прост, но у него есть ряд недостатков:

- После кратковременной потери пакетов (например, из-за сбоя сетевого соединения), REMB стремительно падает, а затем растет очень медленно (в течение 5-15 минут). В результате *Flussonic* долго не может переключиться на трек с более высоким качеством, хотя клиент может его проиграть.
- *Flussonic* не может контролировать это значение, т.к. оно вычисляется на стороне клиента.
- Этот механизм отмечен как deprecated, и его дальнейшее развитие под вопросом.

Чтобы включить механизм REMB, укажите в директиве `webrtc_abr` параметр `bitrate_prober=false`.

TWCC По умолчанию *Flussonic* использует механизм, доступный как расширение RTP: **TWCC (Transport-wide Congestion Control)**. Подробнее о TWCC.

В этом случае *Flussonic* добавляет к каждому отправляемому пакету заголовок RTP, который содержит ID расширения и порядковый номер пакета. В ответ клиент отправляет сообщение RTCP, в котором содержится время получения и порядковый номер каждого полученного пакета. Таким образом, *Flussonic* знает время отправки и получения каждого пакета и может вычислить разницу между ними. Кроме того, *Flussonic* знает размер каждого пакета, так что может вычислить фактический битрейт, с которым он был отправлен.

Чтобы оценить максимально возможный битрейт, *Flussonic* периодически, через регулярные интервалы времени, отправляет группы так называемых пробных пакетов. Эти пакеты отправляются с битрейтом выше текущего. После получения этих пакетов *Flussonic* вычисляет их фактический битрейт, как описано выше. Если после очередной итерации вычисленный битрейт превышает битрейт следующей дорожки (с более высоким качеством) на 10 % и соблюдены условия по потерям пакетов (NACK), *Flussonic* переключается на следующий трек.

Этот механизм дает больше контроля и гибкости, так как большая часть его логики работает на стороне отправителя.

Вы можете задать следующие параметры TWCC в конфигурацию потока следующие параметры директивы `webrtc_abr` :

- `bitrate_prober=true` – включить использование TWCC
- `bitrate_probing_interval`– интервал отправки пробных пакетов, в секундах.

Например:

```
webrtc_abr bitrate_prober=true bitrate_probing_interval=2;
```

10.12.4 Публикация по WebRTC

Когда вы разрабатываете собственное приложение или веб-сайт для обмена видео и/или аудио по WebRTC, то можете отправлять во *Flussonic* видео и/или аудио, чтобы поддержать в своем проекте функции, расширяющие базовый протокол. Здесь вы найдете рекомендации по использованию возможностей *Flussonic* в своем проекте. Ниже описано, как настроить *Flussonic* для приема публикации по WebRTC.

Публикация осуществляется про стандарту WHIP. О том, что такое WebRTC, WHIP и WHEP, читайте в главе Использование протокола WebRTC.

Содержание:

- Настройка приема публикуемого WebRTC-потока в Flussonic.
- Опции публикации по WebRTC.

Предварительные требования

- Настройте HTTPS. В большинстве современных браузеров публикация видео и аудио потоков по WebRTC возможна только по защищенному соединению.
- Не закрывайте UDP-порты. У *Flussonic* нет фиксированного диапазона портов для WebRTC, т.е. используются те порты, которые выделит ОС. Закрыв те или иные UDP-порты, вы можете случайно попасть на используемые. Чтобы безопасно разрешить прослушивание всех портов по UDP, не используйте на сервере с *Flussonic* другое ПО.

Настройка WebRTC-потока в Flussonic

На сервере *Flussonic* добавьте в конфигурацию публикуемый поток с источником `publish://`. Это можно сделать несколькими способами:

- В UI как описано здесь
- Через API `Flussonic-API: PUT /streamer/api/v3/streams/{name}`.

Для публикации потока используйте следующий URL в вашем приложении-клиенте:

`http://FLUSSONIC-IP:PORT/STREAM_NAME/whip`

См. также справочник Streaming API, где подробно описано, как составить тело запроса.

Необязательные настройки

Создав поток с источником `publish://`, вы можете сразу включить публикацию в него. Никакие дополнительные настройки не являются строго обязательными, но они могут потребоваться

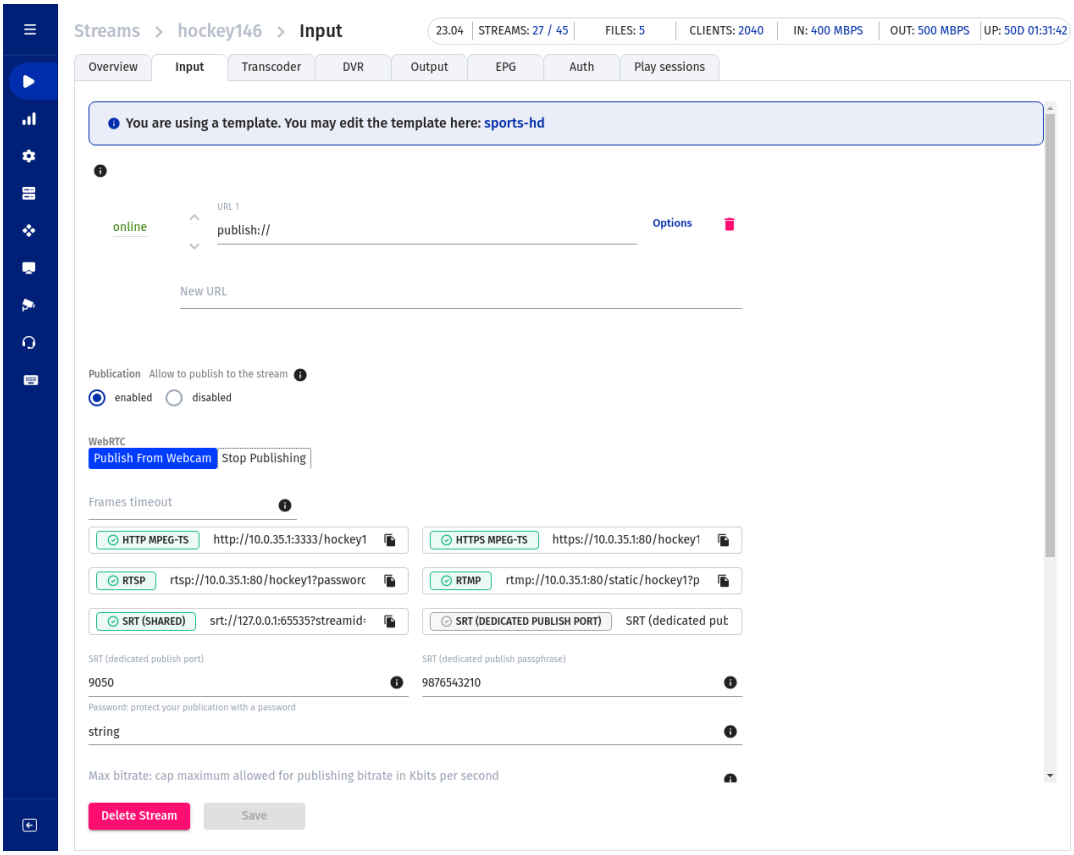


Figure 10.32: Stream settings

вам для решения тех или иных задач. Ниже приведены возможные варианты дальнейших действий с публикацией WebRTC.

Параметры потока Можно настроить параметры публикуемого по WebRTC потока, зайдя в **options** на вкладке **Input** в профиле потока. Доступны следующие параметры:

- Выходной аудиокодек — настройка транскодирования звука.
- Максимальный и минимальный битрейт — базовые настройки адаптивной публикации по WebRTC

Если вы используете публикацию в поток с динамическим именем, эти настройки задаются в шаблоне, а не в конкретном потоке. Полный список опций для WebRTC потоков см. в схеме API.

Защита публикации Чтобы никто, кроме ваших пользователей, не мог публиковать видео на сервер, вы можете защитить публикацию одним из следующих способов:

- Защита паролем.
- Внешняя авторизация сессий публикации.

Тестовая публикация с веб-камеры Чтобы проверить правильность настроек, вы можете включить публикацию с веб-камеры в созданный поток. Для этого нажмите кнопку **Publish From Webcam** на вкладке **Input** в профиле потока и разрешите браузеру доступ к камере и микрофону.

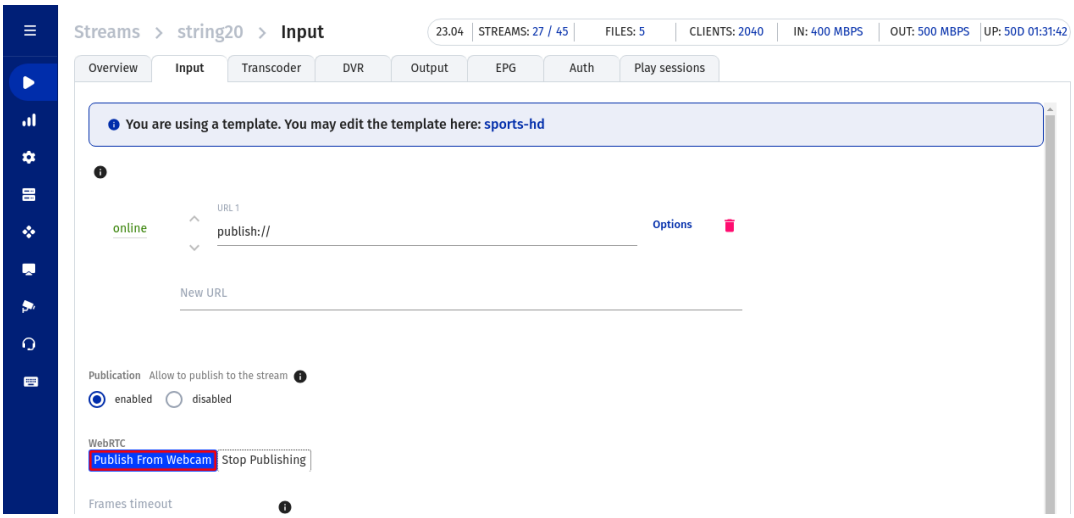


Figure 10.33: Publish from webcam

Окно предпросмотра видео с веб-камеры будет отображено здесь же.

> Этот способ публикации доступен только для тестовых целей. Конечные пользователи как правило не имеют доступа в административный интерфейс *Flussonic* и смогут опубликовать видео только через ваше приложение (веб-сайт).

Публикация с адаптивным битрейтом Публикация на *Flussonic* по умолчанию выполняется с адаптивным битрейтом (Adaptive Bit Rate, ABR). Это означает, что *Flussonic* помогает источнику публикации подстраивать битрейт под пропускную способность канала. Подробнее см. Адаптивная публикация по WebRTC.

Балансировка нагрузки при публикации по WHIP Благодаря тому, что WHIP основан на HTTP POST-запросах, вы можете применять балансировщик нагрузки для распределения запросов на публикацию между серверами в кластере. Балансировщик будет перенаправлять POST-запросы на серверы в кластере, используя HTTP код перенаправления 307.

Опции публикации в WebRTC Player Если вы используете для публикации *WebRTC Player*, следуя рекомендациям по созданию клиентского приложения, то сможете гибко настраивать

публикацию. Подробные инструкции и примеры есть в readme. Например, вы можете сконфигурировать плеер, чтобы:

- Организовать аудио-подкасты по WebRTC. Для этого установите опции `video: false` и `audio: true` в `constraints`.
- Накладывать пользовательские фильтры на видео с помощью `canvas`. Пример страницы с *WebRTC Player* включает использование `canvas`. Если вы попробуете запустить этот пример, то увидите, что на публикуемое видео наложен фильтр "sepia", а поверх видео отображается текст "It's publishing to Flussonic!".

10.13 DASH

10.13.1 Воспроизведение DASH

Flussonic Media Server поддерживает раздачу видео по протоколу DASH.

Поддерживаемые кодеки: H264, H265, AAC, MP3, AC-3.

Flussonic Media Server позволяет получать по MPEG-DASH прямой эфир, видео по запросу и видео из архива (catchup и со сдвигом по времени).

Если входной поток содержал DVB субтитры или телетекст, то они будут переданы в выходной поток, проигрываемый по MPEG-DASH, если настроить Flussonic для этого. Если поток пишется в архив, то и субтитры сохраняются в архиве.

Для передачи информации о потоке в протоколе DASH используется файл-манифест. Для простоты мы называем его здесь "плейлист".

На этой странице:

Простое воспроизведение DASH

Если у вас есть live поток или файл (один видео трек, один аудио трек), то URL для воспроизведения через DASH очень простой:

```
http://FLUSSONIC-IP/STREAMNAME/index.mpd
```

где `flussonic-ip` нужно заменить на адрес и порт вашего Flussonic Media Server.

Воспроизведение отдельных дорожек

Если у потока есть несколько аудио- и видеодорожек, то можно указать, какие именно дорожки следует отдавать. Для этого укажите номера дорожек в параметре ‘filter.tracks‘, добавив его в конец URL потока.

Примеры:

- Выбрать первую аудио- и вторую видеодорожки:

```
http://FLUSSONIC-IP/STREAMNAME/index.mpd?filter.tracks=v2a1
```

- Выбрать только видео:

```
http://FLUSSONIC-IP/STREAMNAME/index.mpd?filter.tracks=v1
```

- Выбрать вторую видеодорожку и первую аудиодорожку для проигрывания отрывка архива DVR длиной 3600 секунд, начиная со момента времени UTC 1362504585:

```
http://FLUSSONIC-IP/STREAMNAME/archive-1362504585-3600.mpd?filter.tracks=
    v2a1
```

Проигрывание архива по DASH (Catchup DVR)

Когда ваш поток уже записан на сервере нашим DVR, вы можете воспроизвести видео через DASH, указав время начала и конца передачи (например, взятые из EPG).

URL для проигрывания из архива:

```
http://FLUSSONIC-IP/STREAMNAME/archive-1362504585-3600.mpd
```

Такой URL будет отдавать список сегментов начиная с UTC 1362504585 (2013, Март, 5, 17:29:45 GMT) и на один час вперед (3600 секунд).

Если в потоке будет больше одной звуковой дорожки или больше одного битрейта, то будет доступен адаптивный стриминг и переключение языков.

Проигрывание по DASH с перемоткой назад

Есть специальный плейлист **”rewind-N.mpd”** с большим «скользящим» окном, позволяющий перематывать и ставить на паузу DASH потоки на долгие часы.

```
http://FLUSSONIC-IP/STREAMNAME/rewind-7200.mpd
```

Здесь **7200** — длина DASH плейлиста в секундах. Это означает, что ваши клиенты могут поставить эфир на паузу на 2 часа или перемотать на начало футбольного матча без обращения по специальными URL для DVR архива.

А также есть плейлист с возможностью получить прямой эфир и отмотать его назад до указанного момента в секундах (timestamp): **"archive-N-now.mpd"**, где N — Unix timestamp того момента, до которого можно будет отмотать поток.

```
http://FLUSSONIC-IP/STREAMNAME/archive-1362504585-now.mpd
```

DVR timeshift playback

Здесь мы опишем ещё один способ проигрывания архива по DASH с возможностью перемотки до указанного произвольного времени. Если вы не создали отложенный поток, то вы всё равно можете проиграть видео по DASH со сдвигом по времени с помощью правильно построенного URL.

Пример URL для абсолютного таймшифта:

```
http://FLUSSONIC-IP/STREAMNAME/timeshift_abs-1584435600.mpd
```

Где 1584435600 — 03/17/2020 @ 9:00am (UTC)

Плеер начнет воспроизведение с live и даст возможность перемотки назад до 1584435600.

DASH-манифест для проигрывания DVR на телевизорах под WebOS

Flussonic может создавать манифест DASH двух типов: с несколькими периодами и с одним периодом.

Первоначально Flussonic разработал свой DASH манифест для воспроизведения архивов, записанных в CDN. Манифест с несколькими периодами был пригоден для этой цели.

Однако такой манифест несовместим с широким спектром устройств и телевизоров, используемых потребителями во многих странах, например в США. К ним относятся телевизоры LG на WebOS и др.

Для устройств, которые не могут воспроизводить DASH с многопериодной временной шкалой, мы разработали однопериодный манифест, позволяющий воспроизводить DASH на этих устройствах.

Добавьте `period=mono` к URL следующим образом:

```
http://FLUSSONIC-IP/STREAMNAME/archive-TIME-DURATION.mpd?period=mono
```

или

```
http://FLUSSONIC-IP/STREAMNAME/archive-TIME-now.mpd?period=mono
```

Замечание. Одно периодный манифест для live с возможностью просмотра записанного архива (archive-TIME-now.mpd?period=mono) чувствителен к качеству источника входного потока — необходимо, чтобы не было пробелов в вещании потока.

Включение соответствия DASH манифеста DVB профилю

Если вы используете валидатор для DASH и выбрали в нем проверку на соответствие манифеста DVB профилю, нужно обеспечить такое соответствие.

Для этого добавьте к URL потока опцию `dvb=1`:

```
http://FLUSSONIC-IP/STREAMNAME/index.mpd?dvb=1
```

Проигрывание потока с субтитрами

Flussonic поддерживает передачу как TTML, так и WebVTT субтитров в DASH потоки, что позволяет показывать субтитры на большем количестве устройств и приставок.

Выбор субтитров для проигрывания по DASH Поскольку в манифест DASH включены два формата субтитров, вы можете выбрать один из них при проигрывании выходного потока:

```
https://FLUSSONIC-IP/STREAMNAME/index.mpd?text=wvtt
```

или (TTML используется по умолчанию)

```
https://FLUSSONIC-IP/STREAMNAME/index.mpd?text=ttml
```

Опцию `text` также можно использовать в запросах со «скользящим» окном:

```
http://FLUSSONIC-IP/STREAMNAME/rewind-7200.mpd?text=wvtt
```

Добавление скриншотов в плейлист DASH

В плейлист DASH можно добавить скриншоты как специальные теги, которые сможет прочитать плеер. Это можно сделать как для потока с включенным DVR, так и для VOD-файла.

Чтобы добавить скриншоты в плейлист, добавьте в URL потока или VOD-файла опцию `?thumbnails=`.

Пример для окна DVR потока:

`http://flussonic:80/ort/archive-1643013512-now.mpd?thumbnails=50`

Пример для VOD-файла:

```
http://flussonic:80/vod/bunny.mp4/Manifest.mpd?thumbnails=100
```

Указанное значение определяет, сколько ссылок на скриншоты будет добавлено в плейлист, чтобы покрыть продолжительность окна DVR или VOD-файла соответственно. Плеер добавит на полосу проигрывания скриншоты через равные интервалы времени. Продолжительность интервала между скриншотами равна всей продолжительности окна DVR или VOD-файла, разделенной на это значение.

Если указать слишком большое число, плеер будет использовать дополнительные ресурсы, что может привести к зависанию плеера или браузера. Уменьшив этот параметр, можно ограничить количество скриншотов и, таким образом, уменьшить расход ресурсов.

Для работы этой опции в настройках потока или VOD-файла необходимо указать параметры `thumbnails enabled=ondemand` и `size` (размер скриншота). Например: `thumbnails enabled=ondem`
Можно указать несколько размеров через пробел, например, `size=320x250 size=640x480`.
В этом случае в плейлист будет включено несколько треков со скриншотами. Каждый скриншот в соответствующем треке будет пропорционально уменьшен, чтобы уместиться в указанный размер.

Больше информации можно найти в схеме Streaming API.

10.14 SRT

10.14.1 SRT

SRT (Secure Reliable Transport) — это протокол передачи видео, похожий на RTP и основанный на UDP, который инкапсулирует MPEG-TS и добавляет к нему механизм повторной передачи.

Основные характеристики SRT:

- Протокол основан на UDP.
- Компенсация потери пакетов осуществляется за счет повторной передачи с фиксированной задержкой, но возможно использование FEC (Forward Error Correction).
- Опциональное шифрование с фиксированным секретом (`passphrase`), известным обеим сторонам. Современные безопасные технологии вроде DTLS не используются.
- Не поддерживается проверка сертификатов TLS.
- Нет сигнализации метаданных: обычно для различения потоков используются порты.
- Нет перенаправлений, повторных попыток подключения и других функций, необходимых для создания системы крупного масштаба.
- Нулевая совместимость с существующей инфраструктурой HTTP: все балансировщики должны быть реализованы заново.

По сравнению с протоколами, основанными на TCP, он решает только проблему блокировки очереди и обеспечивает **стабильную задержку**.

Протокол SRT произошёл от UDT — протокола для доставки файлов через Torrent, но особенности трансляции в реальном времени подразумевают, что вам не нужно задумываться о механизмах управления перегрузками: у вас почти постоянный битрейт трафика и нужно отправлять все данные. Не быстрее, не медленнее, просто стараться отправить каждый кадр.

Реализация SRT в Flussonic

SRT не имеет надежного механизма для определения:

- имени потока, который будет передан по данному соединению,
- того, хочет ли клиент проигрывать или публиковать видео.

Поэтому *Flussonic* поддерживает несколько способов настройки порта для SRT:

- Один порт для одного потока (рекомендуется) или мультиплексированный порт для множества потоков.

- Порт только для воспроизведения/публикации или "однонаправленный" порт, при использовании которого клиент сам должен указать режим.

Протокол SRT предлагает отправку от клиента к серверу только одной строки, которая может быть прочитана и интерпретирована на сервере. Эта строка будет прочитана только если совпадает `passphrase`, поэтому предполагается, что вы не можете иметь разные `passphrase` для разных потоков (если не считать костылей вроде угадывания одной из нескольких подходящих `passphrase`). Только одна правильная `passphrase` на порт.

По каким-то причинам SRT использует собственный стандарт для кодирования этой строки, и это не стандартное кодирование URL, как в других протоколах, а конструкция вида `#!::r=my-stream,m=pu`
То есть в начале идут символы `#!::`, а затем стандартные KEY=VALUE , KEY=VALUE.

У некоторых параметров есть определенные значения, описанные ниже.

> Некоторые клиенты до сих пор не поддерживают передачу строки `streamid`, поэтому нельзя рассчитывать на наличие имени потока и режима. Мы рекомендуем использовать отдельные порты с указанным режимом.

Совместимость SRT с облаком

Отсутствие перенаправлений, балансировки и современных средств шифрования в SRT становится проблемой, когда у вас есть кластер из нескольких десятков серверов, принимающих публикации от клиентов.

