Корпоративное телевидение Agora

Корпоративное телевидение Agora

Максим Лапшин

Table of contents

1. Продукты

2. Agora

2.0.1 Agora

Agora - это система корпоративного телевидения, позволяющая:

- организовывать собственные телеканалы;
- готовить к вещанию трансляции со студийных камер;
- распространять телевизионный сигнал по закрытой сети предприятия и при необходимости через интернет;
- защищенным образом доставлять видео до сотрудников компании.

Основные возможности

- Прием live-трансляций и файлового контента
- Обработка и подготовка видеопотоков к распространению
- Организация каналов внутреннего вещания
- Запись трансляций и формирование VOD-каталога
- Управление пользователями и аудиториями
- Интеграция с корпоративными системами аутентификации (OIDC / OAuth2 / LDAP / AD / SSO)
- Доставка контента на Smart TV, TV-приставки (STB), web-клиент и мобильные устройства
- Администрирование через web-интерфейс
- API для интеграции с внутренними ИТ-системами

Навигация по документации

Эта документация состоит из следующих разделов:

- Обзор архитектуры
- Архитектурные паттерны
  - Cluster Ingest
  - Twincast
  - Double Publish
  - Standby Push
- Управление контентом
  - Обзор управления контентом
  - Управление потоками
  - Захват источника
  - Обработка потока
  - Исходящие публикации
- Руководство администратора
  - Устройства ввода/вывода
  - Стримеры
  - CDN
  - Мониторинг
- Руководство по безопасности
  - Пользователи и роли
  - OIDC/OAuth2
  - Журнал безопасности
  - Сессии пользователей
  - Сетевые взаимодействия компонент

2.1 Architecture

2.1.1 Архитектура Agora

Agora строится как модульная on-prem платформа корпоративного телевидения. Архитектура системы разделяет контур управления, контур обработки видеопотоков и контур доставки контента, чтобы независимо масштабировать администрирование, прием, транскодирование, архив и просмотр.

Архитектурные принципы

При проектировании Agora используются следующие принципы:

- все ключевые компоненты могут размещаться во внутреннем контуре заказчика;
- управляющие сервисы отделены от серверов видеотрафика;
- роли `Ingress сервер`, `origin`, `vod transcoder`, `edge сервера` и `storage система` могут совмещаться или разноситься по разным узлам в зависимости от масштаба внедрения;
- для критичных потоков предусматривается резервирование источников, серверов и маршрутов доставки;
- система может начинаться с компактной инсталляции и разворачиваться в распределенную схему без смены модели управления.

Логическая схема

В типовой инсталляции система включает следующие логические блоки:

- `Контроллер` для централизованного управления и мониторинга;
- `Ingress сервер`, который размещается в DMZ и принимает видео от внешних источников;
- `origin`, который принимает видео от `Ingress сервер`, студийного вещания, HDMI/SDI и других внутренних источников и при необходимости выполняет транскодирование;
- `vod transcoder` для подготовки роликов и архивной нарезки к публикации;
- `edge сервера`, которые организуют CDN по закрытой сети предприятия;
- `storage система` для долгосрочного хранения роликов и материалов, полученных из архивной нарезки;
- `player` для просмотра видео конечными пользователями.

Упрощенно взаимодействие выглядит так:

```
flowchart LR
    central["контроллер"] --> config["Config / Audit / Monitoring"]
    central --> ingress["Ingress сервер"]
    central --> origin["origin"]
    central --> vod["vod transcoder"]
    central --> edge["edge сервера"]

    external["External sources"] --> ingress
    ingress --> origin
    studio["Studio / HDMI / SDI / internal IP sources"] --> origin

    origin --> edge
    origin --> storage["storage система"]

    vod --> storage
    vod --> edge

    edge --> player["player"]

    player -.- telemetry -.-> central
```

Модель поставки

Поставка Agora является гибридной: часть компонентов доступны в виде программного обеспечения, а часть поставляются в виде ПАКов (программно-аппаратных комплексов).

В виде ПО обычно поставляются те компоненты, которые могут быть развернуты на стандартной серверной инфраструктуре заказчика:

- `контроллер`;
- `origin`;
- `vod transcoder`;
- `edge сервера`;
- `storage система`.

В виде ПАК поставляются те компоненты, для которых важна заранее подготовленная аппаратная конфигурация, проверенная совместимость и гарантированные характеристики ввода-вывода.

В случае, когда необходимы специализированные платы захвата, например `HDMI` или `SDI`, поставка осуществляется в виде ПАК.

Слой управления

Плоскость управления отвечает за конфигурацию системы, безопасность и наблюдаемость.

КОНТРОЛЛЕР

`Контроллер` является центральным компонентом управления системой. Он используется администраторами и операторами для:

- настройки узлов системы;
- регистрации и контроля всех серверных ролей;
- создания и изменения потоков;
- мониторинга статусов, метрик и событий;
- управления политиками записи, доставки и защиты;
- просмотра аудита и результатов health-check.

С точки зрения архитектуры `контроллер`:

- хранит логическую модель потоков и серверов;
- предоставляет единый интерфейс управления;
- управляет учетными записями, ролями и сессиями;
- собирает и агрегирует статус серверов и потоков;
- выступает центральной точкой мониторинга и контроля конфигурации.

БАЗА ДАННЫХ

Отдельная база данных используется для:

- конфигурации потоков;
- описания серверов и входных устройств;
- данных учетных записей и ролей;
- хранения сессий;
- журнала административных и security-событий.

Это хранилище не предназначено для самих видеоданных. Видеоархивы, DVR и VOD-файлы выносятся в специализированные контуры хранения.

Слой обработки видео

INGRESS СЕРВЕР

`Ingress сервер` размещается в DMZ и предназначен для безопасного приема видеосигнала от внешних источников. Он используется как пограничная точка входа для входящего видеотрафика.

`Ingress сервер` может принимать:

- IP-потоки по `SRT`, `RTMP` и другим поддерживаемым протоколам;
- сигналы от внешних площадок и подрядчиков;
- резервные внешние источники.

Основные задачи `Ingress сервер`:

- принять внешний поток в DMZ;
- отделить внешний контур от внутренних медиасерверов;
- передать видео дальше на `origin`;
- обеспечить контролируемую и наблюдаемую точку входа.

Для критичных каналов может применяться резервирование входа, когда один и тот же внешний эфир принимается несколькими независимыми трактами.

ORIGIN

Роль `origin` в том, чтобы подготовить поток для дальнейшей раздачи по сотрудникам и телевизорам.

Он:

- принимает поток от `Ingress` сервера;
- принимает видео от студийного вещания;
- принимает HDMI/SDI и другие локальные источники;
- выполняет транскодирование при необходимости;
- формирует мастер-поток для дальнейшей доставки;
- передает контент на `edge` сервера;
- может сам проигрывать клиентам, если нагрузка невелика;
- передает архивные и производные материалы в `storage` систему.

В архитектуре Agora `origin` совмещает роль master-узла, точки агрегации внутренних и внешних источников, а также роль узла транскодирования.

АРХИВ И ПОДГОТОВКА МАТЕРИАЛОВ

Подсистема записи обеспечивает:

- live DVR с ограниченным окном хранения;
- сохранение эфира после завершения трансляции;
- передачу материалов в `storage` систему и публикацию записей как VOD-контента.

Временный DVR-буфер и краткоживущий архив обычно располагаются рядом с `origin`, а долгосрочное хранение выносится в `storage` систему.

VOD TRANSCODER

`vod transcoder` используется для подготовки роликов, записей эфира и архивной нарезки к публикации. Он:

- получает исходные материалы снаружи от `origin` или из `storage система`;
- транскодирует ролики в нужное количество выходных профилей;
- выполняет нормализацию контейнеров, кодеков и битрейтов;
- передает подготовленные материалы в `storage`, откуда уже могут забирать `edge` сервера.

Слой доставки

EDGE СЕРВЕРА

`edge` сервера отвечают за доставку видео конечным потребителям по закрытой сети предприятия. Они могут размещаться:

- в пользовательском сегменте локальной площадки;
- в филиале;
- рядом с крупной группой зрителей для локального offload.

Назначение `edge сервера`:

- разгрузить `origin`;
- сократить межсегментный трафик;
- локализовать клиентские подключения;
- обеспечить отказоустойчивую доставку при недоступности части узлов.

В зависимости от требований заказчика `edge` сервера могут работать как:

- сервер ретрансляции в филиалы;
- точка выдачи multicast или `UDP MPEG-TS` в локальную сеть.

PLAYER

`player` используется для просмотра контента конечными пользователями. Мы предоставляем:

- веб-плеер, пригодный для встраивания в корпоративный портал;
- Android приложение, которое можно поставить на приставку, в том числе поставляем и собранные в ПАК приставки;
- веб-приложение, которое можно открыть на телевизоре.

Контент в `player` может доставляться:

- с `origin` сервера
- через `edge` сервера;.

Хранение контента

Система хранения разделяется на два класса:

`storage` система предназначена для долгосрочного хранения видео как в виде отдельных роликов, так и в виде нарезки из сохраненного архива.

Оперативное хранение включает:

- временный live DVR;
- краткоживущие архивы на `origin`;
- промежуточные файлы транскодирования.

Долговременное хранение включает:

- записи эфиров;
- загруженные видеоролики;
- подготовленные VOD-материалы.

Такое разделение позволяет отдельно выбирать:

- производительные диски для live-задач;
- более емкие и экономичные хранилища для архива;
- разные политики резервного копирования и retention.

Типовые потоки данных

ПРЯМАЯ ТРАНСЛЯЦИЯ

Типичный live-сценарий выглядит так:

1. Внешний источник передает сигнал на `Ingress сервер`, либо внутренний источник поступает сразу на `origin`.
2. `origin` принимает поток, при необходимости транскодирует его и формирует мастер-представление.
3. `origin` передает контент на `edge сервера`.
4. `edge сервера` раздают поток конечным зрителям.
5. Параллельно поток может записываться в `DVR` и архив.

ПУБЛИКАЦИЯ ЗАПИСИ

Сценарий работы с записью обычно включает:

1. Сохранение эфира во временный архив.
2. Перемещение записи в `storage система`.
3. Подготовку выходных профилей и метаданных в `vod transcoder`.
4. Публикацию VOD через `origin` и `edge сервера`.

Варианты размещения

КОМПАКТНАЯ ИНСТАЛЛЯЦИЯ

Для небольших площадок несколько ролей могут быть совмещены на одном или двух серверах:

- `Контроллер`;
- `origin`;
- `vod transcoder`;
- `storage система`.

Такой вариант подходит для пилотов и ограниченного числа каналов и зрителей.

РАСПРЕДЕЛЕННАЯ ИНСТАЛЛЯЦИЯ

Для корпоративной эксплуатации рекомендуется разносить роли:

- отдельный `Контроллер`;
- отдельный `Ingress сервер` в DMZ;
- один или несколько `origin`;
- отдельный `vod transcoder` для подготовки роликов;
- отдельные `edge сервера` в пользовательских сегментах или филиалах;
- отдельная `storage система`.

При наличии требований по сетевой изоляции внешние источники взаимодействуют только с `Ingress сервер`, а пользовательский просмотр идет через `edge сервера`.

Отказоустойчивость

Требования к доступности у разных узлов различаются:

- управляющий сервер критичен для администрирования, но не всегда влияет на уже идущий просмотр;
- `Ingress сервер` и `origin` критичны для live-вещания внешних источников;
- `edge сервера` критичны для массовой доставки;
- `vod transcoder` критичен для подготовки VOD-контента и архивной нарезки;
- `storage система` критична для VOD и архивов, но не обязательно для каждого live-сценария.

Основные механизмы отказоустойчивости доставки видео:

- Cluster Ingest
- Twincast
- Double Publish
- Standby Push
- резервирование входа через основной/резервный источник;
- парное размещение `Ingress сервер` и `origin` для критичных трактов;
- несколько `edge сервера` с балансировкой и fallback;
- разделение оперативного и архивного хранения;
- резервное копирование конфигурации, учетных записей и аудита;
- мониторинг доступности серверов, времени, конфигурации и качества потоков.

Безопасность и сетевые границы

В корпоративной инсталляции архитектура обычно делится на сегменты:

- административный сегмент;
- внутренний сегмент медиасерверов;
- сегмент хранения;
- пользовательский сегмент или DMZ для доставки.

Это позволяет:

- ограничить прямой доступ пользователей к внутренним мастер-узлам;
- отделить web-управление от видеотрафика;
- вынести прием внешнего видео на отдельный `Ingress сервер` в DMZ;
- централизовать аудит и контроль изменений;
- упростить интеграцию с корпоративными средствами ИБ.

Границы с внешними системами

С точки зрения архитектуры Agora взаимодействует со следующими внешними сущностями:

- внешние источники видео;
- корпоративные каталоги пользователей и роли доступа;
- системы мониторинга и SIEM;
- внешние медиаплееры и IPTV-устройства;
- корпоративные порталы и `player`;
- внешние системы публикации и production-инструменты.

Итог

Agora представляет собой многоуровневую платформу, в которой `контроллер`, `Ingress сервер`, `origin`, `vod transcoder`, `edge сервера`, `storage система` и `player` выполняют четко разделенные роли. Это дает возможность строить как компактные инсталляции для одной площадки, так и отказоустойчивые распределенные внедрения с централизованным управлением, защищенным приемом внешнего видео, внутренним master-слоем обработки, отдельной подготовкой VOD-материалов и масштабируемой доставкой по закрытой сети предприятия.

2.1.2 Twincast

`Twincast` - это сценарий отказоустойчивости, при котором один и тот же эфир одновременно принимается двумя независимыми трактами: основным и резервным. Это позволяет платформе контролировать оба канала доставки сигнала и использовать резервный тракт при деградации основного.

Как работает Twincast

В типовой конфигурации один источник подается сразу на два независимых ingest-тракта, после чего система сравнивает их состояние и использует рабочий поток.

```
flowchart LR
    source["Source"] --> primary["Primary ingest"]
    source --> backup["Backup ingest"]
    primary --> origin["origin"]
    backup --> origin
```

При нормальной работе основным считается `primary` тракт. Если на нем возникает ошибка, система может переключиться на `backup`, сохраняя непрерывность вещания.

При этом у `Twincast` есть важное ограничение: `primary` и `backup` принимают сигнал независимо друг от друга, поэтому таймстемпы кадров и `GOP`-структура видео между ними не синхронизированы. Из-за этого переключение между `primary` и `backup` получается менее стабильным, чем в сценариях, где оба тракта заранее согласованы по структуре потока.

Когда применять

`Twincast` особенно полезен в следующих сценариях:

- студийное вещание и критичные live-каналы;
- прием сигнала от важного внешнего источника;
- площадки, где простой даже на несколько секунд нежелателен;
- каналы, для которых требуется постоянный контроль качества основного и резервного входа.

Что дает этот сценарий

Преимущества `Twincast`:

- резервирование не только сервера, но и всего тракта приема;
- постоянное наблюдение за состоянием `primary` и `backup`;
- быстрое переключение на рабочий тракт;
- снижение риска прерывания эфира при частичном отказе оборудования или сигнала.

Ограничения и требования

Для корректной реализации `Twincast` нужно учитывать:

- наличие двух независимых трактов приема;
- корректную идентификацию основного и резервного сигнала;
- мониторинг качества и статуса обоих трактов;
- отсутствие синхронизации таймстемпов кадров и `GOP`-структуры между `primary` и `backup`;
- правила автоматического или ручного переключения между ними.

2.1.3 Double Publish

`Double Publish` - это подход, при котором оригинальный источник сигнала отправляет поток сразу на два стримера. При этом стримеры настроены так, чтобы ретрансляторы всегда могли получить одинаковое видео с любого из них.

Как работает Double Publish

Оригинальный источник сигнала отправляет поток сразу на два стримера. При этом второй стример одновременно:

- принимает поток напрямую от источника;
- забирает поток с первого стримера;
- использует поток с первого стримера как приоритетный.

Это нужно для того, чтобы на обоих стримерах сохранялись одинаковые таймстемпы, одинаковая структура видео и одинаковое представление потока для дальнейшей публикации.

```
flowchart LR
    source["Original source"] -- publish --> streamerA["Streamer A"]
    source -- publish --> streamerB["Streamer B"]
    streamerA --> streamerB
    streamerA --> relayA["ретрансляторы / downstream"]
    streamerB --> relayB["ретрансляторы / downstream"]
```

Пока поток с первого стримера доступен, именно он является приоритетным источником для второго стримера. Если поток с первого стримера прерывается, второй стример немедленно переключается на собственный прямой прием от оригинального источника и продолжает публикацию без ожидания ручного вмешательства.

Когда применять

`Double Publish` рекомендуется использовать, если:

- источник может сразу отправлять сигнал на два стримера;
- нужно получить одинаковый поток на двух publish-стримерах;
- downstream-ретрансляторы должны иметь возможность брать видео с любого из стримеров;
- важно обеспечить быстрое переключение без расхождения по таймстемпам и структуре видео.

Такой режим особенно удобен при работе с программными энкодерами. Для `OBS`, `vMix` и аналогичных программ `Double Publish` является стандартной и естественной конфигурацией, поскольку источник изначально умеет отправлять поток сразу на несколько адресов.

Что дает этот сценарий

Преимущества `Double Publish`:

- отказоустойчивая публикация с двух стримеров;
- синхронизация структуры видео и таймстемпов между стримерами;
- немедленное переключение на публикацию со второго стримера при отказе потока с первого;
- возможность для последующих ретрансляторов брать видео с любого стримера без различий в содержимом потока.

Ограничения и требования

При проектировании `Double Publish` важно учитывать:

- источник должен поддерживать одновременную отправку потока на два стримера;
- второй стример должен уметь одновременно держать прямой прием и прием с первого стримера;
- необходимо корректно задать приоритет потока с первого стримера;
- требуется мониторинг момента переключения со связанного потока на прямой прием;
- downstream-ретрансляторы должны быть готовы подключаться к любому из двух стримеров.

2.1.4 Standby Push

`Standby Push` - это сценарий отказоустойчивости, при котором для основного publish-направления заранее подготовлен резервный push-контур. Пока основной путь работает нормально, резервный находится в режиме ожидания и включается только при отказе или деградации основного.

Как работает Standby Push

В этом сценарии `origin` или другой publishing-узел имеет два направления отправки потока:

- основной push;
- резервный push в режиме standby.

Наиболее частый вариант настройки - использование `Standby Push` вместе с `Twincast`. В такой связке основной и резервный тракт уже существуют на уровне приема, а `Standby Push` позволяет быстро переключать дальнейшую публикацию между ними.

Типовая схема:

```
flowchart LR
    primaryStreamer["Primary streamer"] --> multicast["multicast"]
    multicast --> backupStreamer["Backup streamer"]
    primaryStreamer --> primary["Primary push"]
    primary --> edge["edge сервера"]
    backupStreamer --> standby["Standby push"]
    standby --> reserve["резервный edge контур"]
```

Пока основной push работает штатно, резервный контур не участвует в активной доставке или поддерживается в пассивном режиме. При отказе основного направления система активирует standby-контур.

Характерное время переключения в такой схеме составляет менее секунды. Это делает связку `Twincast` + `Standby Push` самым быстрым вариантом переключения, но она подходит только для локальной сети, где можно гарантировать низкую задержку.

Для работы `Standby Push` backup-стример должен получать multicast с первого стримера.

Когда применять

`Standby Push` подходит для следующих сценариев:

- нужен резервный путь публикации без постоянной двойной нагрузки;
- отказоустойчивость требуется именно на исходящем push;
- важно сократить лишний трафик по сравнению с `Double Publish`;
- резервный контур должен быть подготовлен заранее, но использоваться только при аварии;
- используется связка с `Twincast` для максимально быстрого переключения;
- нужно сохранить старое оборудование в роли резервного, не меняя его конфигурацию и не вмешиваясь в его работу.

Что дает этот сценарий

Преимущества `Standby Push`:

- резервирование publish-контура без постоянной активной отправки в оба направления;
- более экономное использование пропускной способности по сравнению с `Double Publish`;
- быстрое включение резервного пути;
- повышение устойчивости доставки при отказе основного направления;
- возможность использовать старое оборудование как backup-контур, не меняя его роль в системе.

Ограничения и требования

Для корректной реализации `Standby Push` нужно определить:

- критерии перехода с основного push на резервный;
- время активации standby-контура;
- правила возврата на основной путь после восстановления;
- мониторинг основного и резервного publish-направления;
- наличие локальной сети с гарантированно низкой задержкой между primary и backup;
- доставку multicast с первого стримера на backup-стример.

2.1.5 Cluster Ingest

`Cluster Ingest` - это механизм отказоустойчивого захвата, который позволяет выдержать гарантию единоразового получения потока с источника. Он нужен в тех сценариях, где надо обеспечить надежный прием видео, но каждое подключение к источнику дорого, нежелательно или вообще невозможно дублировать.

Как работает Cluster Ingest

В этом сценарии поток с внешнего источника в каждый момент времени захватывает только один стример. Остальные стримеры находятся в резерве и не создают повторное подключение к источнику.

Типовая схема выглядит так:

```
flowchart LR
    source["Source"] --> primary["active origin"]
    primary --> edge["edge сервер"]
    edge --> player["player"]
    backup -.-> edge

    central["контроллер"] -- controls --> primary["primary streamer"]
    central -- controls --> backup["backup streamer"]
```

В `Agora` этот сценарий реализуется управляющим сервером `контроллер`. Именно он следит за состоянием стримеров и управляет переключением захвата. Если первичный стример недоступен или неработоспособен, `контроллер` отдает команду резервному стримеру на включение захвата.