Вы не можете дать клиенту один IP-адрес, потому что этот сервер может быть выключен по любой причине. Поэтому вы предоставляете имя хоста, за которым стоит пул серверов. При этом помните, что на всех этих серверах должен быть зарезервирован один и тот же порт для этого клиента.

Учитывая проблему с отсутствием стандартного способа сигнализации о том, хочет ли клиент публиковать или проигрывать, а также проблемы с `passphrase`, мы разработали систему балансировки, основанную на портах, вместо имен потоков.

Когда клиент подключается к порту SRT, *Flussonic* может сделать HTTP-запрос на конфигурационный бэкенд, сообщая IP-адрес сервера и порт сервера. Бэкенд знает, что это должен быть клиент N с потоком S и сообщит все детали конфигурации порта в ответе. Таким образом, порт SRT будет настроен динамически при первом запросе. Подробнее см. в схеме.

Режимы SRT

Протокол SRT имеет несколько режимов работы: *Caller* (Инициатор), *Listener* (Слушатель) и *Rendezvous*. *Flussonic* поддерживает следующие режимы SRT:

- Прием публикации по SRT. *Flussonic* выступает в роли *Listener*, т.к. ожидает получения

потока от Инициатора (Caller).

- Захват SRT. *Flussonic* выступает в роли *Caller*, т.к. сам инициирует соединение, запрашивая поток из источника.
- Отправка SRT (push). *Flussonic* выступает в роли *Caller*, т.к. сам является источником потока и отправляет поток на другой сервер.
- Проигрывание SRT. *Flussonic* выступает в роли *Listener*, т.к. ожидает запроса потока от Инициатора (Caller).

Режим задается в URL потока с помощью `streamid`.

Streamid

Для передачи потока по SRT обычно необходимо указать параметр `streamid`. Это строка длиной до 512 символов, которая задается на сокете перед установлением соединения. Она отправляется в составе вызова *Caller* и регистрируется на стороне *Listener*. Основываясь на переданной в ней информации, *Listener* может принять или отклонить соединение, выбрать нужный поток данных или задать нужный пароль для подключения.

Эта строка начинается с символов `#!::`, после которых могут быть указаны следующие параметры:

- `r=` — имя потока
- `m=` — ожидаемый режим соединения: `publish` (если *Caller* хочет отправить поток) или `request` (если *Caller* хочет получить поток)
- `password=` — пароль для авторизации публичного сеанса (не рекомендуется, лучше используйте параметр `passphrase` в параметрах URL, т.к. его ожидает большинство клиентов)

Во время SRT сеанса между двумя экземплярами *Flussonic* в `streamid` автоматически добавляются следующие параметры:

- `s=` — идентификатор сеанса
- `a=` — версия *Flussonic*

Пример: `streamid="#!::r=my-stream,m=publish"`.

См. также документацию разработчика.

Порт для SRT

При захвате и отправке по SRT настройки порта и `streamid` задаются на стороне клиентского приложения, которое формирует URL потока. При настройке публикации и проигрывания URL потока формируется на стороне *Flussonic*, поэтому важно выбрать правильный способ указания порта для потоков. *Flussonic* предлагает гибкую настройку SRT-портов для передачи и получения видео. От того, какой способ задания порта вы выберете, будет зависеть вид `streamid`, который вам нужно будет использовать для приема или передачи потоков. Есть следующие варианты настройки:

- Задать отдельный SRT-порт для каждого потока или группы потоков (в шаблоне). Это самый оптимальный вариант, поддерживаемый большинством сторонних клиентов. Можно использовать в конфигурации потока опцию `srt`, если для проигрывания и публикации будет использоваться один порт, либо `srt_play` и `srt_publish`, если порты для проигрывания и публикации разные. Задавать имя потока в параметре `r=` необязательно, т.к. оно уже задано в конфигурации. Если используете `srt`, обязательно указывайте режим соединения в параметре `m=` в `streamid`.
- Задать глобальный порт для всех режимов приема и передачи всех SRT потоков на вашем сервере. Этот способ больше подходит для тестовых целей, так как не поддерживается сторонними клиентами. Задать порт можно в UI в разделе **Config** - **Settings**.

Соответствующая глобальная опция в конфигурационном файле `srt`.

Также можно задать отдельные глобальные порты для проигрывания `srt_play` и публикации `srt_publish`. Это можно сделать только в конфигурационном файле. Такой режим также не поддерживается большинством клиентов.

Если используете `srt`, обязательно указывайте режим соединения в параметре `m=` в `streamid`, а при использовании `srt_play` и `srt_publish` эта опция не обязательна, т.к. режим соединения по указанному порту и так известен. Задавать имя потока в `r=` обязательно, т.к. при использовании глобального порта непонятно, к какому потоку вы хотите обратиться.

- Задать диапазон портов для публикации и проигрывания потоков в директиве `listeners`. Этот способ используется совместно с `config_external`, когда нужно определять имя потока по его SRT-порту. Подробнее см. в схеме.

> Если вы определяете **одновременно** глобальные и локальные настройки портов, локальные имеют больший приоритет и применяются первыми.

Сводная таблица значений `streamid` для различных режимов соединения в зависимости от способа указания порта:

| | Локальный `srt` | Локальные `srt_play/srt_publish` | Глобальный `srt` | Глобальные `srt_play/srt_pub
| | ---- | ------ | ------------- | ---------- | ------- | | Публикация*|

”#!::m=publish” | Можно не указывать `streamid` | ”#!::r=stream_name,m=publish” | ”#!::r=stream_name” | | Отправка | ”#!::m=publish” | Можно не указывать `streamid` | ”#!::r=stream_name,m=publish” | ”#!::r=stream_name” | | Проигрывание | ”#!::m=request” | Можно не указывать `streamid` | ”#!::r=stream_name,m=r | ”#!::r=stream_name” | | Захват* | ”#!::m=request” | Можно не указывать `streamid` | ”#!::r=stream_name,m=reques | ”#!::r=stream_name” |

> * Режим в `streamid` является характеристикой источника, поэтому для публикации указан `m=publish`, хотя *Flussonic* работает в режиме *Listener*, а для захвата `m=request`, хотя *Flussonic* выступает в роли *Caller*.

Параметры SRT

Во всех режимах приема и передачи потоков по SRT *Flussonic* поддерживает следующие параметры URL:

Параметры	Тип данных	
`minversion`	x.y.z	
`version`	x.y.z	
`enforcedencryption`	булево значение	Если значение равно `true`, то отправитель и получатель долж
`passphrase`	строка	
`timeout`	секунды	
`linger`	секунды	
`connect_timeout`	секунды	
`latency`	миллисекунды	

10.14.2 Воспроизведение SRT

Flussonic поддерживает проигрывание SRT-потоков. Подробнее об SRT см. на странице Использование протокола SRT.

Настройка SRT-порта обычно производится для *одного потока*, т. е. один SRT-поток на один порт. Кроме того, *Flussonic* предоставляет Вам способ настроить один *глобальный* SRT-порт для *нескольких потоков*. Например, если Вы используете протокол SRT для ретрансляции. От способа задания порта зависит вид URL для проигрывания, подробнее читайте здесь.

Один SRT-поток на один порт

Чтобы настроить SRT-порт для проигрывания потока, используйте `srt_play` в настройках потока:

```
stream example_stream {
  input fake://fake;
  srt_play {
  port 9998;
  }
}
```

Формат ссылки для проигрывания потока выглядит следующим образом:

- `srt://FLUSSONIC-IP:SRT_PORT`

, где:

- `FLUSSONIC-IP` — IP-адрес сервера *Flussonic*.
- `SRT_PORT` — порт для проигрывания.

Следовательно, применительно к нашему примеру ссылка будет следующая: `srt://localhost:9998`. Таким образом, с помощью настройки `srt_play` вы разрешаете проигрывать SRT-поток по указанному порту.

Вы также можете определить SRT-порт для конкретного потока и для публикации (подробнее о публикации SRT-потоков во *Flussonic*, см. Публикация по SRT). Чтобы настроить один порт одновременно для публикации и проигрывания потока, используйте опцию `srt PORT_NUMBER`:

```
stream example_stream {
  input publish://;
  srt 9998;
}
```

В примере выше мы можем публиковать SRT-поток `example_stream` по порту 9998 и проиграть его по этому же самому порту, используя URL следующего формата:

- `srt://FLUSSONIC-IP:SRT_PORT?streamid=#!::m=request`

, где:

`streamid` — строка, сформированная как описано здесь.

- `m=request` — режим для проигрывания.
- `streamid` используется для того, чтобы указать режим `m=` для проигрывания потока `example_stream`, поскольку иначе неочевидно будем мы публиковать поток или проигрывать его.

Таким образом, URL для проигрывания потока `example_stream` будет выглядеть так: `srt://localhost:9`

Flussonic позволяет не только использовать один SRT-порт для одного потока, *разрешая проигрывание для этого потока*, но и настроить один порт для проигрывания нескольких потоков.

Один SRT-порт для проигрывания нескольких потоков

Для того, чтобы настроить SRT-порт для проигрывания нескольких потоков, используйте `srt_play` в качестве глобальной настройки:

```
srt_play {
  port 9998;
}
stream example_stream {
  input fake://fake;
}
```

Чтобы проиграть `example_stream` по SRT-порту 9998, используйте URL следующего вида:

- `srt://FLUSSONIC-IP:SRT_PORT?streamid=#!::r=STREAM_NAME`

, где:

- `streamid` — строка, сформированная как описано здесь.
- `r=STREAM_NAME` — имя потока.

Следовательно, для примера выше ссылка выглядит так: `srt://localhost:9998?streamid=#!::r=exa`

Рассмотрим еще один пример. Допустим, вам необходимо публиковать SRT-поток во *Flussonic* по отдельному определённому для публикации порту, а затем проигрывать этот поток наряду с несколькими другими. В таком случае конфигурация выглядит следующим образом:

```
srt_play {
  port 9998;
}
stream example_stream {
  input fake://fake;
}
stream another_stream {
  input publish://;
  srt_publish {
    port 8888;
  }
}
```

URL для проигрывания another_stream отличается от URL для example_stream и имеет следующий вид:

- srt://FLUSSONIC-IP:SRT_PORT?streamid=#!::r=STREAM_NAME,m=request

, где:

- m=request — режим для проигрывания.
- r=STREAM_NAME — имя потока.

Тогда URL для проигрывания another_stream выглядит так: srt://localhost:9998?streamid=#!::r

Для вашего удобства мы собрали всё в одну таблицу:

URL для проигрывания	
srt://FLUSSONIC-IP:SRT_PORT	
srt://FLUSSONIC-IP:SRT_PORT?streamid=#!::m=request	
srt://FLUSSONIC-IP:SRT_PORT?streamid=#!::r=STREAM_NAME	
srt://FLUSSONIC-IP:SRT_PORT?streamid=#!::r=STREAM_NAME,m=request	srt_play port 9988

Параметры для управления проигрыванием потоков по SRT

Flussonic также даёт возможность управлять проигрыванием потока с помощью параметров.

*

```
stream example_stream {
  input fake://fake;
  srt_play {
    passphrase 0987654321;
    port 9998;
```

```
  }
}
```

URL будет иметь следующий вид:

- `srt://FLUSSONIC-IP:9998?passphrase=0987654321&streamid=#!::m=request`

10.15 MSS

10.15.1 Воспроизведение MSS

Flussonic Media Server позволяет проигрывать видео по протоколу MSS.

Поток доступен по адресу: `http://FLUSSONIC-IP/STREAMNAME.isml/manifest`

Выбор дорожек для проигрывания

Указание дорожек необходимо для проигрывания потока на клиентских устройствах, которые, к примеру, не поддерживают многоязыковой MSS-манифест.

Если у потока есть несколько аудио- и видеодорожек, то можно указать, какие именно дорожки следует отдавать для проигрывания. Для этого укажите номера дорожек, добавив параметр 'filter.tracks' к URL потока.

Посмотрите примеры ниже:

- `http://FLUSSONIC-IP/STREAMNAME.isml/manifest?filter.tracks=v1a1` — выбрать первую аудио- и вторую видеодорожки.
- `http://FLUSSONIC-IP/STREAMNAME.isml/manifest?filter.tracks=a1` — только аудио.
- `http://FLUSSONIC-IP/STREAMNAME.isml/manifest?filter.tracks=v1` — только видео.
- `http://FLUSSONIC-IP/STREAMNAME.isml/manifest?filter.tracks=a1t2` — выбрать первую аудиодорожку и вторую дорожку с субтитрами.
- `http://FLUSSONIC-IP/STREAMNAME.isml/manifest?filter.tracks=v1t1t2t3` — первая видеодорожка и три дорожки с субтитрами.

Проигрывание DVR catch up по MSS

Вы можете запросить фрагмент архива как файл с помощью следующего URL-адреса:

`http://FLUSSONIC-IP:PORT/STREAM_NAME(archive=1651829645-120).isml/manifest`

См. справочник API.

Проигрывание событий по MSS

Доступ к архиву с настоящего момента (то есть, прямая трансляция) с возможностью перемотки назад до указанного времени (в нашем примере 1651829645):

`http://FLUSSONIC-IP:PORT/STREAM_NAME(archive=1651829645-now).isml/manifest`

См. справочник API.

Перемотка MSS

Плейлист со ”скользящим окном”, который позволяет перематывать и ставить на паузу потоки MSS на несколько часов:

```
http://FLUSSONIC-IP:PORT/STREAM_NAME(rewind=7200).isml/manifest
```

MSS с абсолютным таймшифтом

URL-адрес для проигрывания по MSS с абсолютным таймшифтом:

```
http://FLUSSONIC-IP:PORT/STREAM_NAME(timeshift_abs=1651829645).isml/manifest
```

MSS с относительным таймшифтом

URL-адрес для проигрывания по MSS с относительным таймшифтом:

```
http://FLUSSONIC-IP:PORT/STREAM_NAME(timeshift_rel=600).isml/manifest
```

10.16 MSE-LD

10.16.1 HTML5 (MSE-LD) воспроизведение с низкой задержкой

Долгое время Flash Player был лучшим и единственным способом доставлять видео на веб-страницу с относительно низкой задержкой. Низкая задержка требуется для вебинаров, спортивных мероприятий (ставок), видеонаблюдения, удаленного управления.

Сейчас запланировано постепенное удаление Flash из современных браузеров. Протокол WebRTC был добавлен в браузеры, но он имеет ограниченную поддержку аудио и видео кодеков (поддерживаются не все разновидности H.264, нет поддержки AAC).

Мы предлагаем новый способ решения этой проблемы с нашим плеером, который позволяет смотреть видео с действительно низкой задержкой с помощью встроенного в браузеры HTML5 и Media Source Extensions](https://en.wikipedia.org/wiki/Media_Source_Extensions) (MSE) механизма.

Воспроизведение с низкой задержкой

Проиграем поток с вашего Flussonic в браузере через механизм MSE. Откройте в браузере ссылку, заменив домен сервера и имя потока на свои:

```
http://flussonic-ip/STREAMNAME/embed.html?realtime=true
```

Если все в порядке (правильные кодеки, работающий поток, работающие вебсокеты), то вы получите видео с довольно низкой задержкой (около 1 секунды).

Детали

Мы используем MSE для доставки и проигрывания видео. Следовательно, поддерживаемые кодеки будут теми же, что и в браузере. Обычно H.264 и AAC поддерживаются, а остальное не считается пригодным.

Вам не нужно ничего, кроме HTTP или HTTPS, чтобы запустить MSE плеер, это может быть хорошим способом для проигрывания видео в условиях ограниченного окружения.

Вы также можете использовать наш плеер внутри своего приложения без использования iframe. Читайте о том, как встроить MSE JavaScript плеер в ваши приложения.

10.17 Codecs

10.17.1 Воспроизведение H265

H.265 (High Efficiency Video Compression — HEVC) — это формат сжатия видео, который постепенно приходит на смену H.264. Уменьшение размера файла по сравнению с H.264 может достигать 25-50%. При это качество изображения остается высоким. Поддерживается разрешение до 8K (UHDTV, 8192×4320 пикселей).

Вместо макроблоков, которые применялись в H.264, в HEVC используются блоки с древовидной структурой кодирования. Выигрыш кодера HEVC — в применении блоков большего размера. Применение таких блоков повышает эффективность кодирования при одновременном сокращении времени декодирования.

H.265 подается как решение для размеров экранов выше чем FullHD и поддерживается на большом количестве энкодеров: софтверных, обычных аппаратных (Nvidia NVENC, Intel QSV) и железных аппаратных. Также, H.265 можно встретить на спутниковом телевидении и IP камерах.

Поддержка H.265 в браузерах

> Десктопные браузеры с поддержкой H.265: Chrome (версия 107 и выше), Microsoft Edge (версия 16 и выше) и Safari (версия 11 и выше). Мобильные браузеры с поддержкой H.265: Chrome (версия 107 и выше) и Safari для iOS,(версия 11.0 и выше).

Десктопные браузеры:

| InternetExplorer | Microsoft Edge |MozillaFirefox | GoogleChrome | Safari | | — — — — — — — | — — — — — — | — — — — — — | — — — — — — | — — | | □ | □ | □ | □ | □ |

Мобильные браузеры:

| iOSSafari | OperaMini | ChromeforAndroid | UC BrowserforAndroid | SamsungInternet | | — — — — | — — — — — | — — — — — — — — | — — — — — — — — — — | — — — — — — | | □ | □ | □ | □ | □ |

На смартфонах H.265 скорее всего будет проигрываться на процессоре, таким образом сильно нагружая аккумулятор устройства.

Полезные ссылки: - https://caniuse.com/hevc - https://www.chromium.org/audio-video/ - debug: https://developer.chrome.com/docs/devtools/media-panel?hl=ru - debug: chrome://media-internals

HEVC в видеонаблюдении

HEVC уже давно используется там, где не требуется воспроизведение видео в браузере: - домашнее видеонаблюдение, там где абонент смотрит видео через отдельное приложение на своем смартфоне - диспетчерская, там где видео показывается с экрана NVR или отдельного плеера установленного на ПК сотрудника

Там же, где требуется доступ большого сотрудника

HEVC в телевидении

Построить популярный OTT или VOD сервисы кодируя только в HEVC не получится, т.к. у абонентов до сих пор много устройств, которые не поддерживают аппаратное декодирование HEVC.

Изяшное решение этой проблемы - кодирование сразу в несколько кодеков. HLS-плееры автоматически выберут поддерживаемый формат.

Поддержка H265 в протоколах

- В протоколе HLS формат H.265 поддерживается уже очень давно.
- В протоколе MPEG-TS формат H.265 поддерживается.
- В протоколе RTSP H.265 поддерживается. Есть упаковка и в SDP, и в RTP. Остается старый нюанс с передачей bframes по RTSP, но это отдельная головная боль.

Поддержка H265 в плеерах

Последние версии VLC Media Player имеют встроенную поддержку формата H.265.

H.265 также проигрывается на телевизионных приставках и SmartTV.

10.17.2 Воспроизведение AV1

AV1 (AOMedia Video 1) — это бесплатный кодек с открытым исходным кодом, созданный консорциумом технологических компаний Alliance for Open Media. Он разработан как замена HEVC/H.265. В сравнении с H.265/HEVC по результатам различных исследований выигрыш по битрейту составляет от 9,5 до 22%. По сравнению с H.264 кодек AV1 позволяет на 50% уменьшить битрейт при одинаковом качестве, и чем выше разрешение, тем эффект сжатия лучше.

Кодек AV1 основан на VP9, и для сжатия в нем применяется схожий способ блочного частотного преобразования, но предусмотрены дополнительные методы и фильтры для адаптивного улучшения качества картинки.

AV1 подходит для передачи видео высокого разрешения (4K, 8K) и поддерживается на большом количестве энкодеров, как программных, так и аппаратных (Nvidia NVENC, Intel Media SDK). В некоторых чипах для медиаплееров и телевизоров также поддерживается аппаратное декодирование AV1 в разрешении 4K или 8K. Аппаратная поддержка кодирования AV1 реализована в ряде видеокарт Nvidia, AMD и Intel.

Поддержка AV1 в браузерах

Десктопные браузеры с поддержкой AV1: Chrome (версия 70 и выше), Opera (версия 57 и выше), Mozilla Firefox (версия 67 и выше). В Microsoft Edge на Windows 10 требуется установка дополнительного расширения для использования AV1.

В большинстве мобильных браузеров есть поддержка AV1, но мобильные версии Opera и Safari не поддерживают воспроизведение AV1.

Десктопные браузеры:

| InternetExplorer | Microsoft Edge |MozillaFirefox | GoogleChrome | Safari | Opera | | — — — — — — — | — — — — — — — | — — — — — — | — — — — — — | — — | — — | | □ | с расширением для AV1 | □ | □ | □ | □ |

Мобильные браузеры:

| iOSSafari | OperaMini | ChromeforAndroid | UC BrowserforAndroid | SamsungInternet | | — — — — | — — — — — | — — — — — — — — — | — — — — — — — — — — | — — — — — — | | □ | □ | □ | □ | □ |

Поддержка AV1 в протоколах

- HLS поддерживает AV1 при использовании контейнера fMP4.
- WebRTC поддерживает AV1 в связке с аудиокодеком Opus.
- SRT поддерживает AV1.

Поддержка AV1 в плеерах

Кодек AV1 поддерживается большинством популярных плееров, в том числе VLC media player, K-Lite Codec Pack, Media Player Classic, MX Player и т.д.

10.18 Protection

10.18.1 Авторизация

Одним из способов обеспечения безопасности и защиты от злоумышленников является *авторизация*. ***Авторизация*** — это механизм предоставления пользователю определённых разрешений и прав на доступ к чему-либо или выполнение определённых действий.

Во *Flussonic Media Server* реализован механизм идентификации пользователей и отслеживания подключений с помощью ***авторизационных бэкендов***. Авторизационный бэкенд определяет правила авторизации, чтобы разрешать или отклонять запросы. Роль авторизационного бэкенда может играть внешняя система (например, ваш сайт), скрипт на диске, часть конфигурации Flussonic или IPTV плагин — в любом случае он определяет правила авторизации.