Таким образом, отказоустойчивость достигается не за счет параллельного двойного приема, а за счет управляемого переключения активного захвата между стримерами.

Когда применять

`Cluster Ingest` имеет смысл использовать в следующих случаях:

- каждое получение канала с источника является дорогим;
- повторное подключение к источнику нежелательно;
- пропускной способности канала недостаточно для двойной доставки одного и того же потока;
- требуется отказоустойчивость именно на этапе захвата, а не за счет дублирования входа.

Типичный пример - передача видеопотока из головного офиса в региональный. В такой схеме повторная передача того же канала может быть слишком дорогой по полосе пропускания или вообще невозможной.

Что дает этот сценарий

Основные преимущества `Cluster Ingest`:

- отказоустойчивость на уровне захвата;
- сохранение модели единоразового получения потока;
- отсутствие постоянного дублирования трафика от источника;
- возможность переключить захват на резервный стример при отказе первичного.

Ограничения и требования

При проектировании `Cluster Ingest` нужно учитывать:

- корректный health-check стримеров со стороны `контроллера`;
- время обнаружения отказа первичного стримера;
- время переключения захвата на резервный стример;
- требования источника к повторному подключению после сбоя;
- синхронность конфигурации между primary и backup стримерами.

2.2 Admin

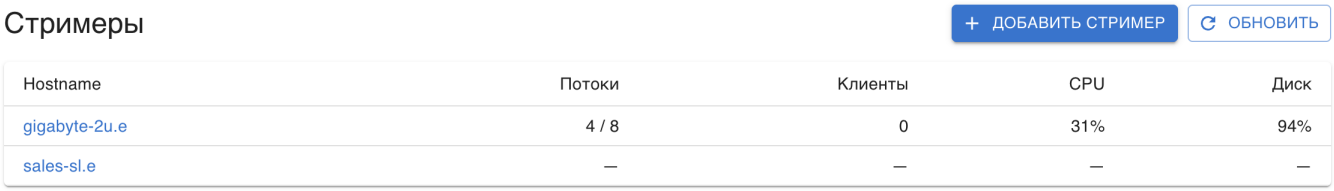
2.2.1 Управление стримерами

Стримеры в Agora являются исполнительными узлами, на которых выполняются прием сигнала, публикация, работа с аппаратными устройствами и сбор эксплуатационной статистики. Управление стримерами нужно для того, чтобы подключить реальные серверы к платформе, контролировать их состояние и использовать их в потоках и `IO`-устройствах.

Через раздел стримеров администратор:

- регистрирует серверы `origin` и `edge` в системе;
- задает параметры подключения к ним;
- контролирует их загрузку и текущее состояние;
- подготавливает инфраструктуру для настройки потоков и `IO-устройств`.

Список стримеров



На странице списка стримеров отображаются все зарегистрированные узлы. Для каждого стримера видны:

- `hostname`;
- количество работающих потоков и общее количество сконфигурированных потоков;
- количество подключенных клиентов;
- загрузка CPU;
- использование диска.

Из списка можно:

- открыть карточку конкретного стримера;
- добавить новый стример;
- обновить список вручную.

Этот экран используется как быстрый обзор состояния инфраструктурных узлов Agora.

Создание стримера

При создании нового стримера в текущем интерфейсе достаточно указать:

- `hostname`.

После создания стример получает внутренний идентификатор и становится доступен для дальнейшей настройки.

Карточка стримера

В карточке стримера администратор может:

- изменить `hostname`;
- выбрать схему подключения: `HTTP` или `HTTPS`;
- указать API-порт;
- настроить `config_api_key`;
- настроить `edit_auth`;
- сохранить изменения;
- перезагрузить данные стримера;
- удалить стример.

Если на странице есть несохраненные изменения, интерфейс предупреждает об этом при попытке уйти со страницы.

Параметры подключения

Для работы с удаленным узлом задаются:

- схема подключения `HTTP` / `HTTPS`;
- порт API;
- `config_api_key` для доступа к внешней конфигурации;
- `edit_auth` для операций редактирования.

Эти параметры определяют, как `контроллер` взаимодействует со стримером.

`config_api_key` нужен для того, чтобы на стримере можно было указать `config_external` и получить оттуда полную конфигурацию сервера. Иными словами, этот ключ используется как пароль для централизованной конфигурации

Работа с секретами

В интерфейсе для секретных параметров доступны дополнительные действия:

- скрытие и отображение значения;
- копирование `config_api_key`;
- генерация нового `config_api_key`.

Это упрощает первичную настройку и ротацию ключей доступа.

Получение команды для внешней конфигурации

Для уже созданного стримера интерфейс может показать готовую `curl`-команду для обращения к endpoint внешней конфигурации.

Это помогает:

- проверить какая конфигурация будет доступна стримеру;
- быстро проверить правильность URL для конфигурации;
- убедиться, что `config_api_key` настроен корректно.

Связь с другими разделами

Настроенные стримеры используются в других частях Agora:

- при привязке `IO`-устройств;
- при настройке аппаратного захвата `HDMI` / `SDI`;
- при проектировании кластерных режимов приема и публикации;
- при мониторинге состояния инфраструктуры.

2.2.2 Внутрикорпоративная CDN

Внутрикорпоративная `CDN` в Agora используется для доставки потока в удаленные офисы и сегменты корпоративной сети, где прямое вещание с центрального origin-сервера создает лишнюю нагрузку на медленные или дорогие каналы связи.

В этом режиме Agora направляет зрителя не сразу на центральный origin, а на подходящий рестример внутри его сетевой зоны. Благодаря этому один и тот же поток доставляется в удаленный офис один раз, а дальше раздается локально через рестример.

Как работает CDN в Agora

В схеме внутрикорпоративной `CDN` используются следующие сущности:

- `origin`-стримеры - исходные узлы, на которых поток принимается или формируется в Agora;
- `restreamer` - узел раздачи, который получает поток от origin и обслуживает зрителей внутри своей зоны;
- `zone` - логическая зона корпоративной сети, обычно удаленный офис;
- `CIDR`-маршруты - правила, по которым IP-адрес зрителя сопоставляется с зоной;
- `fallback zone` - резервная зона, которая используется, если в основной зоне нет доступных рестримеров.

Зритель открывает ссылку на поток в браузере. Agora определяет IP-адрес клиента, находит подходящую зону, выбирает в ней доступный рестример с минимальной текущей нагрузкой и отдает клиенту адрес воспроизведения.

Если в зоне нет доступных рестримеров, Agora использует резервную зону. Если IP-адрес клиента не попал ни в одно правило маршрутизации, CDN-маршрутизация для этого клиента не применяется.

Когда администратору нужно настраивать CDN

`CDN` настраивают в тех случаях, когда:

- сотрудников много и один `origin` не справится с раздачей
- сотрудники смотрят одни и те же live-потоки из удаленных филиалов;
- между центральной площадкой и филиалом есть медленный приватный канал;
- нужно уменьшить количество одновременных подключений к центральному origin;
- необходимо закрепить определенные диапазоны IP-адресов за локальными узлами раздачи.

Что нужно подготовить заранее

Перед настройкой `CDN` администратор подготавливает:

- хотя бы один рабочий `origin`-стример, на котором уже доступен поток;
- хотя бы один сервер, который будет зарегистрирован как `restreamer`;
- сетевые диапазоны филиалов или офисов в формате `CIDR`, например `10.20.30.0/24`;
- внешний адрес раздачи для каждого рестримера, который будет использоваться зрителем, например `https://edge-1.office.example`;
- при необходимости резервную зону для аварийного переключения.

Перед настройкой `CDN` надо убедиться, что origin-стримеры уже видны в разделе Стримеры, находятся в рабочем состоянии.

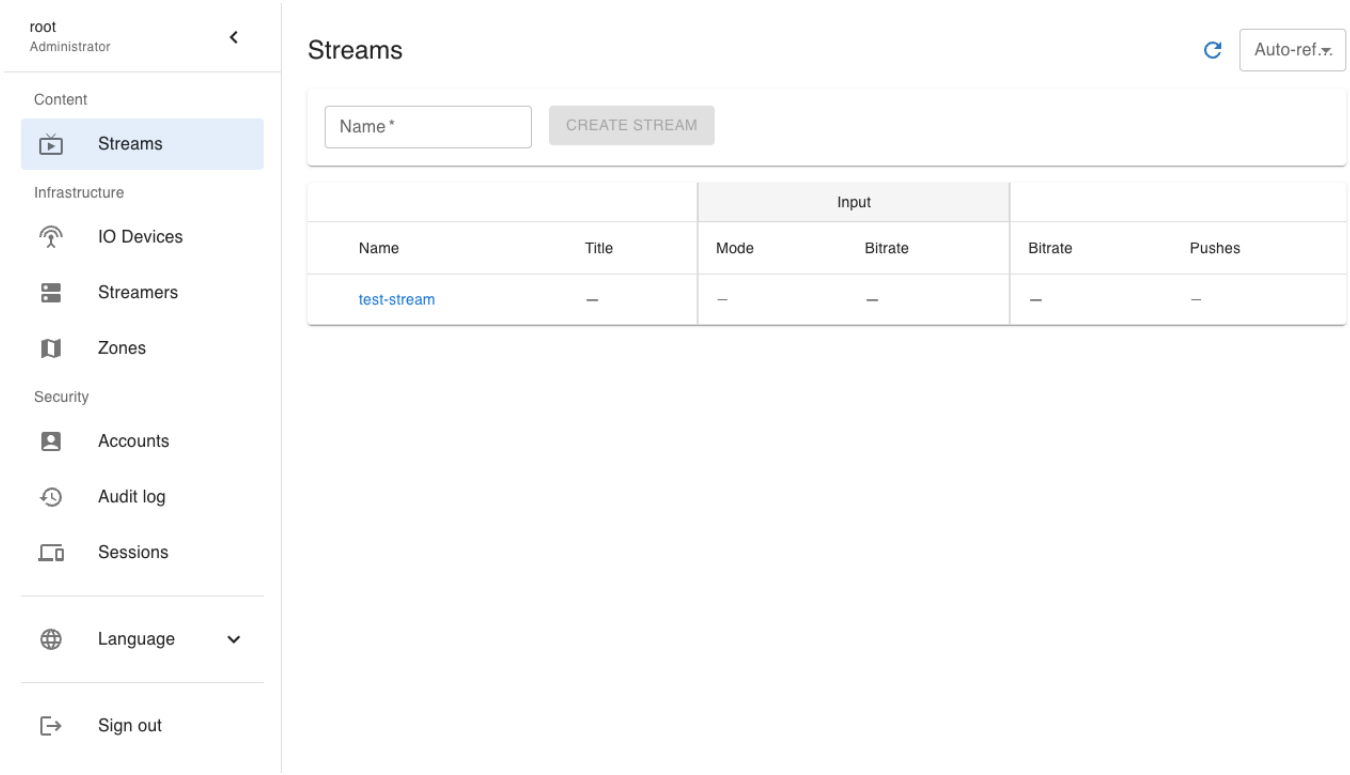
Порядок настройки CDN

Рекомендуемый порядок настройки:

1. Настроить origin-стримеры.
2. Создать зоны.
3. Настроить `CIDR`-маршруты зон.
4. Указать резервные зоны при необходимости.
5. Настроить рестримеры.
6. Назначить рестримеры в зоны.
7. Проверить выдачу конфигурации рестримерам.
8. Проверить viewer portal из нужной сети.

Именно такой порядок уменьшает количество ошибок, связанных с пустыми зонами, незаполненными адресами раздачи и неправильной маршрутизацией клиентов.

Раздел Streams: список потоков



Потоки создаются и редактируются в разделе **Streams** левого меню. Для сценария CDN на origin должен быть настроен хотя бы один поток, который зрители будут открывать через viewer portal и balancer. Подробнее о настройке потоков см. в разделе Потоки.

Создание зоны

Зона описывает отдельный офис, филиал или сетевой сегмент, где зрители должны обслуживаться через локальный рестример.

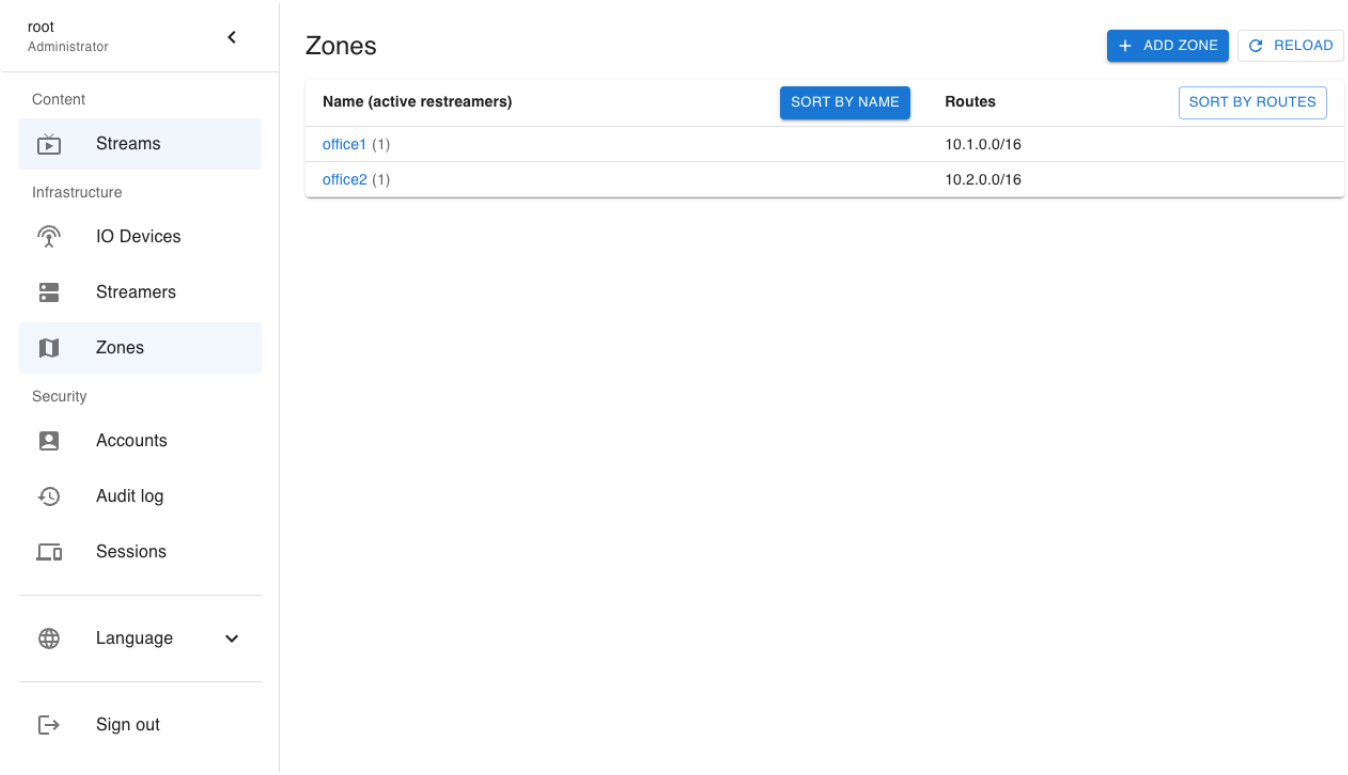
В карточке зоны администратор задает:

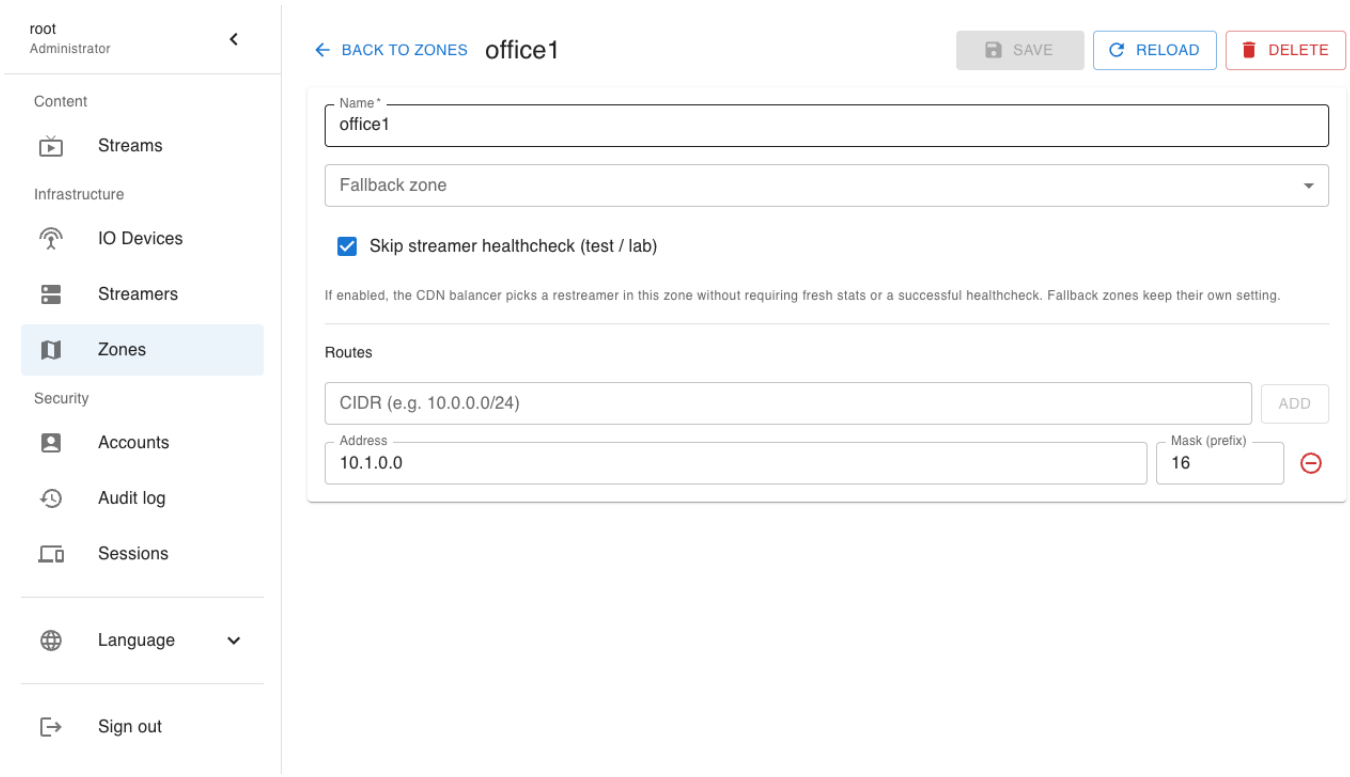
- `name`;
- `fallback zone`, если нужен резерв;
- список `CIDR`-маршрутов;
- опцию «**Пропускать проверку живости стримера**» (`skip_streamer_healthcheck` в API): если она включена, встроенный balancer при маршрутизации **в эту зону** выбирает рестример без требований к свежей статистике и успешному healthcheck на рёбре. Отключённые (`disabled`) рестримеры по-прежнему не участвуют. У резервной зоны своё значение этой опции. Имеет смысл только для лабораторных проверок CDN, без продакшена с реальными зрителями.

Название зоны должно быть уникальным и понятным, например:

- `msk-office-1`;
- `spb-office`;
- `plant-a`;
- `vpn-users`.

Резервная зона используется, если в основной зоне нет ни одного доступного рестримера.





На скриншоте карточки показан пример зоны с сохранённым `CIDR`, без резервной зоны и с включённой опцией «**Пропускать проверку живости стримера**» для лабораторной проверки.

Настройка CIDR-маршрутов

Маршруты в зоне задаются в формате `CIDR`.

В admin API каждый маршрут передаётся полями `address` (строка — IPv4-адрес сети в **dot notation**, например `10.0.0.0`) и `mask` (целое число — длина префикса, как в CIDR: `24` для `/24`). Внутреннее хранение в базе может отличаться; на границе API используется именно строка с точками между октетами. Запись вида `10.0.0.0/24` в интерфейсе админки можно разобрать на эту пару полей.

Примеры корректных маршрутов:

- `10.12.13.0/24`;
- `10.12.0.0/16`;
- `192.168.100.0/24`;
- `0.0.0.0/0`.

Agora сопоставляет IP-адрес зрителя с маршрутами всех зон и выбирает правило с самой длинной маской. Это означает, что более специфичное правило имеет приоритет над более общим.

Например:

- зона `Office-A` содержит маршрут `10.12.0.0/16`;
- зона `Office-A-floor-3` содержит маршрут `10.12.13.0/24`.

Зритель с адресом `10.12.13.42` попадет в `Office-A-floor-3`, потому что маршрут `/24` точнее, чем `/16`.

Если один и тот же `CIDR` уже используется в другой зоне, его нужно перенести или удалить из старой зоны. Один и тот же маршрут не должен одновременно принадлежать нескольким зонам.

Маршрут `0.0.0.0/0` отвечает за дефолтный роутинг. Если его не указать, то клиент, который не попал ни в один маршрут, окажется без выданного рестримера и без проигрывания.

Настройка резервной зоны

Резервную зону используют в тех случаях, когда в основной зоне все рестримеры недоступны или отключены.

При настройке резервирования рекомендуется:

- выбирать резервную зону, до которой у офиса действительно есть рабочая связность;
- не создавать циклические цепочки резервирования;
- использовать простую и понятную схему резервирования, чтобы ее было легко сопровождать.

Типичный пример:

- зона `office-ekb` имеет основной локальный рестример;
- зона `office-msk` назначена как `fallback zone` для `office-ekb`.