Основная идея авторизации во Flussonic заключается в следующем: ваш сайт опознает клиента, который собирается играть видео, (например, по cookie-файлам) и добавляет к URL-адресу Flussonic в плеере уникальный ключ. Плеер запрашивает у Flussonic видео с этим ключом. Flussonic отправляет запрос авторизационному бэкенду (например, обратно вашему сайту) и уточняет, можно ли этому плееру играть видео с этим ключом. Если авторизационный бэкенд разрешил проигрывание, то клиент получает доступ к видео.

Чтобы наглядно увидеть, как *Flussonic Media Server* взаимодействует с авторизационным бэкендом, посмотрите видео ниже:

Подробнее о работе *Flussonic Media Server* с бэкендом читайте в разделе Авторизация через бэкенд.

Во *Flussonic* по протоколу HLS используются HTTP-механизмы отслеживания сессий, а по протоколам RTMP, RTSP и MPEG-TS — обрабатываются постоянные TCP-сессии. Также отслеживается экспорт архива в формате MPEG-TS и MP4.

Кроме того, *Flussonic Media Server* имеет встроенные механизмы базовой защиты от вставки плеера на других сайтах. Более подробно про такую защиту вы можете прочесть в разделах Domain lock и CORS для защиты плеера.

Flussonic Media Server также может проверять пароль при публикации потока. Подробнее см. раздел Защита публикации паролем.

Авторизация через бэкенд

Авторизационный бэкенд устанавливает правила, по которым запросы одобряются либо отклоняются. *Flussonic Media Server* поддерживает несколько авторизационных бэкендов.

Flussonic позволяет настроить *внешнюю* или *внутреннюю авторизацию* через бэкенд:

***Внешняя авторизация** используется в случае, если вы предпочитаете, чтобы внешняя система утверждала или отклоняла запросы. В этом случае* Flussonic *не знает** кому можно/нельзя давать доступ к потоку. Алгоритм действий следующий:

1) Пользователь запрашивает доступ к потоку у *Flussonic*. 2) *Flussonic* обращается к авторизационному бэкенду. 3) Бэкенд проверяет можно ли этому пользователю дать доступ к потоку и возвращает

соответствующий ответ. 4) *Flussonic* разрешает или запрещает пользователю доступ.

***Внутренняя авторизация** *используется в случае, если вы предпочитаете, чтобы* Flussonic Media Server *утверждал или отклонял запросы. Теперь* Flussonic *знает** кому можно/нельзя давать доступ к потоку. Подробнее см. Конфигуратор бэкендов.

Во *Flussonic* реализованы два типа авторизации с помощью бэкенда:

- Авторизация сессий проигрывания (`on_play`)

Ограничивает доступ к проигрыванию неавторизованным пользователям. См. Авторизация сессий проигрывания.

- Авторизация сессий публикации (`on_publish`)

Ограничивает доступ к публикации неавторизованным пользователям. См. Авторизация сессий публикации.

Описание процедуры авторизации *Этап 1.*

Вы размещаете Flash-плеер или HTML-тег `<video>` на веб-сайте или Middleware, указывая в пути к видео ключ авторизации (`token`), созданный веб-сайтом, одним из следующих способов:

- В виде строки запроса (*query string*) для HLS, HTTP MPEG-TS и других доступов по HTTP:

`http://192.168.2.3:80/stream1/index.m3u8?token=60334b207baa` для HLS

* В виде адреса для RTMP: `rtmp application rtmp://192.168.2.3/static stream name: strea`
* В виде адреса для RTSP: `rtsp://192.168.2.3/stream1?token=60334b207baa`

Если веб-сайт или Middleware не указывает ключ `token` в пути к видео, то *Flussonic Media Server* генерирует его автоматически.

Этап 2.

Получив запрос к потоку с ключом `token`, сервер *Flussonic Media Server* проверяет, открыта ли сессия (транслируется ли уже поток с сервера на этот клиент) с помощью *идентификатора сессии*. ***Идентификатором сессии**хеш-сумма, создаваемая для имени потока (`name`), IP-адреса клиента (`ip`) и токена (`token`) следующим образом*:

hash(name + ip + token)

Если пользователь меняет свой IP-адрес или переключается на другой поток, то создается новая сессия.

Этап 3.

Если открытых сессий еще нет, *Flussonic Media Server* отправляет запрос в авторизационный бэкенд. Полный список параметров, которые отправляются в запросе, можно найти в схеме Authorization Backend API.

Этап 4.

Бэкенд возвращает ответ, в котором содержится следующее:

- Информация о сессии: в течение какого периода времени сессия активна, сколько открытых сессий разрешено для этого пользователя.
- Конфигурация вставки рекламных видео клипов в проигрываемый поток. См. Вставка рекламы.

Полный список параметров, которые возвращаются в ответ, можно найти в схеме Authorization Backend API.

Включение бэкенда Настройка бэкенда производится путём добавления директивы `on_play` в конфигурационный файл:

```
on_play PATH_TO_AUTH;
```

, где в качестве **PATH_TO_AUTH** можно указать:

- **сетевой адрес HTTP**

Если в качестве бэкенда указан сетевой адрес HTTP, то *Flussonic Media Server* будет делать HTTP-запросы по этому адресу, передавая параметры сессии бэкенду.

- **путь на диске**

Если в качестве бэкенда указан путь на диске, то он интерпретируется как путь к скрипту, который будет выступать в роли бэкенда.

Как можно сохранить сессию при изменении IP адреса клиента? Каждая сессия характеризуется своим идентификатором (`session_id`), который зависит от токена. Если токен меняется, то идентификатор сессии меняется соответственно. Но что если нам нужно сохранить сессию и пропустить повторную авторизацию при смене адреса клиента, например на переходе между сотами? Это можно сделать с помощью ключей сессии, которые используются для генерации идентификатора сессии.

Можно использовать сделующие ключи сессии:

Ключ	Описание
proto	протокол
name	имя потока
ip	IP-адрес
token	токен

Идентификатор сессии рассчитывается по следующей формуле (хеш-сумма):

hash(name + ip + proto + token)

Таким образом, сессия будет "сброшена", если поменяется любой из ключей (необязательно токен).

Чтобы указать ключи сессии, перейдите в раздел **Media** > нажмите имя потока > **Auth** > выберите ключи сессии из выпадающего списка **Select session keys**.

Вы также можете указать параметр `session_keys` в настройках `on_play` URL, в конфигурации потока.

На список элементов `session_keys` накладываются следующие ограничения:

- ключи `ip`, `name` и `proto` **обязательны** и указываются **явно** (порядок не имеет значения);
- список может содержать **дубликаты**, которые **обрабатываются явно**, что повлияет на конечный результат;
- элементы списка перечисляются через запятую `,` без пробелов;
- если значение ключа **не** найдено, в хеш будет включён ключ со значением `undefined`.

Пример конфигурации:

```
stream example_stream {
  input fake://fake;
  on_play http://IP-ADDRESS:PORT/php-auth-script.php session_keys=ip,name,
   proto,token;
}
```

В примере выше `ip`, `name`, `proto` и `token` используются для вычисления ID сессии (`session_id`).

Сессия открыта Если бэкенд разрешил открытие сессии, то, по умолчанию, *Flussonic Media Server* будет перепроверять статус сессии **раз в 3 минуты**, чтобы определить, что сессия всё ещё активна.

Чтобы изменить это время, отправьте новое значение в HTTP-заголовоке `X-AuthDuration` (указывается в секундах).

Через 3 минуты (или другой промежуток времени, если он был изменен с помощью `X-AuthDuration`) запрос к сессии приведёт к повторному обращению к бэкенду.

Figure 10.34: Session keys

Если бэкенд недоступен или возвращает `статус 500`, то *Flussonic Media Server* сохранит предыдущий статус, полученный от бэкенда, и попробует ещё раз обратиться к нему.

Сессия закрыта Если бэкенд запретил открытие сессии, то информация о ней кешируется на сервере. В случае, если пользователь пытается ещё раз с тем же токеном получить доступ к потоку, *Flussonic Media Server* будет отказывать, не делая повторных обращений к бэкенду.

Просмотр видео в веб-интерфейсе Администратор может просматривать любое видео в веб-интерфейсе *Flussonic* без авторизации, т. е. обращений к бэкенду авторизации *не* производится.

Технически это реализовано следующим образом: при просмотре из веб-интерфейса генерируется

специальный токен ADM-xxx, который перехватывается *Flussonic Media Server*. Такой токен воспринимается как разрешение воспроизводить видео без авторизации.

Можно запретить Администратору просматривать видео, защищенное при помощи механизма бэкенд-авторизации.

Простейший пример скрипта авторизации (PHP)

Будем хранить авторизацию в файле `auth.txt`, заполненном такими данными:

```
user1:token1
user2:token2
user3:token3
```

Следующий скрипт на PHP будет проверять, содержится ли токен в этом файле. Если ответ положительный, то разрешать открытие сессии:

```
<?php

$get = print_r($_GET, true);
$token = $_GET["token"];
if(!$token || !strlen($token)) {
    header('HTTP/1.0 403 Forbidden');
    error_log("No token provided", 4);
    die();
}

$tokens = array();
$contents = explode("\n", file_get_contents("auth.txt"));
foreach($contents as $line) {
    if(strlen($line) > 3) {
        $parts = explode(":", $line);
        $tokens[$parts[1]] = $parts[0];
    }
}

if($tokens[$token]) {
    header("HTTP/1.0 200 OK");
    header("X-UserId: ".$tokens[$token]."\r\n");
    header("X-Max-Sessions: 1\r\n"); // Turn this on to protect from
   multiscreen
} else {
    header('HTTP/1.0 403 Forbidden');
}
?>
```

Сбор статистики с помощью X-UserId

Бэкенд при открытии сессии может отправить *Flussonic Media Server* HTTP-заголовок `X-UserId` (к примеру, `X-UserId: 100`), который после закрытия сессии будет записан во внутреннюю базу данных вместе с данными о сессии. Вы можете запрашивать данные о сессии по протоколу MySQL с указанием `X-UserId` для сбора статистики.

Если бэкенд отправляет заголовок `X-Unique: true` вместе с `X-UserId`, то происходит отключение всех остальных открытых сессий, которые имеют такой же `X-UserId`.

> Отключенные сессии на некоторое время остаются в памяти сервера. Клиенты с теми же сочетаниями IP-адреса, имени потока и токена **не** смогут получить доступ к контенту.

При использовании опции `X-Unique` следует генерировать различные токены при каждом обращении пользователя к странице.

Таймаут авторизационного бекенда

В случае, если авторизационный бекенд не успевает ответить за 3 секунды, возможны следующие исходы:

| **Состояние сессии** | **Исход** | | ——————— | ——————— | | Не открыта | Не открывается. | | Разрешена | Остается открытой. | | Запрещена | Остается запрещенной. |

Авторизация сессий проигрывания и публикации

Flussonic позволяет настроить авторизацию сессий проигрывания и публикации потоков для сессии (*per session*).

Авторизация сессий проигрывания Перед проигрыванием потока *Flussonic* должен убедиться, что у пользователя есть права доступа для просмотра контента.

Когда клиент запрашивает поток для проигрывания, *Flussonic* отправляет запрос авторизационному бэкэнду. Если поток запущен и у клиента есть разрешение на его просмотр, авторизационный бэкэнд возвращает HTTP 200 и *Flussonic* отправляет клиенту URL потока.

Чтобы настроить авторизацию для сессии проигрывания, используйте опцию `on_play` в рамках шаблона конфигурации `template` или потока `stream`:

```
template on-play-example {
  on_play http://IP-ADDRESS:PORT/PATH_TO_SCRIPT;
}
```

Авторизация сессий публикации *Flussonic* позволяет вам настроить авторизацию для публикаторов с помощью стороннего программного обеспечения, чтобы избежать ненадёжных источников и предоставлять права на публикацию только проверенным публикаторам. Публиковать контент могут только те пользователи, у которых есть на это разрешение. Как только они устанавливают соединение для начала сессии, *Flussonic* проверяет, есть ли у пользователя разрешение на публикацию.

Это работает следующим образом: пользователь устанавливает соединение с *Flussonic Media Server* и запрашивает разрешение на публикацию. Перед началом публикации *Flussonic* отправляет POST-запрос авторизационному бэкенду. JSON-тело этого объекта содержит поля, описанные в схеме API. По сути это параметры, идентифицирующие сессию и позволяющие проверить, есть ли у пользователя разрешение на публикацию контента. *Flussonic*, основываясь на ответе от авторизационного бэкенда (`HTTP 200` или `HTTP 403`), разрешает либо запрещает пользователю публиковать поток.

Чтобы настроить авторизацию для сессии публикации, используйте опцию `on_publish` в настройках шаблона конфигурации `template` или потока `stream`:

```
template on-publish-example {
  prefix on-publish-example;
  input publish://;
  on_publish http://IP-ADDRESS:PORT/on_publish.json;
}
```

10.18.2 Конфигуратор бэкендов

Flussonic позволяет создавать авторизационные бэкенды в основном файле конфигурации.

Можно указать белые и черные списки для IP-адресов, токенов, User-Agents, стран, несколько параллельных HTTP-бэкендов.

Настройка авторизации

Добавьте в файл `/etc/flussonic/flussonic.conf`:

```
auth_backend myauth1 {
  allow ip 127.0.0.1;
  allow ip 192.168.0.1;
  allow ip 172.16/24;
  deny ip 8.8.8.8;
  allow country RU US;
  deny country GB;
  allow token test_token1;
  deny ua "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.10)";
  backend http://stalker-1.iptv.net/auth.php;
  backend http://stalker-2.iptv.net/auth.php;
}
```

- `allow` — объявляет "белый" список, т.е. немедленно разрешает просмотр без дальнейших проверок.
- `deny` — объявляет черный список, запрещает просмотр.

Flussonic проверяет правила в следующем порядке:

- allow token
- deny token
- allow ip
- deny ip
- allow country (страна)
- deny country (страна)
- allow ua (useragent)
- deny ua (useragent)
- Опрашивает несколько параллельных бэкендов

10. Запрещает доступ, если не указано ‘allow default‘.

Приоритет правил важен. Правила с более высоким приоритетом применяются немедленно, и тогда правила с более низким уже не учитываются. Например, если вы разрешили IP-адрес, но клиентское приложение или устройство приходит с запрещенным токеном, доступ будет запрещен, т.к. у токена приоритет выше.

Чтобы использовать этот авторизационный бэкенд для потока, укажите `auth://myauth1`:

```
stream example_stream {
  input udp://239.255.0.1:1234;
  on_play auth://myauth1;
}
```

Правила срабатывают после перезагрузки конфигурации.

Опция allow default

Опция `allow default` позволяет определить поведение в случае, когда ни один авторизационный бэкенд не отвечает (например, в случае получения ошибки в HTTP-ответе или неработоспособности скрипта). Если эта опция включена, всем клиенатам или устройствам, за исключнием явно перечисленных в опции `deny`, будет предоставлен доступ к содержимому. Если же эта опция отключена, то всем клиенатам или устройствам, за исключнием явно перечисленных в опции `allow`, доступ к содержимому будет запрещен.

.

Рассмотрим, как *Flussonic* поступает с разными ответами от бэкенда и как включенная опция `allow default` влияет на решение о предоставлении доступа к видеопотоку.

Опция allow default в случае одного бэкенда Если авторизационный бэкенд запрещает доступ (отвечает с кодом ошибки 4xx, например, `403 Forbidden`), *Flussonic* не разрешит доступ к содержимому, даже если в конфигурации указано `allow default`.

Однако если бэкенд не отвечает (не работает по причине ошибки) или произошла ошибка сервера, на котором работает бэкенд-скрипт (с кодом ошибки 5xx, например, `500 Internal Server Erro`
Flussonic разрешает доступ к содержимому всем клиентам или устройствам, кроме перечисленных в опции `deny`.

Опция allow default в случае нескольких бэкендов Если у вас несколько параллельных бэкендов, правила примерно те же.

Если хотя бы один из бэкендов разрешает доступ, а остальные бэкенды запрещают его или не отвечают, то доступ будет разрешен.

Если хотя бы один из бэкендов запрещает доступ, а остальные не отвечают, т.е. ни один не разрешает, доступ будет запрещен.

Однако если *все* бэкенды не работают (не отвечают), *Flussonic* разрешает доступ к содержимому всем клиентам, кроме перечисленных в опции deny.

В таблице показана логика авторизации при использовании нескольких бэкендов для потока:

Backend 1	Backend 2	Backend 3	Результат
allow	allow	allow	Allow
ban	ban	ban	Ban
ban	allow	ban	Allow
not responding	not responding	not responding	Allow
not responding	allow	not responding	Allow
not responding	ban	not responding	Ban

Примеры

Мультиавторизация и доступ из локальной сети

```
auth_backend multi_local {
  allow ip 192.168.0/24;
  backend iptv://localhost; # iptv plugin
  backend http://examplehost/stalker_portal/server/api/
    chk_flussonic_tmp_link.php;
}
```

Заблокировать несколько пользователей, остальным разрешить

```
auth_backend blacklist {
  deny ip 1.1.1.1;
  deny ip 2.2.2.2;
  deny ip 10.10/16;
  allow default;
}
```

Использовать HTTP бэкенд и разрешить просмотр видео клиентам с указанными токенами

```
auth_backend myauth2 {
  allow token friend_token1;
  allow token friend_token2;
  backend http://examplehost/stalker_portal/server/api/
    chk_flussonic_tmp_link.php;
}
```

Разрешить только свои приставки по User-Agent, остальных блокировать

```
auth_backend agents {
  allow ua MAG;
  allow ua TVIP;
}
```

10.18.3 Middleware в IPTV OTT

В старом аналоговом эфирном телевидении пользователь должен был настраивать самостоятельно все нужные каналы: первый канал на 1-ю кнопку, второй канал на 2-ю. Пока каналов меньше 10 такой вариант всех устраивал. Никакого контроля доступа или детализированного учета: что чего смотрит в древнем телевидении не было.

С переходом на IPTV технология настройки не сильно отличалась: в приставку при заливке прошивки добавляется плейлист, т.е. список каналов с их multicast UDP адресами. Контроль доступа осуществляется либо с помощью шифрования (CAS), либо с помощью сетевых методов, например авторизация IGMP запросов, которая возможна только в простой локальной сети.

При этом все функции реализуются на приставке. Например, такая услуга как PVR (personal video recorder) осуществляется на приставке: пользователь заранее заказывает запись передачи и в нужный момент приставка сама начинает на свой жесткий диск записывать нужную передачу.

С развитием IPTV пользователям начали предлагать такие новые сервисы, как:

- EPG (electronic program guide), т.е. расписание передач.
- Организация каналов по платным пакетам.
- Каталогизация каналов (по жанрам).
- Родительский контроль (не показывать детям эротику).
- Просмотр записанных передач.
- Сопутствующие сервисы типа прогноза погоды или курса валют.
- Подписка на платный пакет с пульта.
- Предоставление VOD, т.е. просмотр фильмов.

Важно понимать, что большинство решений в сфере IPTV разрабатывалось при постоянных мыслях о том, как бы это реализовать в условиях спутникового вещания, т.е. когда связи у приставки с каким-то центральным сервером нет. Из-за этого в транспортных протоколах, используемых в телевидении есть очень много деталей бизнес-логики, которые были нужны раньше, но становятся не очень актуальны сегодня.

С развитием разнообразия услуг в IPTV возникло понимание, что развивать услуги за счёт усложнения приставки не очень удобно или не всегда возможно, потому что когда программная часть услуги реализована на приставке, её обновление и поддержка требуют небезопасной процедуры обновления прошивки на приставке.

Для упрощения ввода в строй новых услуг и управления ими, а так же для реализации услуг, невозможных в классическом телевидении, в IPTV появилось звено под названием Middleware.

За этим страшным термином прячется обычный веб-сайт (отдающий HTML, javascript или отвечающий на HTTP API запросы), на который приставки заходят либо обычным веб-браузером, либо чем-то экзотическим типа SVG браузера. На приставке размещается доработанный веб-браузер

(обычно Opera или webkit), который умеет проигрывать все варианты видео, доступных приставке (десктопные браузеры обычно не умеют и 5% от видео-возможностей приставки) и умеет работать с пультом, превращая нажатия кнопок пульта в Javascript-события.

Важный момент: на приставках не используется Java (за исключением андроидных приставок, которые по ряду причин очень непопулярны). Обычно люди путают Java и Javascript, не надо так делать.

До сих пор существует дискуссия среди специалистов, что лучше: веб-браузер или специализированное приложение на устройстве. Этот выбор совершенно аналогичен выбору технологии на мобильных устройствах: делать HTML приложение или писать на C.

Многие современные middleware предлагают оба варианта для покрытия максимального количества устройств. Так, например, для приставок Amino (с браузером opera), Mag250, tvip и т.п. (с браузером на базе webkit) будет отдаваться HTML с интерфейсом.

Обычно middleware практически не взаимодействует с видеопотоком, потому что лишь предоставляет приставкам ссылки для просмотра каналов и фильмов: либо мультикаст url, либо уникаст. Иногда в Middleware реализован механизм мониторинга каналов, что бы не показывать клиентам канал или явно сообщать, что просмотр канала невозможен из-за аварии.

Ниже мы рассмотрим, каким образом происходит интеграция Flussonic Media Server и Middleware для улучшения качества сервисов пользователям.

Авторизация пользователей

В IPTV ограничение доступа к видео используется для того, чтобы:

- Управлять пакетами каналов. Не подписался на футбол — смотри что попало.
- Усложнять задачу воровства контента неавторизованными пользователями.
- Усложнять задачу несанкционированной записи передач.

Тут смешиваются две системы: CAS и DRM. CAS (conditional access system) — это механизм технического ограничения доступа к контенту. DRM — механизм для запутывания пользователя, что бы он никак не добрался до расшифрованного видео.

CAS системы работают хорошо и исправно. DRM системы в силу своей природы ненадежны, глючат и создают массу проблем всем, кроме их продавцов.

В Flussonic Media Server реализована CAS с помощью авторизации доступа к потокам и файлам.

Интеграция с Middleware выглядит следующим образом:

- Middleware при формировании HTML страницы или ответа на API, отдает ссылку на HLS (или HTTP MPEG-TS) поток с уникальным ключом.