Если локальный рестример в `office-ekb` недоступен, зрители из этой зоны начинают получать поток через `office-msk`.

Настройка рестримера

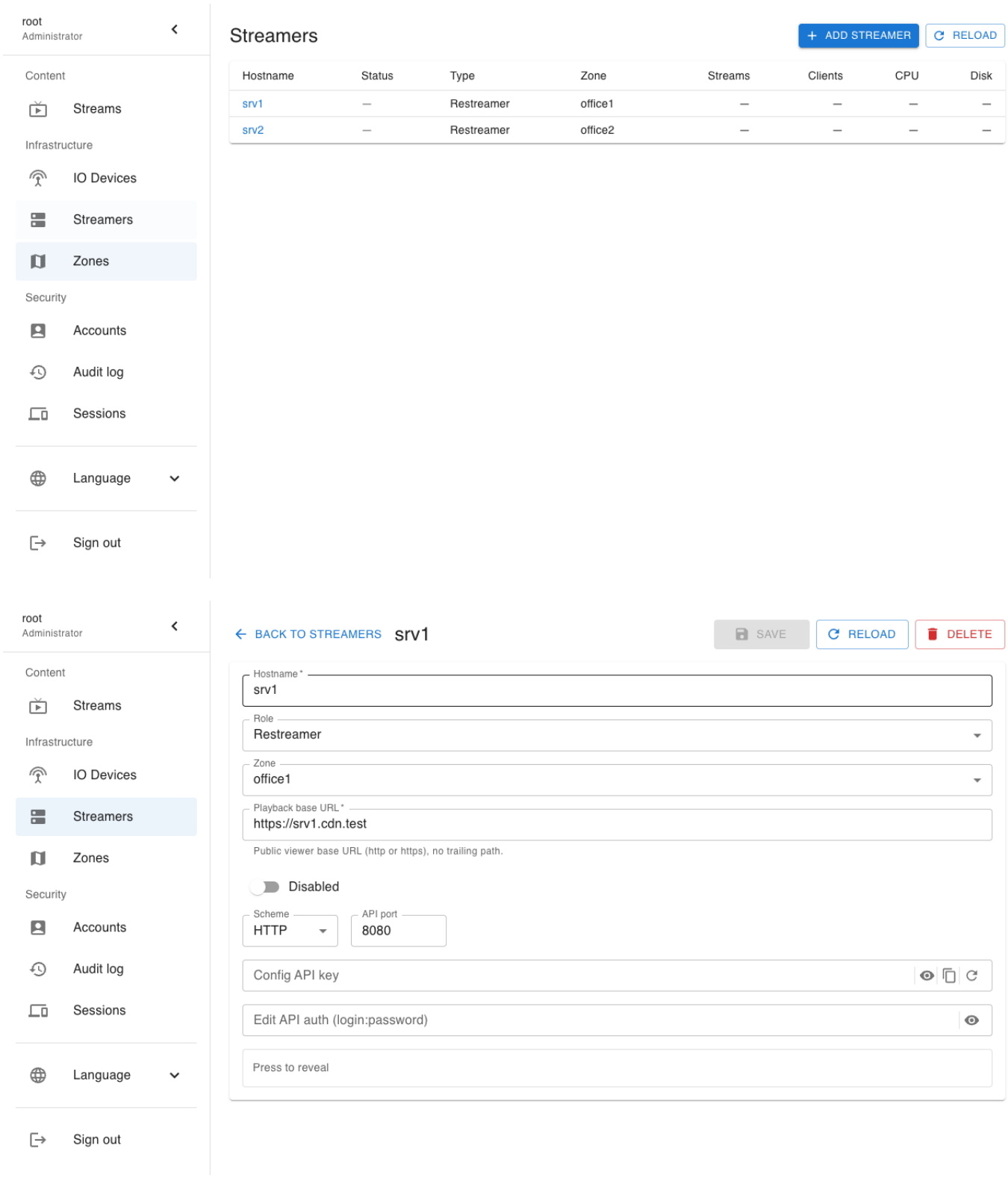
Рестример создается в том же разделе стримеров, что и обычный стример, но для него выбирается тип `restreamer`.

В карточке рестримера администратор заполняет:

- `hostname`;
- тип узла `restreamer`;
- схему подключения `HTTP` или `HTTPS`;
- API-порт;
- `zone`;
- `playback_base_url`.

Поле `playback_base_url` задает адрес, который Agora возвращает зрителям для воспроизведения потока. Обычно это внешний URL рестримера в корпоративной сети или в пользовательском сегменте, например `https://edge-1.office.example`.

Для рестримера привязка к зоне обязательна. Один рестример может относиться только к одной зоне.



Что получает рестример от Agora

Рестример получает конфигурацию через `config_external`, как и другие стримеры, но в `CDN`-сценарии Agora выдает ему on-demand-конфигурацию для раздачи потоков.

Это означает следующее:

- поток на рестримере не поднимается заранее без запроса зрителя;
- рестример получает вход от origin в формате `m4s`;
- публикация на рестримере запускается по первому обращению клиента.

Такой режим уменьшает постоянную нагрузку на рестример и не заставляет держать активными все возможные потоки одновременно.

Как Agora выбирает рестример для зрителя

При обращении зрителя к потоку Agora выполняет следующие действия:

1. Определяет IP-адрес клиента.
2. Ищет подходящую зону по `CIDR`.
3. Находит в зоне доступные рестримеры.
4. Исключает отключенные и недоступные узлы.
5. Выбирает рестример с минимальным количеством текущих клиентов.
6. Возвращает viewer portal адрес воспроизведения на выбранном рестримере.

В качестве доступных рассматриваются только те рестримеры, которые:

- не отключены в административном интерфейсе;
- отвечают на healthcheck как рабочие;
- имеют актуальную эксплуатационную статистику.

Если для зоны включена опция «**Пропускать проверку живости стримера**», то для шага выбора именно **в этой зоне** требования к healthcheck и к свежести статистики не применяются: в кандидаты попадают все привязанные к зоне рестримеры, кроме отключённых в админке. При переходе по цепочке `fallback` к другой зоне снова действуют её собственные настройки.

Отладка CDN: query-параметр source_ip

Для проверки маршрутизации balancer и viewer portal с рабочего места **вне** целевой корпоративной подсети в URL страницы портала можно добавить query-параметр **`source_ip`**: Agora будет считать «адресом зрителя» указанный IPv4 или IPv6 и подбирать зону по `CIDR` так, как если бы запрос пришёл с этого IP. Страница портала передаёт тот же параметр в запрос к `balancer/streams/{stream}`; при необходимости вместе с ним пробрасывается и `token` viewer API.

Типичный сценарий отладки в связке с **`skip_streamer_healthcheck`**:

- в зонах, которые хочется проверить, можно включить «**Пропускать проверку живости стримера**», чтобы не зависеть от живой статистики и healthcheck;
- в `CIDR` маршрутах этой зоны вписать правильные адреса, которые должны относиться к этой зоне;
- открыть viewer portal с `?source_ip=…` и убедиться, что в ответе balancer попадаете в нужную зону и получаете ожидаемый `playback_url`.

Так можно пошагово проверить зоны, fallback и выбор рестримера без физического нахождения в сегменте сети и без поднятого «зелёного» edge по метрикам. В бою опцию **`source_ip`** не используют: она обходит естественное сопоставление с реальным адресом клиента и предназначена для диагностики и лабораторных стендов.

Ниже — пример страницы viewer portal: в блоке отображается итоговый `playback_url`, выданный balancer. При `source_ip=10.1.0.1` (зона с маршрутом `10.1.0.0/16`) в URL виден edge первого офиса (`srv1` в примере стенда):

test-stream

```
https://srv1.cdn.test/test-stream/index.m3u8
```

При `source_ip=10.2.0.2` выбирается зона `10.2.0.0/16` и соответствующий рестример (`srv2`):

test-stream

```
https://srv2.cdn.test/test-stream/index.m3u8
```

Как администратору проверить, что CDN настроена правильно

После настройки рекомендуется проверить CDN по следующему сценарию:

1. Открыть раздел стримеров и убедиться, что нужный рестример зарегистрирован и не отключен.
2. Проверить, что у рестримера заполнены `zone` и `playback_base_url`.
3. Открыть карточку зоны и убедиться, что нужный `CIDR` сохранен корректно.
4. Проверить, что у зоны при необходимости указана резервная зона и что опция пропуска проверки живости включена только там, где это осознанно нужно для тестов.
5. Проверить, что поток существует и работает на origin.
6. Проверить, что рестример получает внешнюю конфигурацию через `config_external`.
7. Открыть viewer portal из сети, IP-адрес которой попадает в нужную зону.
8. Убедиться, что воспроизведение идет через локальный рестример, а не напрямую с origin.

Практически это удобно проверять из рабочего места в соответствующем филиале или через тестовый хост в нужном сетевом сегменте.

Что видит пользователь при корректной настройке

Для пользователя `CDN` работает прозрачно. Он открывает обычную ссылку на поток, а Agora сама:

- определяет подходящую зону;
- выбирает оптимальный рестример;
- направляет воспроизведение на локальный адрес раздачи.

Пользователю не нужно вручную выбирать офис, узел или точку вещания.

Типовые ошибки настройки

Если `CDN` не работает ожидаемым образом, администратор в первую очередь проверяет следующее:

- рестример создан как `restreamer`, а не как обычный origin;
- рестример не отключен;
- у рестримера заполнен `playback_base_url`;
- рестример привязан к правильной зоне;
- в зоне сохранен корректный `CIDR`;
- IP-адрес клиента действительно попадает в нужный маршрут;
- у зоны не задана ошибочная резервная зона;
- origin-стример доступен и отдает поток;
- рестример получает `config_external`;
- в зоне есть хотя бы один доступный рестример.

Если IP-адрес клиента не попадает ни в один маршрут, CDN-маршрутизация для такого клиента не применяется. В этой ситуации в первую очередь проверяют правильность сети, маски и порядок фактической маршрутизации трафика в корпоративной инфраструктуре.

Рекомендуемый эксплуатационный подход

Для устойчивой работы внутрикорпоративной `CDN` рекомендуется:

- называть зоны по физической или организационной топологии;
- не смешивать в одной зоне несвязанные офисы;
- держать хотя бы один резервный маршрут обслуживания для критичных площадок;
- регулярно проверять состояние рестримеров в разделе мониторинга;
- отдельно контролировать доступность origin и edge-узлов;
- документировать, какой диапазон IP относится к какому офису.

Такой подход упрощает сопровождение, масштабирование и аварийное переключение при проблемах в сети или на отдельных узлах.

2.2.3 Мониторинг Agora

Мониторинг Agora состоит из двух составляющих:

- система, которая снимает первичные показатели (метрики)
- система, которая агрегирует и может выдать алерты.

Первичные показатели оформляются в виде метрик и доступны по протоколу HTTP(S) в виде JSON или openmetrics (Prometheus). Большинство показателей оформляются в виде накапливающихся счетчиков (единый подход, стандартный для Prometheus): байты, количество ошибок и т.п.