- Приставка получает этот уникальный url для просмотра, обращается с ним к Flussonic Media Server.
- Flussonic Media Server при первом запросе обращается обратно к Middleware с вопросом: можно ли этой приставке смотреть этот канал по этому адресу.
- Middleware проверяет, не подсмотрен ли этот адрес (проверяется IP клиента, user agent, протокол и время) и разрешает или запрещает.

Если злоумышленник подсмотрит в сети адрес, то он им не сможет воспользоваться, потому что не совпадет какой-нибудь из параметров и Middleware скажет стримеру не отдавать ему видео.

Работа с EPG

EPG, он же электронный телегид, оно же расписание передач. Обычно представляет из себя источник головной боли, если хочется точно попадать в передачу, потому что практически никто на российском рынке не предоставляет точного расписания передач.

Телеканалы не особо следят за точностью программы передач (погрешность до 15-20 минут) и постфактум точное время начала и конца передач не сообщают.

EPG можно получать со спутника в MPEG-TS потоке, но там информация достаточно ограничена, а можно забирать из интернета у поставщиков программы передач, таких как teleguide.info

Частый формат для программы передач — XMLTV.

Традиционно приставки повторяют функциональность древних видеомагнитофонов: пользователь заранее заказывает желаемую передачу и приставка записывает её. И так же традиционно из-за ошибок в расписании передач, пользователь записывает хвост предыдущей рекламы, 20 минут рекламы, потом нужную ему передачу, обрезанную в самом конце, потому что расписание поехало.

Flussonic Media Server предлагает совершенно другой подход к записи передач. Не нужно издеваться над пользователем и заставлять его заранее вспоминать о нужной передаче. Middleware должна предоставлять возможность пользователю посмотреть передачи, которые уже прошли и сформировать правильную ссылку к Flussonic Media Server для просмотра уже записанной передачи.

Тут есть два механизма работы: для уже прошедшей передачи и для ещё идущей.

Если передача уже прошла и закончилась, то Middleware на основании EPG формирует ссылку для просмотра из архива (которая так же проходит через механизм авторизации). Пользователь получает возможность посмотреть записанную передачу, как обычный файл.

Например, если передача началась в 18:15 по Москве (14:15 UTC) 27 августа и длилась час, то Middleware должен при выборе передачи в списке прошедших сформировать URL вида http://streamer/ort/index-1409148900-3600.m3u8

Если передача сейчас всё ещё идет, то Middleware может сформировать специальный url к архиву, позволяющий отматывать назад прямой эфир на начало передачи. Данная функциональность, к сожалению, поддерживается далеко не на всех устройствах и приставках, но тем не менее она существует.

URL для такой незакончившейся передачи будет выглядеть http://streamer/ort/index-1409148900-now.m3u8

Важный момент здесь в том, что информация о записанных передачах и об их времени хранится в middleware, а Flussonic предоставляет доступ к своему архиву как к бесконечной ленте.

В документации более подробно описана работа с архивом DVR

В Flussonic Media Server есть поддержка и других вариантов доступа к записанным передачам:

- проигрывание архива по HTTP MPEG-TS с определенного момента: http://streamer/ort/timeshift_abs-1409148900.ts
- проигрывание архива в режиме потока по HLS с определенного момента: http://streamer/ort/timeshift_ab 1409148900.m3u8

Так же Middleware может обратиться к Flussonic Media Server к API о состоянии записи потока, чтобы показать в интерфейсе передачи, которые можно посмотреть и которые нельзя записать.

Таймшифт

Под термином таймшифт подразумевают две разных функции: возможность отмотки назад прямого эфира и постоянный сдвиг эфира в другую временную зону.

Эта услуга нужна, когда видео захватывается в одном часовом поясе, а показывать хочется в другом для того, что бы, например, пользователи в США видели утреннюю передачу в 9 утра, а не в час ночи.

Flussonic предлагает два варианта таймшифта: запуск постоянного потока, который отстает на фиксированное время от эфира и предоставление ссылок на просмотр архива в режиме потока.

Разница между ними в количестве чтения с диска. Если к каналу редко обращаются, логичнее воспользоваться вторым вариантом. Если канал смотрят со сдвигом часто, то нужно запускать постоянный поток.

Задача Middleware здесь — знать как настроен канал и выдавать ссылки либо к смещенному каналу, либо индивидуальные ссылки на просмотр архива вида:

- http://streamer/ort/timeshift_rel/7200 — проигрывание архива по HTTP MPEG-TS с отставанием в 2 часа
- http://streamer/ort/timeshift_rel-7200.m3u8 — проигрывание архива по HLS с отставанием в 2 часа

Интеграция Flussonic с Middleware

На сегодняшний день поддержка возможностей Flussonic Media Server есть в следующих Middleware:

- iptvportal (они же распространяют Flussonic Media Server в составе пакета)
- Stalker
- CloudWare
- Telebreeze

С другими Middleware можно провести интеграцию по вашему запросу.

На стороне Flussonic Media Server реализованы всё, что нужно для интеграции с Middleware. Просите вашего поставщика Middleware проверить совместимость с Flussonic Media Server самостоятельно. Также вы можете протестировать следующие Middleware:

- www.magoware.tv
- www.abvtc.com

10.18.4 Защита контента с помощью DRM

DRM (Digital Rights (Restrictions) Management) — это способ защиты видео-контента при помощи шифрования парой ключей. Ключи выдаются *сервером ключей* DRM-системы.

Ниже на этой странице приведены настройки, одинаковые для всех поддерживаемых DRM систем.

По ссылкам приведены настройки для конкретных DRM. *Flussonic Media Server* поддерживает работу со следующими системами DRM:

- Axinom
- EzDRM
- BuyDRM (KeyOS)
- drmnow!
- DRMtoday
- Widevine
- PlayReady DRM

Описание механизма защиты DRM

Apple в спецификации протокола HLS описывает два штатных механизма шифрования: AES-128 и SAMPLE-AES. *Flussonic Media Server* поддерживает оба.

Механизмы отличаются лишь непосредственным способом шифрования данных и работают по трёхсторонней схеме. Вот как это реализовано во *Flussonic*:

- *Flussonic* запрашивает ключ для шифрования контента от сервера ключей и получает его вместе с URL этого ключа.
- Клиент получает от *Flussonic* шифрованный контент и URL ключа для дешифрования.
- Сервер ключей получает от клиента запрос на ключ для дешифрования и решает: отдавать ключ или нет.

Если клиент получает контент от *Flussonic* по безопасному соединению и общается с сервером ключей по протоколу HTTPS, то можно надеяться на то, что он сможет расшифровать видео и проиграть его, не открыв доступ к нешифрованному контенту неавторизованным пользователям.

Механизмы получения ключа для видеопотоков и для файлов не отличаются.

Настройка шифрования в общем случае

Flussonic Media Server хранит контент в нешифрованном виде. Контент шифруется при отдаче клиенту.

Чтобы включить шифрование, добавьте параметр `drm` в настройки потока или зоны файлов в файле конфигурации (`/etc/flussonic/flussonic.conf`). Затем укажите метод шифрования и сервер ключей (в зависимости от выбранной системы DRM могут потребоваться и другие данные):

```
stream channel0 {
  input fake://fake;
  drm aes128 keyserver=http://examplehost:5000/cas-server;
}
```

Вы также можете осуществить настройку шифрования в веб-интерфейсе *Flussonic UI*. Для этого:

- Перейдите на вкладку **Media** в раздел **Streams** и выберете необходимый поток. После чего кликните на его название.
- В открывшемся окне настроек потока, перейдите на вкладку **Auth** и найдите раздел **Require DRM authorization**. Выберете желаемую систему DRM из списка доступных и укажите необходимую информацию:

Примеры настроек для отдельных DRM-систем приведены в одноименных разделах (см. ссылки в начале этой странице). Актуальный список настроек для каждой DRM-системы вы можете найти в 7Cbody%7Cdrmtag/stream/operation/stream-save%7Cbody%7CdrmFlussonic API reference.

После сохранения настроек *Flussonic* будет шифровать контент для всех протоколов, которые могут работать с указанным методом шифрования.

Вы также можете редактировать настройки DRM с помощью API, передавая 7Cbody%7Cdrmtag/stream/operatio save%7Cbody%7Cdrmобъект ‘drm‘ в запросе `Flussonic-API: PUT /streams/{name}`.

Вы также увидите следующее предупреждение на вкладке **Overview** в настройках потока:

Опции для проигрывания по HLS Для успешного проигрывания AES128-шифрованного потока по HLS на некоторых современных устройствах (работающих на Tizen 5) необходимо добавить опцию `hls_ext_x_key_iv=false`:

```
stream channel0 {
  input fake://fake;
  drm aes128 keyserver=http://examplehost:5000/cas-server hls_ext_x_key_iv=
   false;
}
```

Figure 10.35: Flussonic drm

Необходимо отключить все протоколы, которые несовмести
системой. Если указанный метод шифрования поддерживаетс
вас остался работающим протокол DASH, то пользователь сможе
видео по DASH без шифрования.

Запретить проигрывание по протоколам

Чтобы этого не произошло, надо отключить все лишние протоколы для нужного потока или зоны файлов:

```
stream channel0 {
  input udp://239.0.0.1:1234;
  protocols hls;
  drm aes128 keyserver=http://examplehost:5000/cas-server;
}
```

Figure 10.36: DRM alert

```
vod vod_files {
  storage /storage;
  protocols hls;
  drm aes128 keyserver=http://examplehost:5000/cas-server;
}
```

Теперь доступ к контенту доступен только по протоколу HLS.

DRM для файлов

В случае с файлами, внешний сервер ключей не может явно указывать ключ, потому что не знает, когда файл будет открыт на чтение.

Необходимо сконфигурировать файл на явное обращение к серверу ключей:

```
vod vod_files {
  storage /storage;
  protocols dash hls;
  drm aes128 keyserver=http://examplehost:5000/cas-server;
}
```

При такой конфигурации Flussonic будет делать HTTP GET-запросы к серверу ключей с параметром ?name=: `http://192.168.0.80:4500/?name=drm/bunny.mp4&number=1`

Flussonic ожидает ответ, в котором первые 32 байта будут HEX-представлением ключа, используемого для шифрования. Также в этом ответе ожидается заголовок `X-Key-Url`, который будет сообщён клиенту. По этой причине `X-Key-Url` клиент будет ожидать 16-байтный ключ (не в HEX виде) для дешифровки.

Защита DVR архива с помощью DRM

Архив шифруется посегментно одним ключом, и каждые 10 минут *Flussonic* запрашивает новый ключ для следующей группы сегментов.

> Чтобы DRM защита DVR архива работала, сервер ключей должен хранить старые ключи (по старым URL) по продолжительности не менее, чем глубина архива.

Включение шифрования всех кадров

По умолчанию *Flussonic* шифрует только ключевые кадры. Как правило, этого достаточно для защиты от несанкционированного доступа к потоку, однако некоторые Smart TV и приставки требуют, чтобы шифрованию подвергались все кадры в потоке. К тому же шифрованию ключевых кадров снижает энергопотребление при дешифровке на стороне клиента.

Чтобы включить шифрование всех кадров, используйте опцию `encryption=full` при настройке DRM.

Для включения режима шифрования только ключевых кадров используйте опцию `encryption=sparse`.

```
stream channel0 {
  url fake://fake;
  protocols dash hls;
  drm aes128 keyserver=http://examplehost:5000/cas-server encryption=full;
}
```

Ротация ключей шифрования

Многие серверы DRM периодически меняют (ротируют) ключи с целью более надежной защиты. По умолчанию *Flussonic* не ротирует ключи шифрования. Для включения ротации и изменения интервала ротации ключей используйте опцию `expires` и укажите требуемое время в минутах:

```
stream channel1 {
  url fake://fake;
  protocols dash hls;
  drm aes128 keyserver=http://examplehost:5000/cas-server expires=60;
}
```

В случае использования опции `expires` параметр `drm_id` генерируется автоматически при каждом новом запросе ключа шифрования.

> Включение опции `expires` означает регулярное получение новых ключей шифрования от сервера ключей. В зависимости от условий провайдер DRM может взимать оплату за каждый выданный ключ. Рекомендуем перед включением опции `expires` проверить свой договор с провайдером DRM.

CPIX API

CPIX — это открытая спецификация, разработанная DASH-IF, которая предоставляет совместимый формат на основе XML для обмена конфигурациями защиты контента между различными системами.

Flussonic Media Server поддерживает CPIX API. Используя этот API, любой поставщик DRM может интегрироваться с *Flussonic*, если в его системе поддерживается CPIX API.

CPIX использует некоторую XML схему для описания взаимодействия между сервером ключей и *Flussonic*.

Добавление защиты CPIX DRM Чтобы настроить DRM с обменом ключами в формате CPIX, укажите ваш сервер ключей с помощью опции `drm cpix`.

Для потока:

```
stream mystream {
  input udp://239.0.0.1:1234;
  protocols dash hls mss;
  drm cpix keyserver=http://my.keyserver resource_id MYSTREAM;
}
```

Для файла:

```
vod drm {
  storage /storage/vod;
  protocols dash hls mss;
```

```
  drm cpix keyserver=http://my.keyserver;
}
```

Обратите внимание, что при использовании конфигурации для файла нужно поместить `drm_id` в файл `*.cpix_id`, потому что для VOD нет опции `meta`. Этот `drm_id` будет использоваться в запросах CPIX.

Пример запроса Когда запрашивается поток `mystream`, *Flussonic* отправляет POST-запрос по URL-адресу `http://my.keyserver` со следующим телом запроса:

```
<?xml version="1.0" encoding="UTF-8"?>
<cpix:CPIX xmlns:cpix="urn:dashif:org:cpix" xmlns:pskc="urn:ietf:params:xml
    :ns:keyprov:pskc" xmlns:speke="urn:aws:amazon:com:speke" id="MYSTREAM">
  <cpix:ContentKeyList>
    <cpix:ContentKey kid="2d70751b-972e-1479-7ef9-9fc835860120"/>
  </cpix:ContentKeyList>
  <cpix:DRMSystemList>
    <cpix:DRMSystem kid="2d70751b-972e-1479-7ef9-9fc835860120" systemId="
   edef8ba9-79d6-4ace-a3c8-27dcd51d21ed"/> # widevine
    <cpix:DRMSystem kid="2d70751b-972e-1479-7ef9-9fc835860120" systemId="9
   a04f079-9840-4286-ab92-e65be0885f95"/> # playready
    <cpix:DRMSystem kid="2d70751b-972e-1479-7ef9-9fc835860120" systemId="94
   ce86fb-07ff-4f43-adb8-93d2fa968ca2"/> # fairplay
  </cpix:DRMSystemList>
</cpix:CPIX>
```

Поля, используемые в запросе и ответе, такие как ContentKeyList и DRMSystemList, описаны в документе DASH-IF Implementation Guidelines: Content Protection Information Exchange Format

Пример ответа Flussonic ожидает ответа следующего вида:

```
<?xml version="1.0" encoding="UTF-8"?>
<cpix:CPIX xmlns:cpix="urn:dashif:org:cpix" xmlns:pskc="urn:ietf:params:xml
    :ns:keyprov:pskc" xmlns:speke="urn:aws:amazon:com:speke" id="MYSTREAM">
   <cpix:ContentKeyList>
      <cpix:ContentKey explicitIV="" kid="2d70751b-972e-1479-7ef9-9
   fc835860120">
         <cpix:Data>
            <pskc:Secret>
               <pskc:PlainValue>iufSFDzgKQ+6pnV88WyZnA==</pskc:PlainValue>
            </pskc:Secret>
         </cpix:Data>
      </cpix:ContentKey>
   </cpix:ContentKeyList>
   <cpix:DRMSystemList>
        <cpix:DRMSystem kid="2d70751b-972e-1479-7ef9-9fc835860120" systemId
   ="94ce86fb-07ff-4f43-adb8-93d2fa968ca2">
            <cpix:URIExtXKey>
   aHR0cHM6Ly83azR5dHV4cTVkLmV4ZWN1dGUtYXBpLnVzLXdlc3QtMi5hbWF6b25hd3MuY29tL0VrZVN0YWdl
   </cpix:URIExtXKey>
```

```
        </cpix:DRMSystem>
      <cpix:DRMSystem kid="2d70751b-972e-1479-7ef9-9fc835860120" systemId="
   edef8ba9-79d6-4ace-a3c8-27dcd51d21ed">
        <cpix:PSSH>AAAAd3Bzc2gAAAAA7e+LqXnWSs6jyCfc1R0h7QAAAFcIARIQzLxOBq
   /7
   WMlQjQ4jrSMwnxoIbW92aWRvbmUiM3sia2lkIjoiekx4T0JxXC83V01sUWpRNGpyU013bnc9PSIsInRyYWNr
   =</cpix:PSSH>
      </cpix:DRMSystem>
      <cpix:DRMSystem kid="2d70751b-972e-1479-7ef9-9fc835860120" systemId
   ="9a04f079-9840-4286-ab92-e65be0885f95">
        <speke:ProtectionHeader>
   mAIAAAEAAQCOAjwAVwBSAE0ASABFAEEARABFAFIAIAB4AG0AbABuAHMAPQAiAGgAdAB0AHAAOgAvAC8AcwBj
   +ADwAQQBMAEcASQBEAD4AQQBFAFMAQwBUAFIAPAAvAEEATABHAEkARAA+
   ADwALwBQAFIATwBUAEUAQwBUAEkATgBGAE8APgA8AEsASQBEAD4AQgBrADYAOAB6AFAAdQB2AHkAVgBoAFEA
   +
   AGgAdAB0AHAAcwA6AC8ALwBwAGwAYQB5AHIAZQBhAGQAeQAuAGUAegBkAHIAbQAuAGMAbwBtAC8AYwBlAG4A
   +ADwALwBXAFIATQBIAEUAQQBEAEUAUgA+AA==</speke:ProtectionHeader>
        <cpix:PSSH>
   AAACuHBzc2gAAAAAmgTweZhAQoarkuZb4IhflQAAApiYAgAAAQABAI4CPABXAFIATQBIAEUAQQBEAEUAUgAg
   +
   ADwASwBFAFkATABFAE4APgAxADYAPAAvAEsARQBZAEwARQBOAD4APABBAEwARwBJAEQAPgBBAEUAUwBDAFQA
   +
   ADwASwBJAEQAPgBCAGsANgA4AHoAUAB1AHYAeQBWAGgAUQBqAFEANABqAHIAUwBNAHcAbgB3AD0APQA8AC8A
   +
   AGwAMQA2AFcAdgBwAGsANQBUAHAAUQA9ADwALwBDAEgARQBDAEsAUwBVAE0APgA8AEwAQQBfAFUAUgBMAD4A
   /AHAAWAA9ADUAMQA0ADUAOAA5ADwALwBMAEEAXwBVAFIATAA+
   ADwALwBEAEEAVABBAD4APAAvAFcAUgBNAEgARQBBAEQARQBSAD4A</cpix:PSSH>
      </cpix:DRMSystem>
    </cpix:DRMSystemList>
</cpix:CPIX>
```


Опции, которые используются только на *Flussonic* и могут быть полезны:

- `save_template` — сохраняет ответ сервера ключей в файл.
- `dump_url` — логирует запрашиваемый URL.

10.18.5 JPEG-криншоты

Flussonic Media Server умеет вырезать скриншоты из видео. С их помощью вы сможете:

- показать предпросмотр текущего транслируемого видео на веб-странице;
- оценить качество потока;
- захватить нужный момент времени;
- быстро искать в архиве нужный момент видео по скриншотам;
- создать страницу со скриншотами потока, чтобы быстро просмотреть сутки архивной записи;
- делать что вам угодно с маленькими статическими картинками из большого видео потока.

Flussonic Media Server делает скриншоты двумя принципиально разными способами:

- Извлекает видео-кадры в виде JPEG-изображений и может сохранять их в архиве. Создание и хранение JPEG скриншотов потребляет много ресурсов. Подробнее.
- Создает экономичные MP4 видео-скриншоты. В H.264-потоке с ключевыми кадрами уже есть сжатые изображения, для получения которых не нужна ресурсоемкая обработка потока. Flussonic Media Server берет первый ключевой кадр из каждого сегмента и показывает его как MP4 видеофайл, состоящий из одного кадра. Подробности в разделе .

О JPEG-скриншотах

Flussonic Media Server выполняет операцию, которая достаточно требовательна к ресурсам процессора: берётся первый ключевой кадр из сегмента, декодируется и кодируется обратно в JPEG изображение. Выглядит просто, но когда у вас 300 потоков, то этот процесс требует много процессорного времени.

Flussonic позволяет оптимизировать нагрузку: меняя длительность сегмента, вы можете менять количество JPEG скриншотов. Тот факт, что Flussonic Media Server берёт только первый ключевой кадр из сегмента, означает, что если вы настроили длительность сегмента равной трём секундам, у вас будет 20 JPEG изображений в минуту. Если длительность сегмента установлена в 6 секунд, получится 10 JPEG изображений в минуту. Если вы получаете поток с IP камеры, у вас может быть 60 ключевых кадров в минуту, но *Flussonic Media Server* создаст меньшее количество JPEG-изображений.

Когда вы включаете для потока запись архива, все эти JPEG-скриншоты пишутся на диск.

Можно облегчить нагрузку на процессор, используя загрузку скриншотов по заданному URL. Такой способ применяется с IP камерами, так как IP камеры создают свежий JPEG скриншот для показываемого видео. В этом случае *Flussonic Media Server* будет загружать JPEG изображения каждый раз, когда начинается новый сегмент.

Настройка генерации JPEG-скриншотов

Для создания JPEG скриншотов *Flussonic Media Server* использует свой встроенный пакет.

Добавьте опцию `thumbnails` в конфигурацию потока:

```
stream example {
  input fake://fake;
  thumbnails;
}
```

Это запустит отдельный процесс `flussonic-thumbnailer`. Возможно, он будет потреблять значительное количество ресурсов, но, к сожалению, это неотъемлемая черта процессов сжатия видео и изображений.

Настроить получение скриншотов можно также и в панели администратора в настройках потока (**Media** > выбрать поток > **Output**). Выберите опцию **enabled** в разделе **Thumbnails**.