Предполагается, что агрегирующая система (встроенная в Agora или используемая клиентом) будет самостоятельно дифференцировать эти счетчики.

Для агрегирующей системы мы подготовили ряд шаблонов для алертов, которые должны помочь настроить первичный мониторинг и в последствии дорабатывать под свои нужды.

Встроенный мониторинг в админке

Помимо внешнего сбора метрик, в Agora есть встроенные экраны мониторинга в административном интерфейсе. Они позволяют оператору и администратору быстро оценивать текущее состояние платформы без обращения к внешней системе наблюдаемости.

Мониторинг стримеров

На странице стримеров отображаются агрегированные показатели по каждому зарегистрированному стримеру.

В списке доступны:

- hostname стримера;
- количество online streams и общее количество потоков;
- количество подключенных клиентов;
- загрузка CPU;
- использование диска.

Страница позволяет:

- открыть карточку конкретного стримера;
- обновить текущие данные вручную;
- использовать этот экран как быстрый обзор состояния инфраструктуры.

Такая страница удобна для первичной диагностики деградации узлов, перегрузки сервера или оценки общей загрузки платформы.

Мониторинг потоков

Мониторинг потоков в Agora доступен как на странице списка потоков, так и в карточке отдельного потока.

На обзорной странице потоков видны:

- общий статус потока;
- входной битрейт;
- текущий выходной битрейт;
- режим работы входа;
- состояние исходящих публикаций.

Для списка потоков также доступно ручное обновление и автообновление с заданным интервалом.

Детальная статистика потока

В карточке потока Agora показывает подробную статистику по кластеру.

В зависимости от конфигурации потока интерфейс отображает:

- общий статус потока в кластере;
- состояние `primary` и `backup` в режиме `Twincast`;
- эффективный источник входного сигнала;
- входной битрейт;
- статистику входа:
- байты;
- кадры;
- retries;
- переключения входа;
- изменения media info;
- ошибки;
- время последнего `DTS`;
- состояние и статистику исходящих публикаций.

Если поток работает в `Twincast`, статистика выводится отдельно для `primary` и `backup`, что упрощает диагностику проблем на конкретном тракте приема.

Мониторинг исходящих публикаций

Agora также показывает отдельную статистику egress и исходящих публикаций.

Для каждой исходящей публикации могут отображаться:

- URL публикации;
- статус push;
- количество переданных байт;
- количество кадров;
- количество ошибок.

В режиме `Twincast` статистика исходящих публикаций может быть разнесена по двум серверам, чтобы оператор видел различия между `primary` и `backup`.

Использование внешней системы мониторинга

Встроенный мониторинг удобен для оперативной работы, но для промышленной эксплуатации рекомендуется подключать внешнюю систему мониторинга.

Это позволяет:

- хранить длинную историю метрик;
- строить графики и дашборды;
- настраивать алерты;
- интегрировать Agora в корпоративный контур наблюдаемости;
- передавать данные в `Prometheus`, `Zabbix` или другие системы мониторинга.

Что обычно контролируют в первую очередь

Для первичного production-мониторинга обычно рекомендуется отслеживать:

- доступность стримеров и ретрансляторов;
- загрузку CPU, диска и пропускной способности;
- количество активных потоков и клиентов;
- входной битрейт и ошибки на входе;
- состояние `Twincast` primary / backup;
- ошибки исходящих публикаций;
- состояние записи, архива и доставки контента.

2.2.4 Устройства ввода/вывода

`IO`-устройства в Agora описывают аппаратные входы и выходы, которые связаны с конкретными стримерами. Они настраиваются отдельно от потоков и затем могут использоваться в конфигурации потока как источники сигнала, например для `HDMI` или `SDI`.

Практический смысл IO-устройств

`IO`-устройства нужны для того, чтобы связать логическую конфигурацию Agora с реальными аппаратными входами и выходами на конкретных стримерах. Без этой связи поток нельзя надежно привязать к физическому `HDMI`- или `SDI`-источнику.

Их необходимо описать в Agora для правильного распределения потоков по физическим origin серверам

Зачем нужны IO-устройства

`IO`-устройства нужны для того, чтобы Agora знала:

- на каком стримере находится конкретная плата или вход;
- какой это тип аппаратного источника;
- как этот источник должен использоваться в системе.

Без предварительной настройки `IO`-устройства оператор не сможет корректно выбрать соответствующий аппаратный вход в настройках потока.

Привязка к стримеру

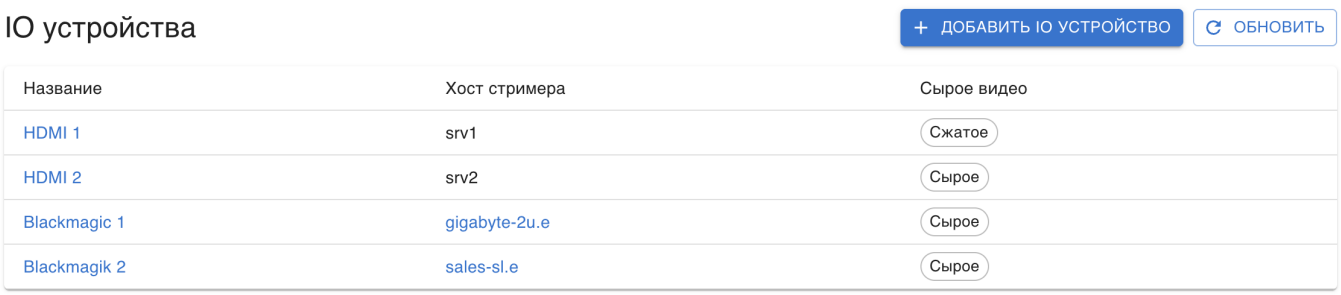
Каждое `IO`-устройство привязывается к стримеру. Это обязательная часть логики системы, потому что аппаратный вход физически существует на конкретном сервере захвата.

Именно поэтому:

- `IO`-устройства нужно настраивать отдельно;
- при создании устройства нужно указать, к какому стримеру оно относится;
- только после этого устройство можно использовать в потоке как источник `HDMI` или `SDI`.

Таким образом, поток с аппаратным захватом всегда связан не только с типом входа, но и с конкретным стримером, на котором этот вход доступен.

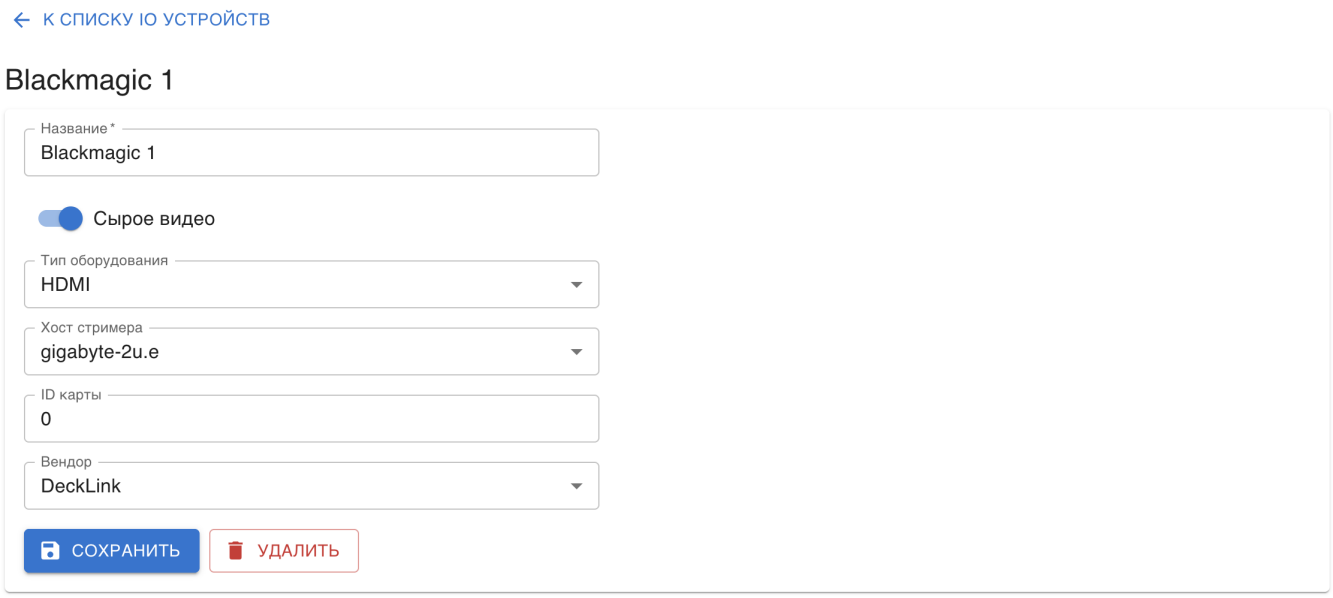
Список IO-устройств



На странице списка `IO`-устройств оператор может:

- просматривать все зарегистрированные устройства;
- видеть имя устройства;
- видеть стример, к которому устройство привязано;
- видеть, работает ли устройство в режиме `raw` или в обычном режиме;
- перейти в карточку устройства;
- создать новое устройство;
- обновить список устройств.

Карточка IO-устройства



В карточке устройства доступны следующие параметры:

- имя устройства;
- признак `raw`;
- тип аппаратного входа `hw_type`;
- стример, к которому привязано устройство;
- `card_id`;
- вендор устройства.

ТИП АППАРАТНОГО УСТРОЙСТВА

В текущем интерфейсе можно указать аппаратный тип:

- `HDMI`;
- `SDI`.

Это позволяет использовать устройство в соответствующих сценариях захвата.

ПРИВЯЗКА К СТРИМЕРУ

В форме настройки оператор выбирает стример из списка известных стримеров Agora. Это связывает устройство с конкретным узлом захвата.

Если устройство не привязано к стримеру, его невозможно корректно использовать в production-сценарии аппаратного приема.

ИДЕНТИФИКАТОР КАРТЫ И ВЕНДОР

Для аппаратных плат также можно указать:

- `card_id` - идентификатор платы или входа на стороне стримера;
- `vendor` - производителя устройства.

Эти параметры нужны для точной привязки логической конфигурации Agora к реальному аппаратному ресурсу.

Использование в потоках

После того как `IO`-устройство настроено и привязано к стримеру, его можно выбрать в потоке как источник сигнала.

Это особенно важно для:

- `HDMI`-захвата;
- `SDI`-захвата;
- других сценариев, где источник определяется не сетевым адресом, а физическим входом на плате.

В интерфейсе настройки потока оператор видит список доступных устройств и выбирает нужное по связке:

- hostname стримера;
- имя устройства.

Управление устройствами

Для `IO`-устройства доступны стандартные действия:

- создание;
- изменение;
- сохранение;
- удаление.

Если оператор изменил поля формы, интерфейс предупреждает о несохраненных изменениях при попытке покинуть страницу.