Настройка скачивания JPEG-скриншотов с камеры

Вы можете указать URL, по которому Flussonic Media Server будет запрашивать скриншоты, что уменьшит использование процессора. Многие камеры имеют специальный URL для скриншотов:

```
stream example {
  input rtsp://localhost:554/source;
  thumbnails url=http://examplehost:5000/snapshot;
}
```

Вы можете попробовать найти этот URL в документации на вашу камеру или поискать в Интернете.

Настроить получение скриншотов можно также и в панели администратора в настройках потока (**Media** > выбрать поток > **Output**). Выберите опцию **enabled** в разделе **Thumbnails** и введите **Thumbnail URL**.

Получение live-скриншота

После того как вы включили скриншоты в настройках, их нужно получить.

URL-адрес для доступа к live-скриншоту такой:

`http://FLUSSONIC-IP:80/STREAM_NAME/preview.jpg` — последний скриншот потока.

Figure 10.37: Flussonic JPEG thumbnails

`http://FLUSSONIC-IP:80/STREAM_NAME/preview.mjpeg` — MJPEG поток скриншотов.

Мы рекомендуем никогда не использовать MJPEG, потому что это неконтролируемый способ передачи видео с очень высоким битрейтом. Вы можете получить MJPEG поток с битрейтом до 50 % от исходного потока со скоростью 0,1 кадр в секунду. Используйте этот способ только при необходимости.

Получение JPEG-скриншотов из архива по времени

Скриншоты автоматически сохраняются в архив. Они могут быть получены с помощью HTTP API.

Figure 10.38: Flussonic JPEG thumbnails URL

Экономично по ресурсам получать JPEG скриншоты, указывая приблизительное время в UTC в URL. *Flussonic* найдет ближайший сохраненный скриншот и вернет URL с точным временем.

```
http://FLUSSONIC-IP:80/STREAM_NAME/1652936935.jpg
```

Человекочитаемый формат времени GMT также поддерживается для совместимости:

```
http://FLUSSONIC-IP:80/STREAM_NAME/2022/05/19/05/08/11.jpg
```

По полученному времени находится реальный скриншот.

См. справочник Streaming API.

Получение JPEG скриншотов из архива по промежутку времени UTC

> Этот способ затратный по ресурсам, мы не рекомендуем его использовать. Лучше получать скриншоты по времени UTC или GMT, см. .

Сначала нужно выбрать временной промежуток, для которого нужно прочитать архив. Например, сейчас 21 апреля 2017, 13:10 GMT, что соответствует 1492780200 UTC. Если вы хотите получить скриншоты за последний час, нужно запросить следующий URL:

```
curl 'http://FLUSSONIC-IP:80/STREAM_NAME/recording_status.json?from
   =1492776600&to=1492780200&request=brief_thumbnails'
```

По умолчанию *Flussonic* не включает в ответ список таймстемпов. Для их получения к запросу нужно добавить `request=brief_thumbnails`.

Ответ может быть таким:

```
[{"stream":"clock","ranges":[{"duration":3642,"from":1492776599}],"
   brief_thumbnails":[1492776599,1492776605,
1492776617,1492776629,1492776641,1492776653,1492776665,1492776677,1492776689,1492776701,

....]}]
```

Вы получите длинный список таймстемпов, которые необходимо сконвертировать в пути к скриншотам. Например, `1492776605` будет преобразован в `http://FLUSSONIC-IP:80/STREAM_NAME/2`

Таким образом, сначала вы получаете список таймстемпов, а затем получаете сами скриншоты по сформированным URL-адресам.

Генерация JPEG по запросу

Иногда хранить все JPEG скриншоты на диске очень накладно, и вы можете дать инструкцию *Flussonic Media Server* генерировать JPEG по требованию. В этом случае необходимо добавить в конфигурацию потока параметр `thumbnails enabled=ondemand`. Например:

```
stream channel {
  input fake://fake;
  thumbnails enabled=ondemand;
  dvr /storage;
}
```

Вы также можете настроить получение JPEG скриншотов по запросу в панели администратора, в настройках потока на вкладке **Output** (**Media** > выберите поток > **Output**). Выберите опцию **On demand** в разделе **Thumbnails**.

Запросите URL скриншота для определенного момента времени, указав его в формате UTC:

`http://FLUSSONIC-IP:80/STREAM_NAME/1643797938-preview.jpg`

Figure 10.39: Flussonic JPEG thumbnails on-demand

Можно также указать дату и время в старом удобочитаемом формате, который поддерживается для совместимости:

```
http://FLUSSONIC-IP:80/STREAM_NAME/2022/02/21/12/10/05-preview.jpg
```

Flussonic Media Server прочитает один сегмент, возьмёт первый ключевой кадр после указанного момента времени и сгенерирует JPEG.

Если в текущем сегменте ключевой кадр не найден, Flussonic возьмет первый ключевой кадр следующего сегмента, создаст для него URL и вернет перенаправление 302. Затем браузер отправит новый запрос с новым URL и получит JPEG для найденного ключевого кадра. Такое перенаправление гарантирует, что в кэше для каждого запроса будет сохранен только один уникальный ответ. Таким образом, если для запрошенного времени ключевых кадров не найдено, после перенаправления из кэша будет забран уже готовый нужный URL. При этом два запроса

к соседним секундам, для которых нет ключевых кадров, приведут к скачиванию всего одного файла JPEG.

Этот способ может привести к непредсказуемой загрузке процессора и, поэтому, не рекомендуется.

Непредсказуемая нагрузка здесь означает, что её действительно трудно предсказать. С включенным процессом генерации JPEG у вас ровная умеренная загрузка процессора и не более. С генерацией JPEG по запросу у вас может быть низкая загрузка процессора, но в час наибольшей нагрузки может произойти её резкий рост и работа сервера станет нестабильной.

См. справочник Streaming API.

10.18.6 MP4 видео-скриншоты

О видео-скриншотах

Преимущества видео-скриншотов

Видео-скриншоты очень экономят ресурсы сервера. Мы разработали их, чтобы решить проблемы JPEG-скриншотов — большое потребление времени процессора и места на диске. При этом способе ресурсы не потребляются, потому что JPEG файлы не создаются. Скриншоты, полученные новым способом, называются *видео-скриншоты*.

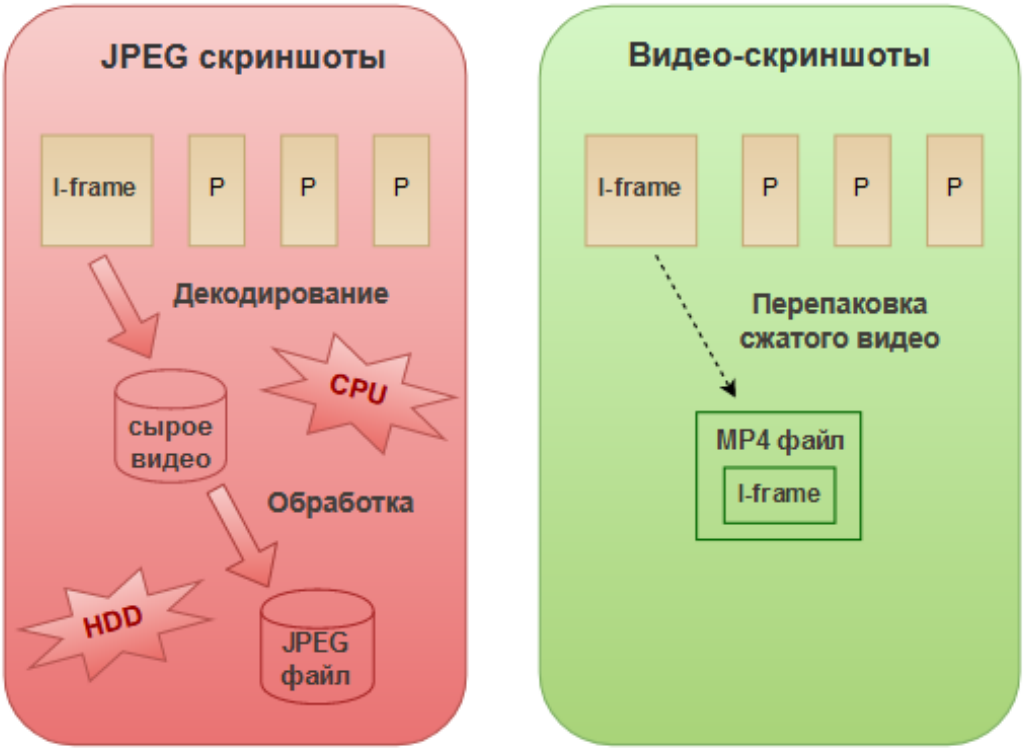


Figure 10.40: видео-скриншоты

Видео-скриншоты совместимы только с H.264 потоками.

Как устроены видео-скриншоты

По сути, видео-скриншоты — это фрагменты видео, содержащие всего один кадр. Если у вас есть видеопоток, сжатый по стандарту H.264, то он уже содержит необходимые изображения — ключевые кадры.

Их не нужно получать путем декодирования, тем самым значительно экономится время процессора. А поскольку Flussonic может извлекать эти кадры из видео ”на лету”, то можно не сохранять их

на диске. Всё, что требуется — получить доступ к готовым кадрам.

Flussonic берет первый ключевой кадр из сегмента и делает из него MP4 файл. Он отсылает этот файл браузеру, где он отображается как картинка. Это и есть *видео-скриншот*.

Пример кода для показа скриншота в браузере:

```
<video src="http://flussonic:80/clock/preview.mp4" style="width: 640px;
   height: 480px;" autoplay />
```

При вставке этого тэга на веб-страницу отобразится скриншот. Можно запрашивать скриншоты для отображения в мобильном браузере, а также они есть в веб-интерфейсе в VoD и в DVR плеере.

> Не забудьте удалить опцию `thumbnails` из настроек потока, она нужна только для JPEG-скриншотов.

Как получить видео-скриншоты Видео-скриншоты нужно запрашивать по специальному URL. Этот URL можно открыть в браузере или поместить в тэг `<video>`, чтобы показать на сайте.

В общем случае, URL имеет вид:

`http://<домен>/<имя потока или путь к файлу>/preview.mp4`

Дополнительно, нужно указать место в видео, на основе которого будет создан скриншот. И здесь есть различия для live потоков, файлов и DVR архивов. Всего четыре случая:

- Скриншоты live потоков – в этом случае Flussonic использует последний ключевой кадр.
- Скриншоты DVR архива. Для них укажите дату и время (точное или приблизительное) как часть URL. Об этом далее на этой странице.
- Скриншоты файлов (VoD). Для них указывайте время относительно начала файла (ЧЧ-ММ-СС). См. далее на этой странице.
- Скриншот первого кадра в файле. Первый кадр иногда содержит просто черный фон.

Получение видео-скриншотов файла

Используя видео-скриншоты Flussonic, вы можете получить скриншот любого места внутри файла, а не только первого ключевого кадра. Вы указываете время, и Flussonic находит ближайший ключевой кадр. На самом деле, Flussonic берет только первый ключевой кадр из каждого сегмента.

Чтобы получить скриншот из файла, добавьте время к URL скриншоту и укажите URL в HTML тэге `<video>`, который вставьте на веб-страницу. Синтаксис URL:

`http://<domain>/<path>/<filename>.mp4/preview-<часы-минуты-секунды>.mp4`

Если не указывать время, то получим изображение самого первого ключевого кадра в файле:

`http://<domain>/<path>/<filename>.mp4/preview.mp4`

Пример ниже показывает, как получить скриншот с момента 02:24:45:

```
<video src="http://flussonic:80/vod/bunny.mp4/preview-02-24-45.mp4" style="
   width: 640px; height: 480px;" autoplay />
```

Flussonic производит редирект на URL с номером вычисленного ключевого кадра вместо указанного времени.

Получение видео-скриншотов из архива

Видеоскриншот из архива доступен тем же способом, что и JPEG скриншот. Для UTC таймстемпа 1654242430 (что соответствует 3 июня 2022 г, 07:47:10 GMT) скриншот будет доступен по URL:

`http://FLUSSONIC-IP:80/STREAM_NAME/1654242430-preview.mp4`

Человекочитаемый формат времени GMT также поддерживается для совместимости:

`http://FLUSSONIC-IP:80/2022/06/03/07/47/10-preview.mp4`

Но мы разработали более удобный способ доступа к таким скриншотам. Если вы знаете, что где-то на протяжении 10 минут после заданного момента во времени у вас есть записанное видео, вы можете сделать запрос по несуществующему URL (по приблизительному времени). *Flussonic* найдет в этом периоде времени существующий ключевой кадр и вернет его. Такой подход помогает меньше загружать кэш: хотя браузер выполняет два запроса, лишь существующий ключевой кадр сохранится в кэше браузера.

Например, вы можете запросить несуществующий URL для момента на 10 минут раньше, чем в предыдущем примере:

`http://FLUSSONIC-IP:80/STREAM_NAME/1654241830-preview.mp4`

Для существующего скриншота *Flussonic Media Server* пришлет HTTP header `X-Thumbnail-Utc: 16542424`3
Его можно использовать для получения настоящего URL скриншота, так как браузер не сообщает вам о редиректе.

Видео скриншоты в мобильном браузере

Чтобы показать видео-скриншот на мобильном устройстве, достаточно использовать такой код:

```
<video src="http://flussonic:80/clock/preview.mp4" autoplay playsinline
   /></video>
```

Параметр `playsinline` нужен для корректного отображения превью (например, чтобы оно не перекрывало расположенные поверх него элементы).

10.19 Ad Insertion

10.19.1 Способы врезки рекламы на стороне сервера

Чтобы монетизировать контент с помощью врезки рекламы и извлечь из этого максимальную выгоду, выберите подходящий способ.

Выделяют два способа врезки рекламы на стороне сервера (Server-Side Ad Insertion, SSAI):

- Врезка рекламы в виде отдельных сегментов (по расписанию).
- Врезка рекламы в виде сегментов основного потока (по меткам SCTE).

С помощью *Flussonic* вы можете настроить SSAI любым из этих способов. Подробнее см. Настройка врезки рекламы.

Врезка рекламы в виде сегментов основного потока

Сервер вставляет рекламу непосредственно в поток для вещания, заменяя сегменты потока.

Врезка рекламы в виде сегментов основного потока имеет следующие особенности:

- Контроль за сессией просмотра на сервере.
- Источник рекламы скрыт от клиента.
- Управление кешированием на стороне CDN для персонализации доставки рекламы.
- Сигнализация о запрете перемотки и паузы в сторону клиента.

Врезка рекламы в виде сегментов основного потока позволяет создать хороший зрительский опыт за счет следующих характеристик:

- **Бесшовность** переключения с основного потока на рекламный ролик. Рекламный ролик идет в основном потоке и не подгружается отдельно.
- **Скорость загрузки рекламы**. Поскольку реклама идет в основном потоке, плееру не нужно отдельно запрашивать ее у сервера.
- **Обход блокировки рекламы**. Реклама врезается в основной поток так, что URL сегмента основного потока не отличается от URL сегмента рекламного ролика в манифесте. В таком случае распознавание рекламы требует декодирования кадра на плеере, что технически сложно и дорого.
- **Качество проигрывания потока**. При переключении на рекламу и обратно для зрителя не происходит смены профиля. Это достигается за счет предварительного транскодирования рекламных роликов под каждый поток в соответствии с выходными профилями потока.

- **Стабильная скорость загрузки сегментов** потока за счет единого источника основного потока и рекламного ролика. Плеер скачивает сегменты с рекламой и контентом из одного источника. Не нужно подкачивать начало рекламного ролика параллельно с основным контентом.
- **Персонализация рекламы** для зрителей. При запросе потока у сервера плеер передает такие данные, как User-Agent и IP-адрес, на основе которых подбирается реклама. CDN не кэширует манифест потока с рекламой, поскольку этот манифест является уникальным для каждой сессии проигрывания.
- **Сбор статистики по просмотру рекламы** зрителями.
- **Приватность данных** на стороне сервиса. Данные не покидают ваши системы, только вы знаете, кто смотрел какие ролики и решаете, какими данными и с кем делиться.

Как работает SSAI в виде сегментов основного потока Сегменты потока в сессии проигрывания заменяются так, что отличить URL сегмента основного потока от URL рекламного сегмента невозможно:

URL сегмента основного HLS-потока	**‘http://FLUSSONIC-IP/STREAM_NAME/tracks-v1a1/2021/09**
URL **сегмента рекламного потока**	`http://FLUSSONIC-IP/STREAM_NAME/tracks-v1a1/2021/09`

Врезка рекламы в виде сегментов основного потока с помощью авторизационного бэкенда по расписанию работает следующим образом:

- Плеер запрашивает манифест потока у сервера, передавая такие данные, как User-Agent и IP-адрес клиента для персонализированной рекламы. Это также может быть и токен.
- Сервер передаёт полученные данные авторизационному бэкенду.
- На основе полученных данных авторизационный бэкенд возвращает серверу список рекламных роликов для врезки.
- Сервер подбирает рекламные ролики в соответствии с характеристиками основного потока, формирует манифест основного потока с рекламными роликами и передаёт этот манифест плееру.
- Плеер получает манифест и начинает проигрывание.

Врезка рекламы по SCTE-меткам работает похожим образом.

Врезка рекламы в виде отдельных сегментов

При врезке рекламы в виде отдельных сегментов плеер запрашивает рекламу у источника рекламы, когда встречает метку начала рекламы в манифесте.

Врезка рекламы в виде отдельных сегментов имеет следующие особенности:

- Источник рекламы отделен от источника контента. Плеер скачивает основной поток и рекламный ролик отдельно.
- Вся информация о врезке рекламы передается на плеер.
- Не все плееры могут проиграть поток с рекламой из-за разнообразия версий плееров на стороне зрителя (уникальные User-Agent).
- Переключение на рекламу и обратно требует предзагрузки в буфер для бесшовности.
- Профили основного потока и рекламного ролика могут отличаться. Это влияет на зрительский опыт, поскольку реклама может не дойти до зрителя.

Как работает SSAI в виде отдельных сегментов Врезка рекламы в виде отдельных сегментов по расписанию через авторизационный бэкенд работает следующим образом:

- Плеер запрашивает манифест потока у сервера, передавая такие данные, как User-Agent и IP-адрес клиента для персонализированной рекламы. Это также может быть и токен.
- Сервер передаёт полученные данные авторизационному бэкенду.
- На основе полученных данных авторизационный бэкенд возвращает серверу список рекламных роликов для врезки.
- Сервер вставляет ссылки на рекламные ролики в манифест основного потока и передаёт этот манифест плееру.
- Плеер получает манифест и начинает проигрывание.

Врезка рекламы со SCTE-меткам работает так:

- Плеер запрашивает манифест потока у сервера.
- Сервер отдает плееру манифест.
- Плеер начинает проигрывание.
- Как только плеер встречает метку начала врезки рекламы в манифесте, он останавливает проигрывание основного потока и запрашивает рекламный ролик у сервера по API.
- Сервер возвращает информацию о рекламном ролике.
- Используя полученную информацию, плеер загружает нужный рекламный ролик и воспроизводит его.
- Как только плеер встречает метку окончания врезки рекламы, то он переключается обратно на основной поток.

10.19.2 Метки врезки рекламы

Для добавления рекламных роликов в линейные потоки, например, трансляции по ТВ в прямом эфире используется метод динамической вставки рекламы (Dynamic Ad Insertion, *сокр.* DAI). С его помощью региональный оператор врезает локальную рекламу в трансляцию.

Метод динамической вставки рекламы использует *метки* для того, чтобы подготовить поток к врезке рекламы. **Метки врезки рекламы** — это метаданные потока, которые сигнализируют о *событии врезки рекламы*. **Событие врезки рекламы** означает, что в определенном месте в видеопотоке можно врезать рекламный ролик определенной длительности. В этой точке потока возможно плавное переключение на рекламу и обратно на основное видео.

Существуют разные стандарты меток, самый популярный из которых SCTE-35. Метки представляют собой теги в клиентском манифесте потока, поэтому метки необходимо создавать отдельно для передачи по каждому протоколу, который поддерживает адаптивное вещание.

Flussonic может читать метки и преобразовывать их в другой формат. *Flussonic* работает с метками в форматах SCTE-35, SCTE-104 и AWS, а также распознает события *splice_insert*.

Следующие протоколы поддерживают разные стандарты меток:

- HLS — метки в формате SCTE-35, AWS (см. описание меток в документации AWS), а также метки CUE-OUT/CUE-IN входного потока.
- MPEG-TS — метки в формате SCTE-35.
- Видео с карт Decklink и DekTec — метки в формате SCTE-104.
- M4F/M4S — метки в формате SCTE-35.

Какие преобразования меток врезки рекламы поддерживает Flussonic

При одинаковых форматах входного и выходного потока метки проходят ”сквозь” *Flussonic* без изменений. Если выходной формат отличается, *Flussonic Media Server* может преобразовать метки врезки рекламы из одного формата в другой и передать их в выходной поток.

| Формат входного потока | Метка вставки рекламы | Выходные форматы потоков и меток | | :—————-: | :—————-: | :—————-: | | MPEG-TS (любой протокол, включая SRT и HTTP/UDP) | SCTE-35 | MPEG-TS SCTE-35, HLS AWS, HLS SCTE-35, SDI VANC SCTE-104, DASH AWS | | HLS | SCTE-35, AWS, HLS cue | MPEG-TS SCTE-35, HLS AWS, HLS SCTE-35, DASH AWS | | SDI | SCTE-104 | MPEG-TS SCTE-35, HLS AWS, HLS SCTE-35, DASH AWS | | M4F, M4S | SCTE-35 | MPEG-TS SCTE-35, HLS AWS, HLS SCTE-35, DASH AWS |

Пример: Необходимо прочитать метки SCTE-35 в транспортном потоке MPEG-TS и преобразовать их в стандарт, пригодный для вещания по HLS. *Flussonic* распознает метки в захваченном потоке, преобразует их в указанный стандарт и добавит в манифест для проигрывания у клиента.

Настройка преобразования меток врезки рекламы

Чтобы получить метки в выходном потоке, необходимо:

- включить чтение и обработку меток во входном потоке
- (только для HLS) указать формат меток, который вы хотите получить в выходном потоке.