2.3 Manage

2.3.1 Управление контентом

Agora управляет несколькими типами медиаконтента, которые отличаются жизненным циклом, способом публикации и сценариями использования.

К основным типам контента относятся:

- прямые эфиры;
- записи прошедших эфиров;
- `VOD`-файлы.

Все эти типы контента поддерживаются в рамках единой платформы управления, но имеют разные правила подготовки, публикации, хранения и доставки.

Подробно каждый из этих сценариев будет описан на отдельных страницах документации.

2.3.2 Streams

Управление потоками в Agora

В отдельные независимые видеопотоки в Agora организуются:

- телеканалы, вещаемые постоянно (как правило, это каналы внутреннего производства);
- разовые трансляции значимых событий;
- потоки с фиксированных камер, которые участвуют в формировании контента.

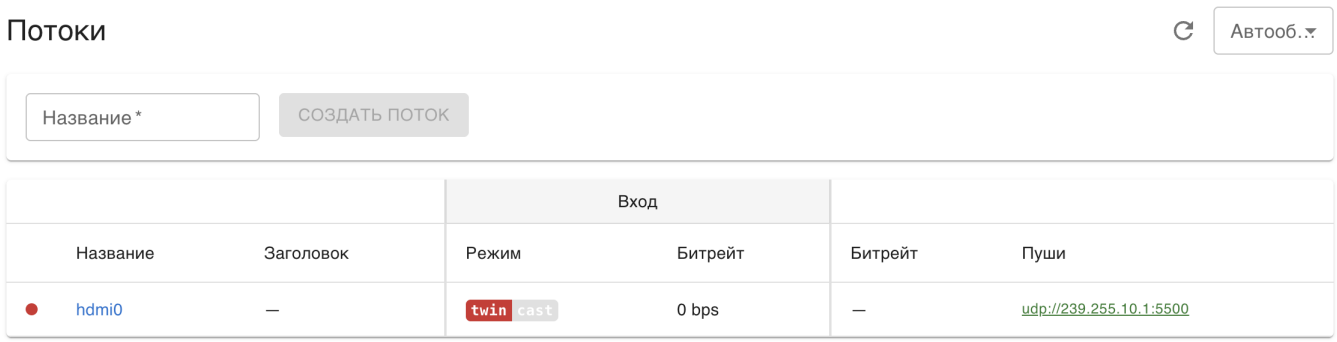
Поток в Agora является кластерным представлением всех потоков на стримерах и ретрансляторах, участвующих в доставке.

Настройка захвата источника и выбор стратегии кластеризации описаны на отдельной странице: Захват источника.

ПРАКТИЧЕСКИЙ СМЫСЛ УПРАВЛЕНИЯ ПОТОКАМИ

Управление потоками в Agora нужно для того, чтобы собрать в одной сущности прием сигнала, его обработку, исходящие публикации и контроль состояния. Именно поток является основной единицей работы оператора: через него задается логика вещания, резервирования, транскодирования и доставки контента.

СПИСОК ПОТОКОВ



На странице управления потоками оператор может:

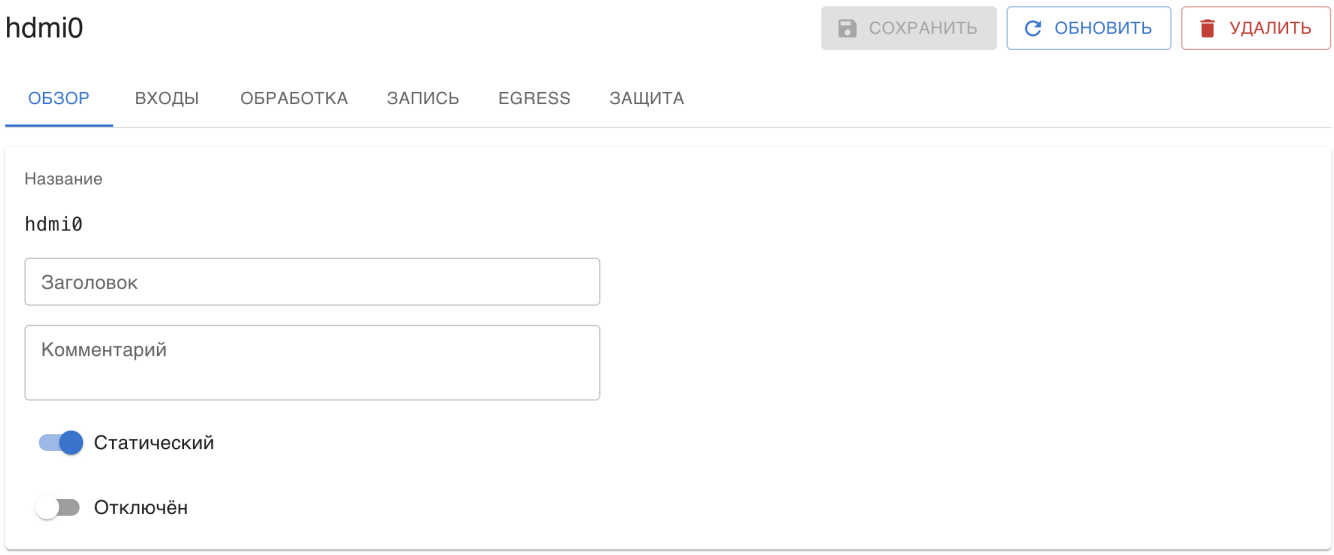
- просматривать список всех потоков;
- создавать новый поток по имени;
- открывать карточку конкретного потока;
- вручную обновлять список;
- включать автоматическое обновление списка с интервалом `5`, `15` или `60` секунд.

В таблице потоков отображаются основные оперативные параметры:

- общий статус потока;
- системное имя потока;
- отображаемый заголовок;
- режим входа;
- входной битрейт;
- текущий выходной битрейт;
- список исходящих публикаций и их состояние.

Для потоков в режиме `twincast` интерфейс отдельно показывает состояние `primary` и `backup` тракта. Для исходящих публикаций в списке отображаются URL и их текущий статус, что позволяет быстро понять, какие публикации активны, а какие находятся в ожидании.

КАРТОЧКА ПОТОКА



Из списка можно перейти в карточку потока. В ней доступны:

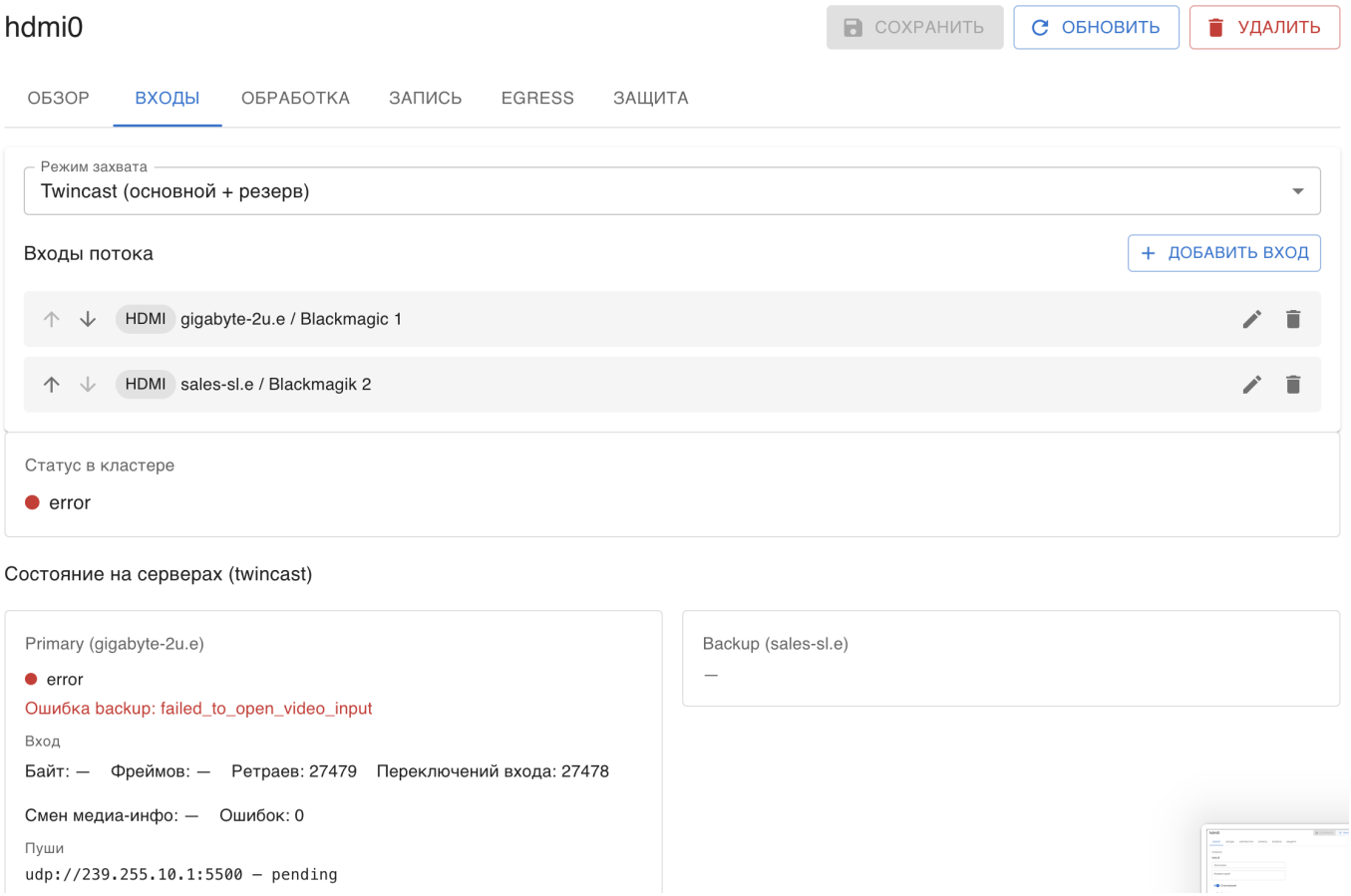
- просмотр и изменение базовых свойств потока;
- редактирование входов;
- настройка обработки потока;
- работа с публикацией и исходящими публикациями;
- просмотр статистики потока и push-статистики;
- сохранение изменений;
- перезагрузка конфигурации;
- удаление потока.

Подробные возможности редактирования отдельных вкладок будут описаны на отдельных страницах документации.

Захват источника

Настройки захвата источника в Agora задаются в карточке потока на вкладке входов. Именно здесь оператор определяет, откуда поток будет приниматься, в каком порядке обрабатываются входы и какая стратегия кластеризации используется для приема сигнала.

ЧТО НАСТРАИВАЕТСЯ НА СТРАНИЦЕ ЗАХВАТА



На странице захвата оператор может:

- выбрать стратегию кластеризации входа;
- добавить новый источник;
- изменить существующий источник;
- удалить источник;
- поменять порядок входов;
- просматривать краткую сводку по каждому входу.