Всё это нужно указывать в настройках **входного** потока.

Чтение меток во входном потоке Чтобы включить чтение меток во входящем потоке:

- для MPEG-TS, M4F, M4S: дополнительные опции не требуются; метки SCTE-35 считываются автоматически.
- для SDI: метки SCTE-104 автоматически читаются из VANC.
- для HLS: добавьте опцию источника `ad=true`, чтобы включить чтение и обработку меток из входного HLS потока. Поддерживается чтение меток как в AWS формате, так и простых cue-меток SCTE-35.

Так *Flussonic* будет считывать метки во входящем HLS потоке.

Настройка формата меток для вывода в HLS Следующий шаг — задать формат меток для *выходного* потока, запрошенного по HLS.

> В выходной MPEG-TS или DASH метки (SCTE-35 или AWS соответственно) передадутся сами.

Метки SCTE-104 из VANC также будут переданы в MPEG-TS, HLS или DASH автоматически.

Для конвертации меток добавьте в настройки входящего потока директиву `hls_scte35`:

```
hls_scte35 aws|scte35|rfc8216;
```

, где:

- `aws` – метки AWS в выходном HLS,
- `scte35` – метки SCTE-35 в выходном HLS,
- `rfc8216` – метки SCTE-35 в выходном HLS, соответствующие стандарту RFC 8216. Передаются в теге `EXT-X-DATERANGE`. Этот формат поддерживается устройствами Apple.

Пример:

```
hls_scte35 scte35;
```

Flussonic проставит метки в формате SCTE-35 в выходной HLS-поток.

> Директива `hls_scte35`, задающая выходной формат меток, имеет смысл только для вывода в HLS.

Пример из двух частей В примере ниже вы узнаете как:

- Получить SCTE-35 метки из MPEG-TS.
- Конвертировать метки в HLS AWS для использования в выходном потоке, запрошенном по HLS.

Получение MPEG-TS SCTE-35 меток, конвертация их в AWS для этого потока, передаваемого по HLS (`hls_scte35 aws`):

```
stream STREAM_NAME {
    input tshttp://IP-ADDRESS:PORT/INPUTSTREAM/mpegts;
    hls_scte35 aws;
}
```

Теперь если запросить этот поток по HLS, в нём будут метки в формате AWS.

Кроме того, есть возможность прочитать эти метки и задать возможность их дальнейшего вывода в формате SCTE-35 при проигрывании потока по HLS, подключившись к получившемуся потоку локально по HLS:

```
stream STREAM_NAME_HLS {
    input hls://FLUSSONIC-IP:PORT/STREAM_NAME/tracks-v1a1/mono.m3u8 ad=true
    ;
    hls_scte35 scte35;
}
```

Проверка потока на наличие меток SCTE-35

Если нужно убедиться, что выходной поток действительно содержит метки SCTE-35, посмотрите логи Flussonic. *Flussonic* создает событие 'scte35', оно находится на стандартном уровне журнала. В журнале `/var/log/flussonic/flussonic.log` будут отображаться сообщения `mpegts_scte35`.

Чтобы вам показывалось событие `scte35`, добавьте эти строки в конфигурацию:

```
event_sink events {
  url log:///var/log/flussonic/example.log;
  only event=scte35;
}
```

Другой способ проверить наличие меток — посмотреть их в HLS манифесте. Это менее удобно, к тому же вам придётся ждать появления меток во время проигрывания в реальном времени.

10.19.3 Файлы VOD

VOD (Video On Demand) — неотъемлемая часть услуг, связанных с передачей видео. Это система персонализированной доставки мультимедиа, которая позволяет пользователям получать доступ к контенту в любое время вне зависимости от привычного телевизионного расписания. VOD имеет широкую сферу применения, например, сфера образования.

Flussonic Media Server поддерживает воспроизведение видеофайлов в приложениях-клиентах. Для этого необходимо настроить *виртуальный путь к файлу*, называемый ***VOD-локацией***. Одна VOD-локация может содержать несколько каталогов. Можно использовать несколько VOD-локаций для организации видеофайлов. VOD-локации удобны, чтобы применять различные наборы настроек к файлам в каждой локации.

Поддерживаемые контейнеры и кодеки

Flussonic Media Server умеет раздавать видео из файлов в контейнерах MP4 с видеокодеками H.264, H.265 (HEVC) и аудиокодеками AAC, MP3, AC3, PCMA, PCMU.

> Мы рекомендуем конвертировать файлы из MKV в MP4, потому что формат MP4 предпочтительнее для проигрывания файлов по HLS или DASH. Для конвертации из MKV в MP4 можно использовать ffmpeg.

Контейнер | Видео | Аудио — | — | — MP4 (.mp4, .f4v, .mov, .m4v, .mp4a, .3gp, .3g2) | H.264, H.265 | MP3, AAC (все профили)

Как видно из списка, *Flussonic* не поддерживает формат MKV, и тому есть причины.

В файле формата MP4 в заголовке заранее есть все данные о дорожках и сегментах. Достаточно прочитать moov-структуру MP4-файла, чтобы *Flussonic* мог узнать всё обо всех кадрах (кроме их содержимого). А так как moov занимает менее 1% всех данных, то прочитать нужно только очень малую часть многогигабайтного файла. И этих данных достаточно, чтобы создать HLS или DASH плейлист.

Самое важное здесь, что в moov содержатся данные о битрейтах, поэтому в случае с MP4 плеер сразу получит валидный мастер-плейлист с данными о битрейтах дорожек, что позволит проиграть файл без ошибок. Если нет данных о битрейтах, плеер не сможет выбрать дорожку для проигрывания. Могут возникать и другие плохо устранимые ошибки.

В случае с MKV-файлами данные о структуре могут отсутствовать. Упаковщики MKV иногда прописывают NUMBER_OF_BYTES, но не всегда, и в этом случае пришлось бы при открытии читать весь файл для того, чтобы узнать его содержимое и сформировать плейлист.

Fragmented MP4

Flussonic поддерживает файлы VOD в формате fragmented MP4 (fMP4). Fragmented MP4 не имеет собственного расширения файла и использует то же расширение .mp4, что и стандартные файлы MP4. Разница между fMP4 и традиционным MP4 заключается в следующем:

- moov описывает дорожки с настройками декодера, но не описывает кадры;
- данные разбиваются на небольшие куски — фрагменты (то, что в HLS, DASH и Flussonic называется сегментами);
- каждый фрагмент описывается областью moof (movie fragment) и mdat, которые можно декодировать, не имея других фрагментов (нужен лишь moov из начала);
- в конце файла есть индекс mfra, который помогает плееру отрисовать таймлайн и перемещаться по нему.

Совместимость с fragmented MP4 важна для VOD потому, что в этом формате записывают видео многие популярные программы, такие как OBS Studio. Вы можете просто поместить сделанную такой программой запись в директорию VOD и раздавать ее с Flussonic Media Server. Ничего настраивать не нужно.

10.20 Configuration

10.20.1 Шаблоны конфигурации потоков

С ростом количества потоков на сервере (например, 10 и более) настраивать и администрировать их становится всё сложнее, особенно в случаях, когда фрагменты настроек дублируются для многих потоков. В таких случаях приходится копировать одни и те же опции для каждого потока, что неэффективно и трудоёмко. К тому же велика вероятность допущения ошибки при настройке таких потоков. Здесь на помощь приходят *шаблоны конфигурации потоков*, или просто *шаблоны*.

Шаблоны конфигурации потоков — это фрагмент опций или настроек для конфигурации потока, который необходим для эффективного и удобного процесса создания потоков и управления ими.

Что даёт использование шаблонов?

- *Возможность многократного применения фрагмента настроек к разным потокам.*
- *Разбиение сложных частей конфигурации на более простые и управляемые части.*
- *Уменьшение повторения одинаковых фрагментов настроек в разных частях конфигурации.*
- *Упрощение внесения изменений в повторяемых фрагментах настроек потоков.*
- *Повышение читабельности и понятности конфигурации.*
- *Уменьшение трудозатрат на управление конфигурацией и поддержание её в актуальном состоянии.*

Таким образом, шаблоны настройки потоков во *Flussonic* помогают управлять конфигурацией большого количества потоков.

Файл с конфигурацией Flussonic

Вся конфигурация *Flussonic* хранится в одном файле — . У конфигурации свой формат, это не JSON, YAML или INI, но очень простой, легко читается человеком, именно поэтому его часто читают, т.к. это быстрее, чем посмотреть несколько страниц в веб-интерфейсе. На одном экране текстового редактора помещается конфигурация десятка потоков, сразу же видно все глобальные настройки.

Простой синтаксис конфиг файла делает его удобным и для редактирования. Опытные пользователи *Flussonic* часто пишут конфигурацию именно в текстовом редакторе — так, как они привыкли при работе с другим серверным ПО, таким как веб-серверы.

```
http 80;
edit_auth flussonic password;

stream example {
        input udp://192.168.0.1:5000;
```

```
        dvr /storage 7d;
}
```

Пример реальной конфигурации, шести строчек достаточно чтобы определить порт, который будет слушать Flussonic, задать пароль к веб-интерфейсу, создать поток и настроить его запись.

Шаблоны

Мы решили сделать конфигурацию большого количества потоков более удобной. Во *Flussonic 21.03* мы добавили шаблоны конфигурации, секцию `template` и опцию `template`. Вот как теперь выглядит тот же пример:

```
template t1 {
        transcoder vb=1000k deinterlace=true ab=128k;
        dvr /storage 1d;
}
stream channel1 {
        input udp://239.255.0.1:1234;
        template t1;
}
stream channel2 {
        input udp://239.255.0.2:1234;
        template t1;
}
```

Все общие настройки потоков выносятся в отдельную секцию, а внутри потока определяются только уникальные настройки. Начиная с 10 потоков, уже хорошо видно, как облегчается `flussonic.conf`.

Теперь достаточно один раз привязать поток к шаблону, а дальше работать только с его конфигурацией. Синхронизация настроек потоков на кластере транскодеров сведется к копированию вами определенных `template` между серверами.

Внутри `template` можно использовать те же опции, что и внутри секции `stream`.

Переопределение настроек в конфигурации потока

Если для одного из потоков потребуется переопределить один из параметров, то это можно сделать так:

```
template t1 {
        transcoder vb=1000k deinterlace=true ab=128k;
        dvr /storage 1d;
}
stream channel1 {
        input udp://239.255.0.1:1234;
        template t1;
```

```
        dvr s3://minioadmin:minioadmin@minio:9001/test 3d;
}
```

Локальная конфигурация потока `channel1` будет приоритетнее, чем настройка из `template t1`. Мы рекомендуем использовать переопределение настроек для тестирования, для редких исключений, ведь иначе шаблоны потеряют смысл, большое количество переопределений вернет обратно к ситуации, когда придется вручную отслеживать конфигурацию каждого потока.

Глобальные настройки потоков

Такие опции как `on_play`, `url_prefix`, `cluster_key` можно было и раньше указать сразу для всех потоков, но тогда возникала проблема с контролем исключений.

```
on_play http://middleware_example/auth;

stream channel1 { # the stream uses global auth
        input udp://239.255.0.1:1234;
}
stream channel2 { # the stream overrides global auth with local defined
        input udp://239.255.0.2:1234;
        on_play securetoken://key;
}
```

На практике часто получалось, что далеко не всем потокам требуется наследовать общую конфигурацию, и Администраторы отказывались от ее использования. Явное определение более понятно, лучше читается, позволяет совершать меньше ошибок, чем неявное наследование.

```
stream channel1 {
        input udp://239.255.0.1:1234;
        on_play http://middleware_example/auth;
}
stream channel2 {
        input udp://239.255.0.2:1234;
        on_play securetoken://key;
}
```

Глобальные опции очень похожи на templates, не так ли? Поэтому начиная с 21.03 вы увидите сообщение:

```
# Stream templates:
# Template globals currently applies to all streams without templates.
# This will change in future, explicit template usage is recommended.
#template globals {
        on_play http://middleware_example/auth;
#}
```

В этом релизе мы оставим это без изменений, но уже скоро перенесем конфигурацию в отдельный шаблон, который будут использовать все потоки без указанного `template`.

Теперь становится понятно, что поток может наследовать конфигурацию только из одного `template`: либо явно указанного, либо глобального, но никак не из обоих.

Шаблоны и префиксы

Следующая опция связана с использованием динамического имени потока. Шаблон создаёт точку публикации с одной и более локациями для публикации, по количеству указанных префиксов.

Динамическое имя

— термин, описывающий заранее неизвестное Flussonic имя публикуемого потока.

Используя параметр `prefix`, вы можете задать один или несколько префиксов, которые будут использоваться при формировании имён потоков. Общая структура имени потока выглядит так: `PREFIX/STREAM_NAME`.

Итак, конфигурация будет выглядеть следующим образом (настройка `input publish://` обязательна):

```
template example_template {
  prefix foo;
  prefix bar;

  input publish://;
  backup priv/bunny.mp4;
  source_timeout 2;
}
```

В примере выше мы определили шаблон `example_template` и указали префиксы `foo` и `bar`, а также файл-заглушку и время ожидания кадров от источника. Таким образом, когда вы будете осуществлять публикацию потока во *Flussonic* в одну из локаций публикации (`foo` или `bar`), то имя потока будет одним из двух возможных: `foo/STREAM_NAME`, `bar/STREAM_NAME`, в зависимости от выбранной вами локации для публикации.

Например, если вы публикуете RTMP-поток в локацию `foo`, то URL будет выглядеть следующим образом: `rtmp://FLUSSONIC-IP/foo/STREAM_NAME`.

Итак, все настройки, определённые вами в рамках шаблона с использованием префиксов, будут применены ко всем потокам, публикуемым с использованием этих самых префиксов (`foo/STREAM_NAME` или `bar/STREAM_NAME` в нашем примере).

Возможно также указать в конфигурации шаблона специальный пустой префикс (`""`). В этом случае шаблон может быть использован для публикации потока с любым префиксом или даже вообще без префикса.

10.20.2 Внешнее управление потоками

Flussonic предоставляет возможности для работы со статическими потоками и потоками с *динамическим именем*. Он позволяет загружать статические конфигурации потоков, использовать live-локации для публикации потоков с *динамическим именем*.

> **Динамические имена** — это заранее неизвестные *Flussonic* имена потоков, которые он получает от клиента. В большой системе имена потоков неизвестны только серверу *Flussonic*, но известны какой-то внешней подсистеме. Внешняя подсистема генерирует имена потоков, хранит их и выдаёт клиенту по запросу. Затем с этим именем клиент обращается к *Flussonic*. Читайте подробнее в статье Публикация по динамическому имени.

Таким образом, имена потоков могут быть:

- заранее известны *Flussonic* и указаны в конфигурации (статические потоки),
- заранее неизвестны *Flussonic*, но известны внешней подсистеме до обращения к *Flussonic* (потоки с динамическим именем).

Когда система состоит из двух-трёх серверов, то такие механизмы для статических потоков и потоков с динамическим именем работают. Если система расширяется и количество серверов увеличивается, то эти механизмы начинают ломаться и возникают сложности в работе системы.

Трудности управления большим кластером серверов

Если система включает в себя 20 и более серверов, работает с большим количеством потоков как статических, так и с динамическим именем, то возникают следующие трудности:

- Неочевидно как эффективно распределять нагрузку между серверами.
- Для внесения изменений в конфигурацию может быть необходимо обходить все серверы в кластере.

* удалить этот поток из конфигурации всех предыдущих серверов и оставить на текущем,

либо

* вернуть захват потока на вновь работающий сервер и удалить с предыдущих, чтобы избежать двойного захвата потока.

Таким образом, если необходимо внести какие-либо изменения, связанные с настройками потока, то нужно обходить все серверы в кластере. Если один из этих серверов на текущий момент не работает, то, когда он поднимется, он может работать с неактуальными настройками. Эти настройки могут повредить источник.

Во *Flussonic* есть решение проблем двойного захвата и отказа одного из серверов захвата — механизм кластерного захвата (`cluster_ingest`). Этот механизм позволяет автоматически захватывать потоки на другом сервере при отказе одного из серверов в кластере, а также удалять поток с других серверов при восстановлении работы первого. Однако этот механизм решает проблемы одним единственным способом без возможности кастомизации и изменения правил его работы. Поэтому мы придумали механизм управления конфигурацией потоков с помощью внешнего конфигурационного бэкенда — `config_external`.

О config_external

`config_external` — это внутренний механизм *Flussonic*, с помощью которого сервер может скачивать актуальную конфигурацию потоков из конфигурационного бэкенда. **Конфигурационный бэкенд, сервер конфигурации** — это внешний ресурс, который хранит в себе логику управления потоками и связан с базой данных для хранения такой конфигурации.

Принцип работы

Порядок работы `config_external` различается в зависимости от того, реализованы ли на стороне конфигурационного бэкенда отдельные запросы для обновления статических потоков GET /update_streams_list и работы с динамическими потоками GET /dynamic_streams_list. По умолчанию все запросы к конфигурационному бэкенду выполняются через один метод GET /streams.

Подробнее принцип работы в этих двух случаях описан ниже.

Один метод для всех потоков (по умолчанию) `config_external` работает циклично и обновляет конфигурацию раз в две-три секунды. Каждый цикл обновления конфигурации выглядит следующим образом:

- *Flussonic* запрашивает по API (метод GET /streams) актуальный список активных статических потоков у конфигурационного бэкенда и запускает их, если они ещё не были запущены.
- *Flussonic* высчитывает разницу между списком имён уже запущенных на сервере потоков и списком имён статических потоков, который вернул конфигурационный бэкенд.
- *Flussonic* отправляет запрос по API конфигурационному бэкенду с получившимся списком имён потоков (метод GET /streams с указанием `?name=..` в строке запроса), чтобы запросить конфигурацию для этих потоков.

При большом списке имён потоков *Flussonic* поделит этот список на части примерно по килобайту. Затем *Flussonic* будет запрашивать конфигурацию для части списка потоков до тех пор, пока список не закончится. Так снижается нагрузка на конфигурационный бэкенд и количество использованного трафика.

- Конфигурационный бэкенд возвращает конфигурацию для запрошенного списка потоков. Если для каких-то из запрошенных потоков не пришла конфигурация, то они автоматически удаляются с сервера.

Мы **не** рекомендуем использовать один сервер конфигурации для всех серверов *Flussonic* в кластере, поскольку сервер не справится с таким количеством запросов и произойдёт перегрузка. Чтобы избежать этого, мы рекомендуем завести полную или неполную копию базы данных на локальной машине. Если вы используете *Kubernetes*, то это может быть sidecar-контейнер. Так в случае потери связи с центральным сервером конфигурации ваша система будет способна продолжать работу, что сделает ваш сервис надёжнее.

Отдельные методы для обновления статических потоков и запроса динамических потоков
Flussonic запрашивает методом GET /streams актуальный список активных статических потоков у конфигурационного бэкенда.

Если в ответе конфигурационный бэкенд вернул заголовок `X-Config-Server-Separate-Endpoints: t`
то дальнейшие запросы к конфигурационному бэкенду выполняются следующим образом:

- Периодически выполняется GET /streams, и если на сервере *Flussonic* есть запущенные потоки, не включенные в ответ на `GET /streams`, то выполняется GET /update_streams_list для получения конфигурации или удаления этих потоков (схоже с поведением по умолчанию).
- Если у сервера *Flussonic* будет запрошен поток с неизвестным (динамическим) именем, то выполняется GET /dynamic_streams_list для получения конфигурации динамического потока.

Для снижения нагрузки на конфигурационный бэкенд можно отключить использование динамических потоков, если в ответе на GET /streams передать заголовок `X-Config-Server-Dynamic-Streams: false`. В этом случае сервер *Flussonic* не будет принимать запросы потоков с неизвестным именем и не будет направлять их в конфигурационный бэкенд.

Сценарии использования

Конфигурационный бэкенд и `config_external` заменяют собой все механизмы работы с кластером *Flussonic* и предоставляют возможность реализовать такие механизмы под себя.

С помощью `config_external` вы можете разработать любую логику распределения потоков по серверам кластера под ваши нужды и задачи. На базе конфигурационного бэкенда вы можете реализовать свою собственную версию геотаргетированного кластерного захвата или механизма ‘source‘ для ретрансляции потоков. Вы можете реализовать следующие механизмы для работы с кластером:

Геотаргетированный кластерный захват Задача — захватить сигнал из источника в одной стране одним из близлежащих серверов.

Алгоритм действий может быть таким:

- *Flussonic* запрашивает конфигурацию потоков у конфигурационного бэкенда.
- Конфигурационный бэкенд просматривает список доступных серверов и выбирает самые географически близкие к источнику.
- Конфигурационный бэкенд возвращает одному из таких серверов *Flussonic* конфигурацию с захватом источника.

Динамические таргетированные републикации Задача — отправлять запрос на публикацию на наименее загруженный транскодер.

Принцип работы следующий:

- Клиент отправляет запрос серверу публикации.
- Сервер публикации обращается к конфигурационному бэкенду и запрашивает конфигурацию потока для клиента.
- Конфигурационный бэкенд смотрит на список доступных транскодеров, определяет из них наименее загруженный и возвращает серверу публикации конфигурацию потока с отправкой (`push`) на наименее загруженный транскодер. При отправке потока сервером публикации не будет потери первого кадра.

Настройка ‘config_external‘

> Не используйте управление потоками с помощью Flussonic API и `config_external` одновременно. Например, если вы попробуете обновить конфигурацию потока, выполнив `Flussonic-API: PUT /streamer/api/v3/streams/{name}` с включенным `config_external`, то *Flussonic* вернёт ошибку HTTP 400.

Настроить адрес внешнего конфигурационного бэкенда можно одним из следующих способов:

- Прописать в файле конфигурации (`/etc/flussonic/flussonic.conf`) глобальную опцию `config_external`.

```
config_external https://example.com/config_backend/streams;
```

- Передать `config_external` через API, см. API Reference.

```
curl --request PUT --url http://127.0.0.1:80/streamer/api/v3/config \
--data '{"config_external":"https://example.com/config_backend/streams"}' \
--header 'Authorization: Basic base64_encoded_username:password' \
--header 'Content-Type: application/json'
```

- Создать переменную окружения `STREAMER_CONFIG_EXTERNAL` и указать в качестве значения путь к конфигурационному бэкенду. Мы рекомендуем использовать этот способ только для k8s окружения (или в другой системе автоматического развертывания). В остальных случаях будет удобнее задавать эту настройку через файл конфигурации, как показано выше.

```
STREAMER_CONFIG_EXTERNAL=https://example.com/config_backend/streams
```

При такой настройке *Flussonic* раз в две-три секунды будет обращаться к конфигурационному бэкенду за:

- актуальным списком потоков, которые должны быть запущены на данном сервере,
- настройками для потоков с динамическими именами, если они есть.

Если конфигурационный бэкенд **не** вернёт настройки для запрашиваемого потока, то *Flussonic* будет использовать конфигурацию потока из конфигурационного файла на диске (`flussonic.conf`). Убедитесь, что вы **не** используете одновременно конфигурационный бэкенд и конфигурационный файл на диске для настройки потоков. Это может привести к **некорректной** работе сервера.

См. также Валидация конфига об отслеживании ошибок в конфигурации.

Управление эпизодами

`config_external` позволяет реализовать механизм защиты определенных частей архива от автоматического удаления, позволяющий решить следующие задачи:

- Организация услуги nPVR (Network Personal Video Recorder), то есть сохранение архива с записанной передачей, у которой известны границы во времени.
- Сохранение архива с видеокамеры, в котором присутствуют важные данные (движение, распознавание лиц и т.п.).

Для защиты записей используются *эпизоды*. ***Эпизод*** представляет собой набор метаданных об участке архива — уникальный идентификатор, время начала участка, опционально время окончания и пр.

Информация об эпизодах хранится на сервере конфигурационного бэкенда (по умолчанию в *Flussonic Central*, см. Защита участков DVR от удаления). Перед тем как начать периодическую очистку архива, *Flussonic* запрашивает у конфигурационного бэкенда список эпизодов для нужного потока и не удаляет те части архива, которая покрывается эпизодами.

Чтобы включить использование эпизодов:

- В ответе на запрос GET /streams передайте заголовок `X-Config-Server-Episodes: true`.
- В конфигурации DVR во *Flussonic* задайте параметр 7Cbody%7Cepisodes-urltag/dvr/operation/dvr-save%7Cbody%7Cepisodes-urlepisodes_url.
- Реализуйте на стороне конфигурационного бэкенда ответ на запрос GET /episodes по тому адресу, который вы пропишете в `episodes_url`.

10.20.3 Валидация конфига

Flussonic выполняет валидацию конфигурационного файла по тем же правилам, что и валидацию API вызовов: по одной и той же OpenAPI схеме. По умолчанию мы предоставляем человекочитаемый формат конфигурационного файла, ориентируясь на тех системных администраторов, которые привыкли редактировать текстовые конфигурационные файлы руками.

При каждом запуске, а также при изменении конфигурационного файла *Flussonic* выполняет проверку конфигурации на наличие проблем. Если при сохранении конфигурационного файла произошла ошибка или какая-то опция стала неподдерживаемой после обновления версии ПО, то *Flussonic* все равно запустится, но перейдет в аварийный режим. О работе в аварийном режиме свидетельствует то, что в UI доступно только страницы **Config** и **Support**. Кроме того, о состоянии конфига можно судить по параметрам `streamer_status` (код ошибки) и `text_alerts` (текстовое описание ошибки) в ответе на запрос `Flussonic-API: GET /streamer/api/v3/config/st`

Валидация до старта сервера не имеет смысла, поскольку система управления сервисами systemd не поддерживает ситуацию при которой сервис не может работать из-за невалидного конфига и никак не сигнализирует оператору системы об этой ситуации. Т.е. у вас просто будет рестартиться сервис без какой-либо внятной индикации.

В случае, когда конфигурация *Flussonic* происходит не руками человека, а генерируется из внешней системы, задача валидации конфига стоит очень остро, ведь надо как-то проверить корректность работы генератора и получается, что при генерации текстового формата единственный способ — сгенерировать, перезапустить сервер и выяснить, что конфиг был невалидный.

Это решение, конечно, не подходит для работы, поэтому мы рекомендуем использование JSON формата для конфига при генерации сторонними средствами.

Т.е. суть подхода:

- формируете конфиг по указанной нами OpenAPI схеме (формат, совместимый с JSON schema);
- генерируете его JSON-представление;
- валидируете любым внешним валидатором по схеме;
- пишете результирующий конфиг в `/etc/flussonic/flussonic.conf`;
- запускаете сервер.

Схема конфига JSON

Вы можете взять актуальную OpenAPI схему для *Flussonic* в одном из следующих мест:

- на нашем сайте;
- в файле `/opt/flussonic/lib/web-1/priv/schema-v3-public.json`.

В схеме определен тип `#/components/schemas/server_config`, именно по нему валидируется конфиг при чтении.

Текстовый формат сначала транслируется в JSON, а потом валидируется по этой схеме. Вам же мы рекомендуем сразу записывать конфиг в JSON формате.

Для того, чтобы подсмотреть, как выглядит JSON-представление конфига, можно воспользоваться имеющейся утилитой:

```
# /opt/flussonic/bin/validate_config -j
{"listeners":{"http":[{"port":80}]}}
```

Для работы с JSON рекомендуем использовать утилиту `jq`:

```
# /opt/flussonic/bin/validate_config -j | jq
{
  "listeners": {
    "http": [
      {
        "port": 80
      }
    ]
  }
}
```

Выше вы видите представление простейшего конфига:

```
http 80;
```

Валидация конфига JSON

Валидация конфига происходит в два этапа:

- формальная проверка по JSON Schema;
- проверка внешних условий и тех, которые нельзя выразить в схеме, например целостность сертификатов.

Для валидации целостности конфига можно воспользоваться внешней утилитой для валидации по схеме, а можно той, которую мы предлагаем:

```
# cat /etc/flussonic/flussonic.conf
htp 80;
# /opt/flussonic/bin/validate_config -j
{"col":1,"config":{},"detail":"htp","error":"unknown_command","line":1,"
   path":[]}
```

Аналогично будет с JSON-конфигом:

```
# cat /etc/flussonic/flussonic.conf
{"listeners":{"http":[{"port":80,"flag":true}]}}
# /opt/flussonic/bin/validate_config -j
{"error":"unknown_key","path":["server_config","listeners","http",0,"flag
   "]}
```

10.21 API

10.21.1 Описание Streaming API

Общая информация

Flussonic предоставляет специальное **Streaming API** (см. публичный справочник), которое позволит вам создать свой собственный плеер или другое приложение с использованием всех возможностей проигрывания во *Flussonic Media Server*.

Методы этого API по сути представляют собой URL-адреса, с помощью которых плеер может проигрывать видеопотоки и файлы по различным протоколам (подробнее см. в главе Проигрывание).

Помимо проигрывания потоков и файлов Streaming API позволяет:

- публиковать потоки по некоторым протоколам,
- управлять изображениями (генерировать скриншоты, получать логотип потока),
- получать информацию о содержимом и о статусе записи DVR проигрываемого потока.

Авторизация

Streaming API работает в контексте сессии проигрывания под токеном зрителя. Токен добавляется к строке запроса, как описано в главе Авторизация.

Возможно также использовать внешнюю авторизацию через авторизационный бекенд.

Примеры запросов API

Пример использования метода `Streaming-API: GET /{name}/index.m3u8` для получения мастер-плейлиста HLS:

```
curl http://FLUSSONIC-IP:8080/stream1/index.m3u8?token=60334b207baa
```

Чтобы проиграть поток, эту ссылку необходимо передать в какой-либо HLS плеер, например, STB, мобильное приложение или веб приложение.

Пример использования метода `Streaming-API: GET /{name}/{from}-preview.jpg` для получения JPEG скриншота из архива DVR:

```
curl http://FLUSSONIC-IP:8080/stream1/1650864271-preview.jpg?token=60334b207baa
```

10.21.2 Сессии (сеансы) проигрывания потоков во Flussonic

Содержание

Понятие стриминговой сессии ***Стриминговая сессия** *во* Flussonic – *это интерактивный обмен данными между* Flussonic Media Server *и* внешней системой. *Под* внешней системой* понимаются:

- головная станция,
- плеер,
- другой сервер (*Flussonic* и не только) и т.п.

Сессия определяется продолжительностью, т.е. она устанавливается в определённый момент времени и позже завершается. Во время сессии по крайней мере одна из сторон сохраняет информацию о текущем состоянии и истории сессии, чтобы сообщение было возможным.

Сессия также характеризуется *типом* и *состояниями*. ***Тип сессии** *определяет направление потока и его инициатора, а сама сессия от начала до завершения проходит через несколько* **состояний**. *Начинает (открывает) сессию* **инициатор***, а завершение (закрытие) сессии может быть осуществлено любой из сторон: инициатором или приёмником.

Ниже мы расскажем, какие сессии существуют во *Flussonic* и что о них нужно знать.

Давайте рассмотрим следующий пример. Когда пользователь начинает смотреть телеканал, он начинает сессию `play`. Когда же он переключается на другой канал, это будет уже начало новой сессии. Проще говоря, ***один зритель и один телеканал — одна сессия***.

До версии 21.02 сессии `play` были единственным видом сессий, которые были учтены в системе событий. Если вы уже работали с системой авторизации, то этот тип сессий должен быть вам знаком.

В версии *Flussonic* 21.03 появились новые типы сессий:

- `publish` — сессия, когда пользователь публикует видео с веб-камеры или из OBS.

- `ingest` — сессия, когда *Flussonic* осуществляет захват указанного вами источника, например, IPTV (`udp://`, `tshttp://` и т.д.), IP-камера (`rtsp://`) или иной (`rtmp://`, `shout://` и т.д.).
- `push` — сессия, когда *Flussonic* копирует поток на другой сервер или в другой сервис, например, на Youtube, Facebook, или отправляет поток в мультикаст-группу.

Мы объединили эти 4 типа сеансов передачи видео (`ingest`, `publish`, `play`, и `push`) в следующую таблицу:

Инициатор	Во Flussonic	Из Flussonic
Пользователь	publish (приём публикации)	play (проигрывание)
Flussonic	ingest (захват)	push (отправка)

Это можно представить в виде следующей схемы:

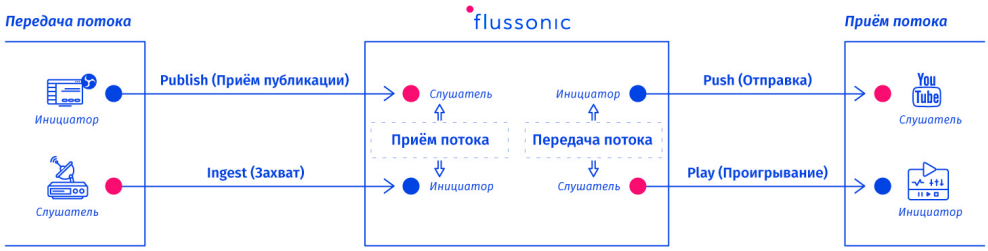


Figure 10.41: Типы передачи потока

Сессии во Flussonic могут быть описаны с помощью OpenAPI 3.0 (читайте подробнее здесь). Справочник нашего публичного API с описанием всех его методов доступен здесь.

Жизненный цикл сессии У *Flussonic* есть единый жизненный цикл всех типов сессий, перечисленных выше. У каждой сессии есть состояния, которые перетекают из одного в другое, вызывая связанные с ними события.

Название события сессии состоит из 2-ух частей (`тип_сессии` и `тип_события`), разделённых символом нижнего подчёркивания (_). Например: `play_opened` (тип сессии: `play`, тип события: `opened`).

Для удобства сессии `ingest` и `publish` генерируют события под одним и тем же именем: `source`.

Пример Рассмотрим пример на основе `play`:

- Пользователь делает первый запрос на HLS-поток. Событие `play_opened` генерируется при открытии новой сессии `play` (событие `opened`).

- Бэкэнд авторизации разрешает эту сессию, вызывая событие `play_authorized`.
- Плеер начинает приём сегментов. Как только количество принятых сегментов превышает пороговое значение, вызывается событие `play_started`.
- Пока пользователь смотрит видео, периодически генерируется событие `play_updated`, которое можно использовать для сохранения информации о сессии в Middleware.
- По истечении некоторого времени ожидания после последнего запроса сессия считается закрытой, вызывая событие `play_closed`.

Этот процесс можно представить в виде следующей схемы:

Теперь мы можем перейти к общему случаю.

Состояния сессии и события Схема ниже представляет состояния во *Flussonic*, которые могут изменяться в течение сессии, а также события, которые могут генерироваться в связи с этим.

> Мы будем именовать состояния сессий с *заглавной буквы*, а события и типы сессий — *со строчной*.

Как происходит переход из одного состояния в другое?

Когда сессия открывается, т.е. переходит из состояния None в `Establishing`, то генерируется событие `opened`. Состояние `Establishing` означает, что сессия ещё только подключается (событие `connected`), готовится, проверяет авторизацию (событие `authorized`), т.е. передачи потока ещё не происходит.

Сессия может завершаться, переходя в состояние `Finished`, с вызовом события `closed`.

Сессии вида `publish` и `play` в состоянии `Establishing` и `Running` могут вызывать событие `authorized`.

В состоянии `Establishing` сессия:

- ожидает первый кадр или первый ключевой кадр от источника, если это сессия типа `source`(т.е. `ingest`/`publish`);
- ожидает достаточного количества байтов, если это сессия типа `play`/`push`.

Затем сессия переходит в состояние `Running`, вызывая событие `started`.

В состоянии `Running` сессия периодически обновляется (`updated`), что позволяет отслеживать сессию. Используйте событие `updated`, чтобы обновить запись в базе данных для этой сессии, поскольку в таком случае предыдущая информация об этой сессии перезаписывается.

При смене *входного или выходного битрейта* или *media_info* в состоянии `Running` вызывается событие `altered`.

В состоянии Running событие overflowed может быть сгенерировано в 2-ух случаях:

- для play или push:

Если не получается передавать данные с требуемой скоростью.

- для source (ingest/publish):

Если протокол передачи видео сообщит об этом, как это делают RTSP/RTCP и SRT.

Если произошла остановка передачи данных, то осуществляется переход из состояния Running в состояние Stalling, который сопровождается вызовом события stalled. Это состояние возникает в том случае, если обратный переход в состояние Running потенциально возможен (такой переход сопровождается событием recovered).

Сессии, инициированные извне *Flussonic* (play и publish), должны проходить авторизацию во внешней системе. Эта внешняя система должна давать ответ на периодические запросы от каждой сессии и уметь останавливать сессию (с вызовом события denied).

Мы собрали все состояния и события в следующую таблицу:

Смена состояний	Тип сессии	Выз
None -> Establishing	source (ingest/publish), play, push	opened Establishing
-> Finished	source (ingest/publish), play, push	
Establishing -> Establishing	source (ingest/publish), play, push	authorized (p
Establishing -> Running	source (ingest/publish), play, push	
Running -> Running	source (ingest/publish), play, push	selected, updated, a
Running -> Stalling	source (ingest/publish), play, push	
Stalling -> Running	source (ingest/publish), play, push	

Как использовать новые события Добавьте директиву event_sink в файл конфигурации (/etc/flussonic/
например:

```
event_sink example {
  url http://examplehost:5000/events;
  only media=example_stream;
}
stream example_stream {
  input fake://fake;
}
```

При такой конфигурации будут отправляться HTTP POST запросы с телом JSON, содержащим сессии, описанные на странице выше.

Подробнее о настройке логирования событий см. Настройка логирования событий.

События подключения к источнику В примере вы можете видеть ряд событий при подключении *Flussonic* к источнику:

- `source_connected` — старт HTTP соединения (`"status":"http_connect"`);
- `source_started` — создание `source_id=7ad153b1-68a5-4304-bbfd-b136603baebd`;
- `stream_updated` — обновление значений `bytes`, `bytes_out` для `source_id=7ad153b1-68`


```
[{
  "event":"source_connected",
  "event_id":1023,
  "id":"4f3c7cec-5c36-4670-921b-a0dcd4a6f0c8",
  "loglevel":"info",
  "media":"example",
  "priority":1,
  "proto":"tshttp",
  "server":"mk1.e",
  "status":{"status":"http_connect"},
  "url":"tshttp://127.0.0.1/fake/mpegts",
  "utc_ms":1614524093408
 },{
  "dts":93606612.44444445,
  "event":"source_started",
  "event_id":1027,
  "id":"4f3c7cec-5c36-4670-921b-a0dcd4a6f0c8",
  "loglevel":"info",
  "media":"example",
  "priority":1,
  "proto":"tshttp",
  "server":"mk1.e",
  "source_id":"7ad153b1-68a5-4304-bbfd-b136603baebd",
  "url":"tshttp://127.0.0.1/fake/mpegts",
  "utc_ms":1614524093414
 },{
  "bytes":0,
  "bytes_out":0,
  "event":"stream_updated",
  "event_id":1028,
  "id":"82c59180-e64e-42fc-8f11-2dec111ca5f7",
  "loglevel":"debug",
  "media":"example",
  "opened_at":1614524093399,
  "server":"mk1.e",
  "source_id":"4f3c7cec-5c36-4670-921b-a0dcd4a6f0c8",
  "utc_ms":1614524093415
  }]
```


Событие начала проигрывания Когда клиент подключается к потоку по протоколу HLS, происходит вывод события `play_opened`:

```
[{
  "bytes":0,
  "country":null,
  "event":"play_opened",
  "event_id":1064,
  "id":"24b79b95-7400-4da2-bf9c-a855603baed1",
  "ip":"192.168.100.7",
  "loglevel":"debug",
  "media":"example",
  "opened_at":1614524113129,
  "proto":"hls",
  "query_string":"token=test",
  "referer":null,
  "server":"mk1.e",
  "source_id":"82c59180-e64e-42fc-8f11-2dec111ca5f7",
  "token":"test","user_agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10
 _15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.192
 Safari/537.36",
  "user_id":null,
  "user_name":"example",
  "utc_ms":1614524113129
 }]
```


Обратим внимание, что `"source_id":"82c59180-e64e-42fc-8f11-2dec111ca5f7"` — это тот же ID, что и в предыдущем примере. Все события связаны посредством этого самого параметра `source_id`.

Все события, связанные с сессиями, перечислены в 7Cevents–0%7Ceventtag/events/operation/available-events-list/response%7Cevents–0%7Ceventсхеме API.

10.21.3 Events API для управления событиями

События Flussonic

Во *Flussonic Media Server* есть удобная и гибкая система внутренних событий с маршрутизацией, обработчиками и возможностями настройки. Данная статья описывает, как настроить *Flussonic* для фильтрации и отправки событий. Список и описание возможных событий см. в 7Cevents–0%7Ceventtag/events/operation/available-events-list/response%7Cevents–0%7Ceventcхеме API.

Подробнее прочитать про работу сессий стриминга можно в отдельной статье.

События инициируются в разных частях системы и могут использоваться для разных сценариев.

Задать настройки, связанные с событиями, можно на странице **Config** -> **Events** или в файле конфигурации *Flussonic* с помощью директивы `event_sink`.

Опция **Sink** (в конфигурационном файле — `url`) определяет получателя событий:

- Чтобы использовать кастомный обработчик, в `url` нужно указать путь до него.
- Чтобы записывать события в журнал событий (лог), в `url` указывают путь к файлу журнала.

Далее укажите различные опции, чтобы отфильтровать события до того, как они попадут в лог или в обработчик.

Содержание:

- Настройка логирования событий
- Настройка обработчиков событий
- Фильтрация событий
- Гарантированная доставка уведомлений

Настройка логирования событий

В дополнение к основному журналу, *Flussonic* позволяет создавать столько лог-файлов, сколько может потребоваться, и записывать в них события, выбранные по разным критериям. Все настройки можно задать на странице **Config** -> **Events**.

Настройку можно выполнить и через конфигурационный файл, добавив директиву `event_sink` и указав путь к файлу журнала в опции `url log://`:

```
event_sink log_name {
    url log:///var/log/flussonic/crash.log;
    level debug;
}
```

Основные настройки:

- **Name** (`log_name`) — просто название настройки. Для удобства можно давать содержательные названия.
- **Sink** (`url`) — получатель событий. В случае логов это файл, в который нужно записывать информацию о событиях. См. также Настройка обработчиков событий
- **Level** (`level`) — уровень логирования по степени важности событий. `debug` (самое подробное логирование), `info`, или `alert` (логирование только важных событий), `notice`, `warning`, `error`, `critical`.

Настройка обработчиков событий

Flussonic может отправлять события любому получателю, которого вы укажете в параметре **Sink** (`url`). Это может быть путь к файлу скрипта, URL удаленного обработчика и т.п. Пример:

```
event_sink handler_name {
    url http://IP-ADDRESS:PORT/SCRIPT_NAME.php;
}
```

Такое объявление создаст обработчик событий с именем `handler_name` и он будет отсылать все события на HTTP URL `http://backend.local/notify.php`.

В этой конфигурации все события *Flussonic Media Server* будут отправляться в JSON-формате как список объектов.

В высоконагруженной системе может генерироваться огромное количество событий, большая часть которых не требуется. Используйте фильтрацию, чтобы уменьшить поток событий.

Вызовы обработчика происходят синхронно: событие не будет отправлено в обработчик, пока он не завершит обработку предыдущей порции событий.

В конфигурации обработчика событий можно указать следующие опции:

Опция	
`url`	
`only`	Белый список ограничений. Можно указать несколько `key=value` и
`except`	
`buffer`	
`sign_key`	(Указывается в разделе **Extended** (`extra`)) Вы можете указать ключ подписи для HTTP-приёмн

Все остальные опции в этом блоке будут переданы указанному приёмнику событий. Список доступных опций см. в описании метода API: `Flussonic-API: PUT /streamer/api/v3/event_sinks/`
В LUA-скрипте они доступны в таблице `args`. По HTTP они передаются вместе с другими параметрами.

Фильтрация событий

Вы можете фильтровать события перед тем, как они попадают в обработчик или файл, с помощью опций **Except** (`except`) и **Only** (`only`). Этот механизм уменьшает нагрузку на обработчик событий. Каждое событие проходит через фильтр непосредственно в треде, создавшем это событие, перед тем, как попасть в обработчик.

Правила фильтрации:

- если любая `except` директива полностью совпала с событием, событие выкидывается и не передаётся в обработчик;
- если в объявлении обработчика нет `only` директив, события передаются в обработчик;
- если директивы `only` есть, то событие передается в обработчик, если оно полностью совпало хотя бы с одной директивой `only`.

Полное совпадение директивы с событием означает, что все пары "ключ=значение" в директиве равны значениям в событии. Если в директиве указано "ключ=значение1,значение2,значение3", то это означает, что поле "ключ" в событии должно быть равным какому-то одному значению из заданного списка.

Примеры:

- `only event=stream_opened;` совпадает с `{event: "stream_opened", media: "cbc"}`
- `only event=stream_opened,stream_closed;` совпадает с `{event: "stream_opened", med`
- `only event=stream_opened,stream_closed media=tnt;` **не** совпадает с `{event: "stream_`
- `only event=stream_opened media=cbc group=news;` **не** совпадает с `{event: "stream_ope`

Простой пример конфигурации, чтобы игнорировать события, касающиеся потоков (но передавать другие события, такие как события сервера *Flussonic*):

```
event_sink log_name {
    url log:///var/log/flussonic/events.log;
    except media=*;
    level debug;
}
```

Пример конфигурации, которая пропускает только некоторые события:

```
event_sink handler_name {
    url http://IP-ADDRESS:PORT/SCRIPT_NAME.php;
    only event=stream_opened,stream_closed,source_opened,source_closed;
}
```

Гарантированная доставка уведомлений

Чтобы не терять неотправленные уведомления, можно настроить *Flussonic* на отложенную повторную отправку. Если принимающий HTTP сервер или скрипт не отвечает, *Flussonic* накапливает события в специальном буфере и периодически повторяет попытки отослать нотификации. Когда принимающий сервер начнёт отвечать, *Flussonic* дошлёт накопленные нотификации.

Для этого нужно указать две опции:

- **Resend limit** `resend_limit` — количество последних событий, которое будет храниться с целью повторных попыток отослать их. Не может превышать 2000.
- **Resend timeout** `resend_timeout` — интервал времени в секундах, через который *Flussonic* будет пытаться отослать события.

В файле конфигурации это будет выглядеть так:

```
event_sink watcher {
   url http://IP-ADDRESS:PORT/SCRIPT_NAME.php;
   resend_limit 10;
   resend_timeout 10;
}
```

Сбор данных о рекламе

Вы можете использовать событие `ad_injected` для сбора информации о вставке и проигрывании рекламы в потоках. Каждый раз, когда в проигрываемый поток вставляется реклама, *Flussonic* генерирует это событие с такими параметрами как DTS первого кадра рекламы, путь к рекламным файлам, тип рекламного ролика и его продолжительность. Более подробную информацию смотрите в схеме API (выберите `ad_injected` в параметре `events` ответа).

Это событие записывается в лог *Flussonic*, позволяя отследить, сколько рекламы показал *Flussonic* и кому.

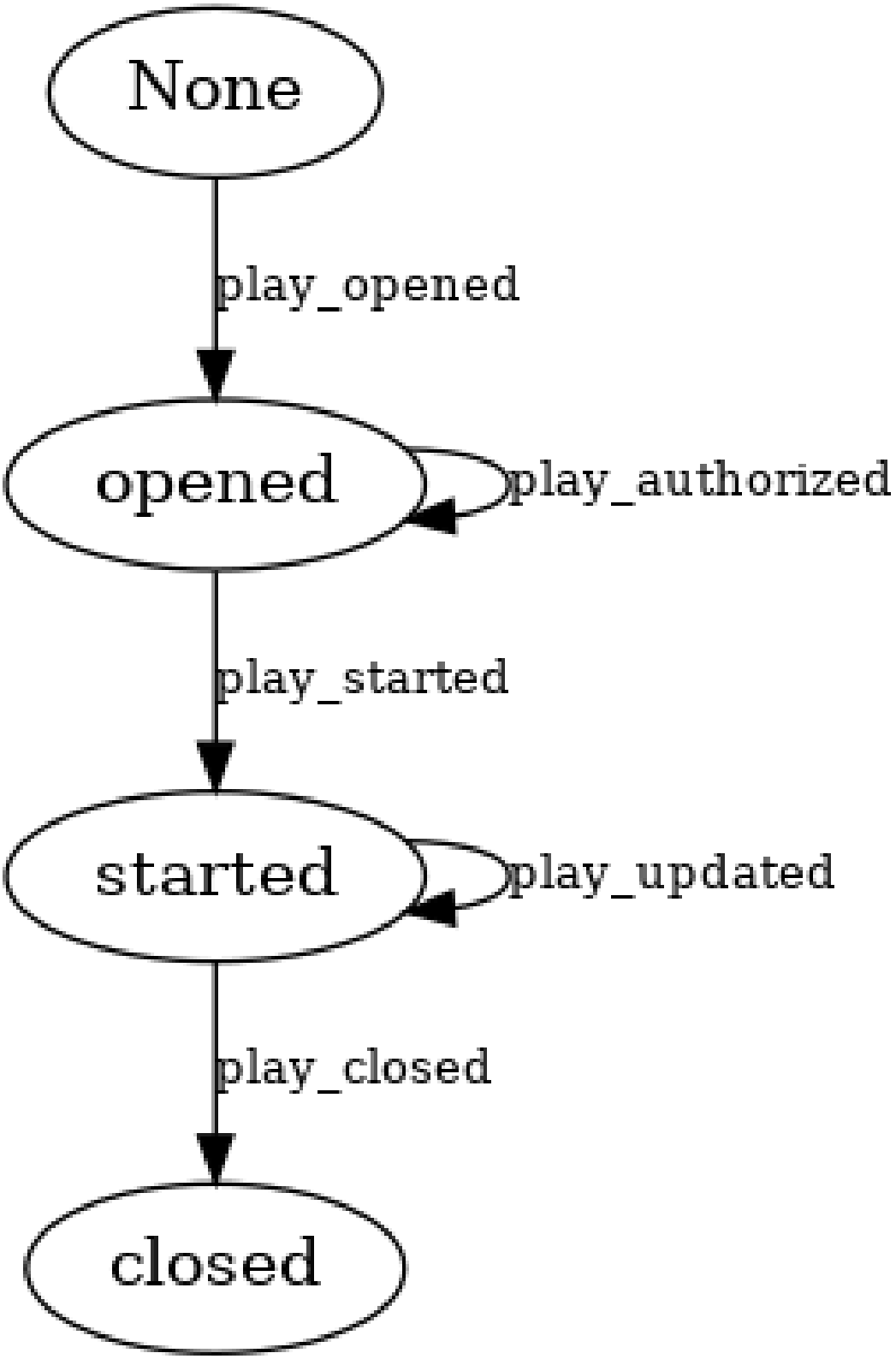

None
play_opened
opened
play_authorized
play_started
started
play_updated
play_closed
closed

Figure 10.42

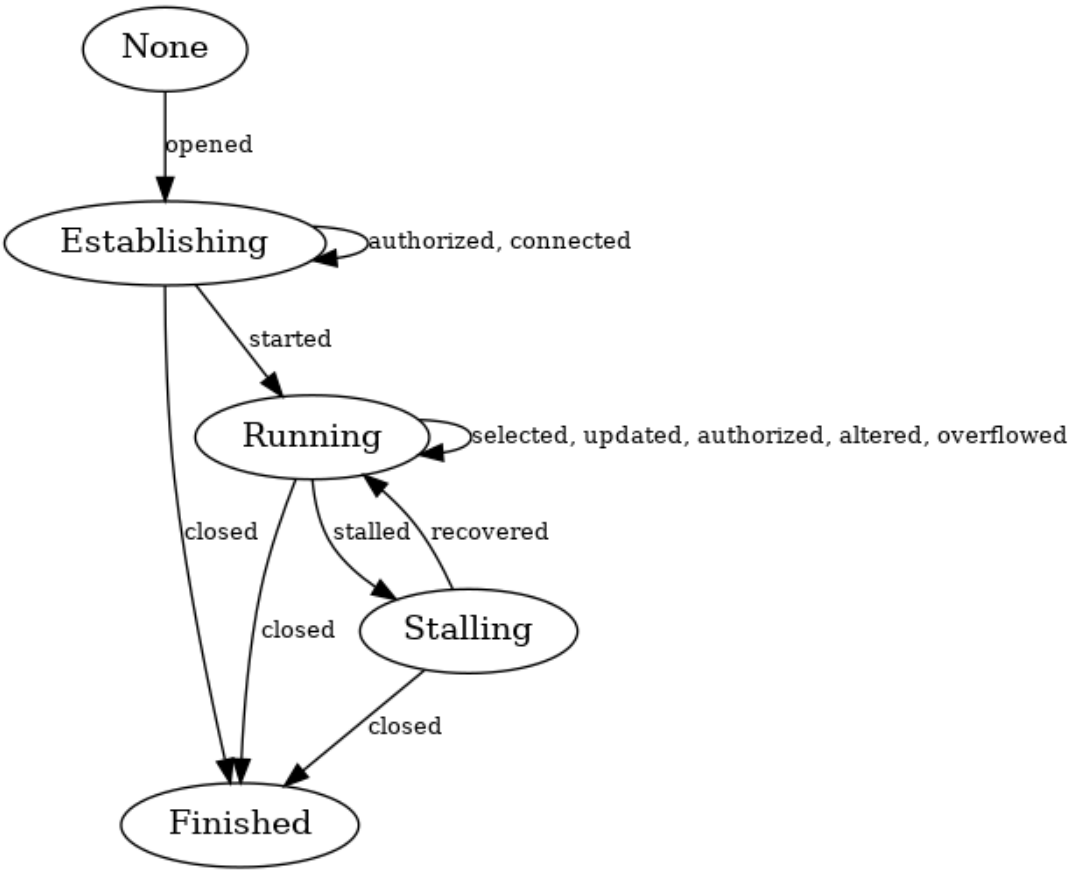

None
opened
Establishing
authorized, connected
started
Running
selected, updated, authorized, altered, overflowed
closed
stalled
recovered
closed
Stalling
closed
Finished

Figure 10.43

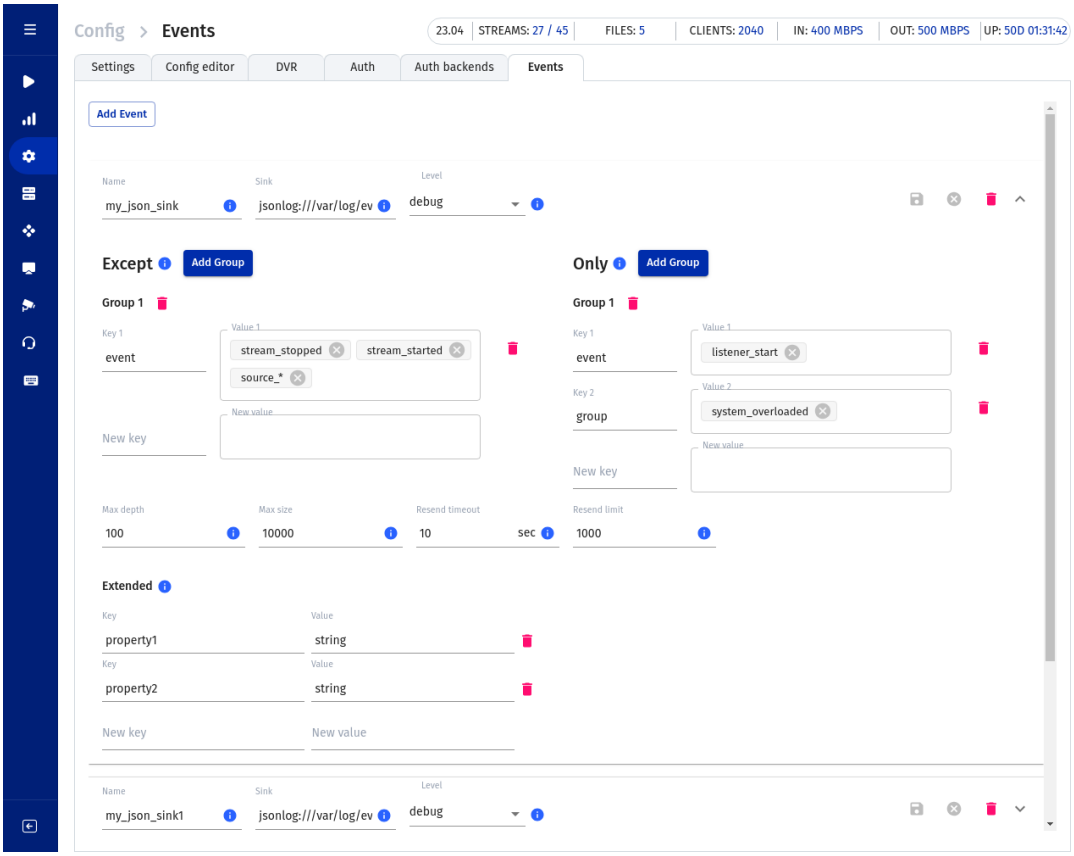


Figure 10.44: Настройки событий

Chapter 11

IP Camera

11.1 IP Camera

11.1.1 Flussonic Agent

Flussonic Agent (далее Агент) — это небольшой модуль к программному обеспечению Flussonic Watcher, который устанавливается производителем на прошивку камеры для обеспечения ее видимости из-за NAT.

Камера с Агентом подключается к Flussonic Watcher к 80 (HTTP) или 443 (HTTPS) порту в обход NAT, организуя прямую связь с серверами (стримерами).

> Обратите внимание, что соединение Агента с управляющим сервером Watcher и стримерами будет зашифровано по протоколу TLS только в случае, если вы настроите HTTPS на всех необходимых серверах. Варианты настройки HTTPS см. здесь.

Если HTTPS не настроен, данные передаются по HTTP в незашифрованном виде.

Агент позволит вам запустить сервис видеонаблюдения без настройки каждой камеры.

Агент является лучшей альтернативой белым IP, проброску портов или OpenVPN.

Как работает Агент

Итак, вы установили и настроили управляющий сервер Flussonic Watcher и стримеры, настроили HTTPS и у вас есть камеры с Агентами, готовые к подключению. Вы подключаете камеру к питанию и к локальной сети. Что происходит дальше?

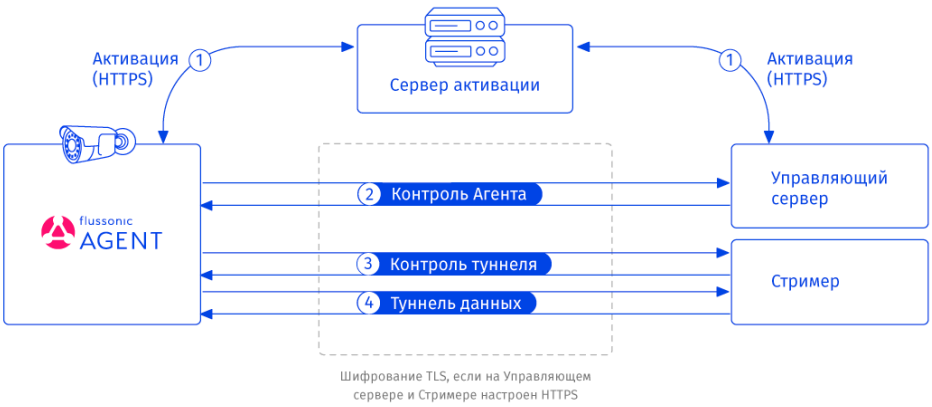


Figure 11.1: Порядок работы Агента

1. Активация При первом запуске камеры выполняется процедура активации Агента: Агент обращается к серверу активации, обслуживаемому компанией Эрливидео, и запрашивает данные для подключения к Watcher. Сервер активации сообщает Агенту адрес управляющего сервера

и секретный ключ для подключения к нему, а управляющему серверу сообщает тот же секретный ключ и идентификатор Агента, подключения которого требуется ожидать. Секретный ключ необходим для защиты от подключений сторонних неавторизованных клиентов и представляет собой аналог одноразового пароля.

Все данные на этом этапе передаются по протоколу HTTPS с шифрованием TLS.

При последующих запусках Агент уже знает адрес управляющего сервера, поэтому этот шаг пропускается. Однако возможны ситуации, когда URL-адрес управляющего сервера перестает быть доступным, например, в случае сбоя или штатного перехода на новое доменное имя. В этом случае Агент не сможет подключиться к управляющему серверу и снова обратится к серверу активации, чтобы запросить новый адрес управляющего сервера. Это очень полезная функция под названием «Повторная активация» или «Реактивация», обеспечивающая бесперебойную работу камер.

2. Провижнинг на Watcher Используя полученный секретный ключ, активированный Агент подключается к предварительно настроенному управляющему серверу с Flussonic Watcher по полученному адресу и сообщает о готовности к передаче видео.

Поскольку управляющий сервер не предназначен для приема видео, он авторизует Агент (проверяя логин и пароль) и сообщает ему адрес одного из запущенных стримеров и секретный ключ для подключения к нему, а стримеру сообщает о возможности подключения Агента. Также управляющий сервер позволяет Агенту быстро переключиться на другой стример в случае падения одного из них.

Это соединение между управляющим сервером и Агентом существует до тех пор, пока Агент не будет деактивирован. По нему Агент передает информацию о своем состоянии, которую вы видите в веб-интерфейсе Watcher, а управляющий сервер передает Агенту команды, например, на подключение к стримеру, деактивацию и перезапуск.

Если на управляющем сервере настроен HTTPS, то все передаваемые данные шифруются по протоколу TLS (инструкцию по настройке HTTPS см. здесь).

3. Подключение к Watcher Агент устанавливает соединение со стримером для контроля туннеля. По этому соединению Агент ожидает команду на открытие соединения для передачи данных, так же, как это устроено в SSH-туннеле. Когда стример решает взять видео с камеры, он по этому соединению посылает команду Агенту с просьбой установить TCP туннель.

Для установки этого соединения стример должен иметь публичный IP-адрес, доступный из Интернета, как указано в инструкции по настройке кластерного режима.

Если на стримере настроен HTTPS, то все передаваемые данные шифруются по протоколу TLS (инструкцию по настройке HTTPS см. здесь).

4. Подключение к стримеру Получив команду со стримера, Агент открывает туннель для передачи данных. Через этот туннель стример отправляет запросы на получение данных, а Агент передает все данные с камеры, включая RTSP-потоки и скриншоты (thumbnails).

Для установки этого соединения стример должен иметь публичный IP-адрес, доступный из Интернета, как указано в инструкции по настройке кластерного режима.

Если на стримере настроен HTTPS, то все передаваемые данные шифруются по протоколу TLS (инструкцию по настройке HTTPS см. здесь).

> После установки и настройки управляющего сервера мы настоятельно не рекомендуем менять его адрес.

О работе Watcher в режиме кластера можно прочитать в этой статье

Сравнение Агента с другими решениями

Существуют следующие альтернативы Flussonic Agent:

Серый IP в локальной сети Это удобный способ для подключения к IP-камерам, если у вас есть эта сеть. Обычно, это означает, что вы строите корпоративную сеть или, что вы строите что-то вроде локальной сети города для таких проектов типа «Безопасного город» или «Безопасный регион». Это не применимо для OTT-провайдеров или когда вам нужно работать с маршрутизаторами, находящимися за NAT.

Белый IP для камер Это наихудшее из возможных решений, т.к. вы рискуете со своими камерами стать частью, например, Mirai ботнета)

Проброс портов Если вы предоставляете сервис видеонаблюдения для домашнего использования или малого бизнеса, то необходимо инструктировать людей, как настроить маршрутизатор, что не самая простая задача. Это либо огромный объем работы для инженеров-установщиков, либо для сотрудников саппорта, которым придется помогать с настройками камер пользователей. Не всегда получится объяснить как узнать IP камеры в DHCP.

OpenVPN Некоторые производители предлагают установку openvpn на камеры для запуска облачного сервиса. Это не лучшее решение, т.к. вам придется дважды платить за оборудование: openvpn — это очень ресурсоемкий для CPU компонент, что повлечет за собой установку отдельного проксирующего сервера максимум на 300-400 камер, и установку к нему второго сервер для потокового видео, а это удвоение затрат на инфраструктуру. Также openvpn не предоставит легкого способа балансировки пользователей между серверами (Streamers), как это возможно при использовании агента.

Flussonic Agent Flussonic Agent — лучше, чем любое из приведенных выше решений, поскольку он не требует настройки и позволяют подключать камеру непосредственно к Flussonic Media Server.

О том, как установить и использовать Агент, читайте в документации Flussonic Watcher.

11.1.2 Flussonic Iris

Flussonic Iris — это прошивка для IP-камер, разработанная Flussonic для удобства пользователя.

Основные преимущества Iris:

- Дает возможность подключать Wi-Fi камеру к сети Wi-Fi и добавлять в Watcher с помощью QR-кода из мобильного приложения Watcher, как описано в инструкции по добавлению камер. Именно поэтому прошивка Iris уже установлена на наших Wi-Fi камерах.
- Позволяет использовать на камере все функции Агента, включая TLS-шифрование передаваемых данных.
- Предоставляет удобный веб-интерфейс для настройки основных параметров камеры в любом современном веб-браузере с любым количеством одновременных подключений к камере.

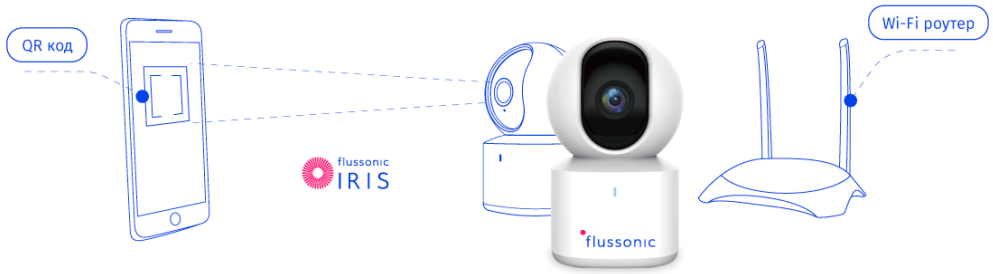


Figure 11.2: Иллюстрация преимуществ Iris