Если у потока настроено несколько входов, их порядок имеет значение, поэтому интерфейс позволяет поднимать и опускать входы в списке.

СТРАТЕГИЯ КЛАСТЕРИЗАЦИИ

При настройке захвата оператор должен самостоятельно решить, какую стратегию кластеризации выбрать:

- `cluster ingest`;
- `double publish`;
- `twincast`.

Выбор этой стратегии определяет, как именно платформа будет принимать сигнал и как будет вести себя система при отказе одного из трактов приема.

По умолчанию используется `cluster ingest`. Это стандартный механизм, который обещает, что внешний стрим будет принят на одном из `origin`-узлов или связанных с ними трактов захвата.

Однако в задачах корпоративного телевидения `cluster ingest` встречается не так часто, как в классическом телевещании или видеонаблюдении. Причина в том, что `cluster ingest` подразумевает наличие источника, к которому система может сама обратиться за потоком. В корпоративном ТВ такой сценарий используется реже, поскольку там чаще встречается не самостоятельный захват, а публикация сигнала в сторону Agora.

Подробное описание этих режимов приведено на отдельных страницах архитектуры:

- `Cluster Ingest`;
- `Double Publish`;
- `Twincast`.

ПОДДЕРЖИВАЕМЫЕ ТИПЫ ВХОДОВ

В текущем интерфейсе Agora для потока можно настроить следующие типы входов:

- `publish`;
- `multicast`;
- `SRT`;
- `RTSP`;
- `HDMI`.

Каждый тип входа имеет свой набор параметров.

Multicast

Для `multicast` указываются:

- IP-адрес;
- порт;
- номер программы, если он используется.

Такой вход подходит для приема потока по локальной IP-сети предприятия

SRT

Для `SRT` в режиме `caller` (т.е. когда соединение открывается со стороны Agora) настраиваются:

- host;
- порт;
- passphrase, если используется защита соединения.

Этот режим используется для надежного приема потока по IP-сети.

`SRT` в Agora может использоваться в двух основных сценариях:

- как захват потока из интернета или с удаленного устройства, когда Agora сама подключается к источнику;
- как прием публикации, когда внешний источник сам публикует поток в сторону Agora.

В случае, если источник сам публикует по SRT, нужно указать режим `publish` и настроить порт для приема SRT

RTSP

Для `RTSP` задается URL источника.

Такой вход используется приема для камер, аудио кодеров и других IP-источников, работающих по `RTSP`.

HDMI

Для `HDMI` оператор выбирает устройство из списка доступных `IO`-устройств.

В списке показываются:

- hostname стримера;
- имя устройства.

Если свободных `HDMI`-устройств нет, интерфейс сообщает об этом отдельно.

Выбор такого устройства автоматически привязывает поток к конкретному стримеру. Для обеспечения надежности, надо указать более одного устройства.

РАБОТА СО СПИСКОМ ВХОДОВ

Для каждого входа в списке отображаются:

- тип входа;
- краткая сводка параметров;
- действия редактирования;
- действия удаления;
- кнопки изменения порядка.

Это позволяет оператору быстро проверить конфигурацию приема и перестроить приоритет входов без ручного редактирования служебных данных.

АВТОМАТИЧЕСКОЕ ПЕРЕКЛЮЧЕНИЕ МЕЖДУ ИСТОЧНИКАМИ

Автоматическое переключение между источниками является одним из самых важных механизмов работы с входами в Agora.

У одного потока может быть несколько источников, отсортированных по приоритету. Верхние позиции в списке считаются более приоритетными, а нижние используются как резервные или альтернативные.

Если на текущем активном источнике в течение времени, указанного как `source_timeout`, не появляются кадры, система переключается на следующий доступный источник по приоритету.

Это позволяет заранее подготовить резервные сценарии показа без ручного вмешательства оператора.

Пример приоритетов источников

Один из типичных сценариев для корпоративного ТВ:

- первым приоритетом задается публикация потока;
- вторым приоритетом задается захват внешнего источника.

В такой конфигурации:

- пока в Agora поступает публикация, зрители видят сигнал из студии;
- если студийная публикация пропадает и в течение `source_timeout` не появляются кадры, система переключается на второй источник;
- в качестве второго источника может использоваться заставка, резервный канал или другой подготовленный поток.

Таким образом можно построить схему, в которой в штатном режиме показывается live-сигнал из студии, а при его пропадании автоматически включается резервный контент без участия оператора.

РЕЖИМ ОЖИДАНИЯ ПУБЛИКАЦИИ

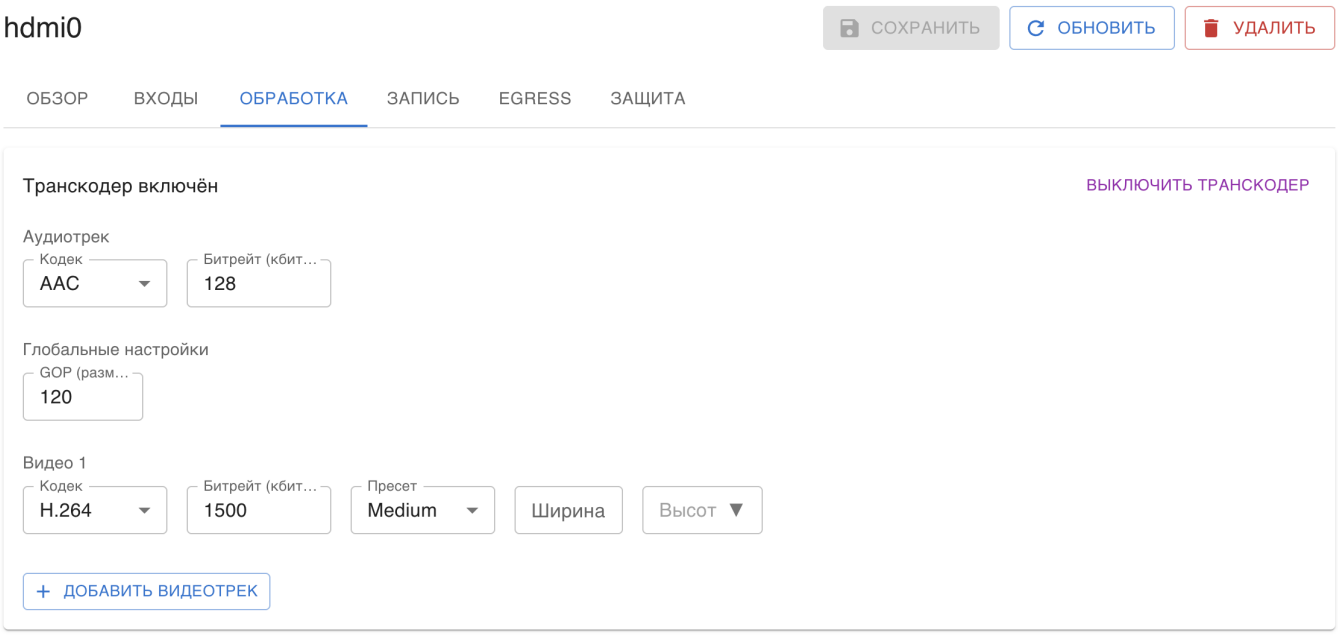
Для части сценариев в корпоративном ТВ источник заранее не задается как фиксированный вход. Вместо этого поток создается в режиме ожидания публикации.

В таком режиме:

- у потока может не быть заранее указанного источника;
- Agora не инициирует самостоятельный захват;
- система ожидает, что внешний кодер или приложение опубликует поток в сторону Agora.

Этот режим особенно характерен для студийной работы, программных энкодеров и разовых эфиров, когда оператор заранее готовит поток в системе, а фактическая подача сигнала начинается только в момент публикации.

Обработка потока



Прежде всего входной поток в Agora обычно нужно транскодировать, чтобы он корректно проигрывался на всех используемых устройствах. Источники сигнала в корпоративном ТВ могут быть очень разными по кодекам, битрейтам, разрешению и структуре потока, а Agora подготавливает их к стабильной доставке по сети предприятия на типовые клиентские устройства.

В этом разделе настраивается транскодер потока.

ПРАКТИЧЕСКИЙ СМЫСЛ ОБРАБОТКИ

В результате настройки обработки Agora позволяет:

- принять разнородные источники
- привести входной поток к предсказуемому формату;
- обеспечить совместимость с клиентскими устройствами;
- ограничить и стабилизировать битрейт для доставки по корпоративной сети;
- подготовить поток к однобитрейтной или мультибитрейтной публикации.

Это особенно важно в корпоративном ТВ, где один и тот же сервис может одновременно использовать:

- студийные источники;
- аппаратные входы `HDMI` / `SDI`;
- сетевые публикации;
- удаленные IP-источники.

Транскодер делает такие потоки пригодными для устойчивой доставки по корпоративной сети и воспроизведения на устройствах, которые используются в организации.

ЗАЧЕМ НУЖНА ОБРАБОТКА ПОТОКА

Транскодирование используется для того, чтобы:

ВКЛЮЧЕНИЕ И ВЫКЛЮЧЕНИЕ ТРАНСКОДЕРА

Во вкладке обработки оператор может:

- включить транскодер;
- отключить транскодер;
- изменить параметры уже включенного транскодирования.

Если транскодер включается, Agora по умолчанию создает базовую конфигурацию:

- один аудиотрек;
- один видеотрек.

Это соответствует типовой однобитрейтной доставке.

АУДИОТРЕК

Для аудиотрека в текущем интерфейсе настраиваются:

- кодек;
- битрейт.

В качестве аудиокодека можно выбрать:

- `AAC`;
- `Opus`.

ВИДЕОТРЕКИ

Для каждого видеотрека можно настроить:

- кодек;
- битрейт;
- preset кодирования;
- ширину;
- высоту.

Поддерживаются видеокодеки:

- `H.264`;
- `H.265`;
- `AV1`.

ОДНОБИТРЕЙТНАЯ И МУЛЬТИБИТРЕЙТНАЯ ДОСТАВКА

По умолчанию Agora использует однобитрейтную доставку: поток подготавливается в одном выходном профиле.

Если требуется мультибитрейтная доставка, оператор может добавить несколько видеотреков с разными параметрами:

- разным битрейтом;
- разным разрешением;
- при необходимости разными preset-параметрами кодирования.

Таким образом один входной поток может быть преобразован в несколько выходных представлений для разных условий сети и разных устройств.

ПОВЕДЕНИЕ ПО БИТРЕЙТУ

По умолчанию Agora использует режим максимального приближения к `CBR` в окне `1` секунды. Это важно для корпоративных сетей, потому что позволяет проектировать доставку потока с более предсказуемой нагрузкой на сеть.

Такой режим помогает:

- оценивать требуемую пропускную способность;
- снижать вероятность кратковременных перегрузок;
- упростить планирование доставки внутри сети предприятия.

GOP

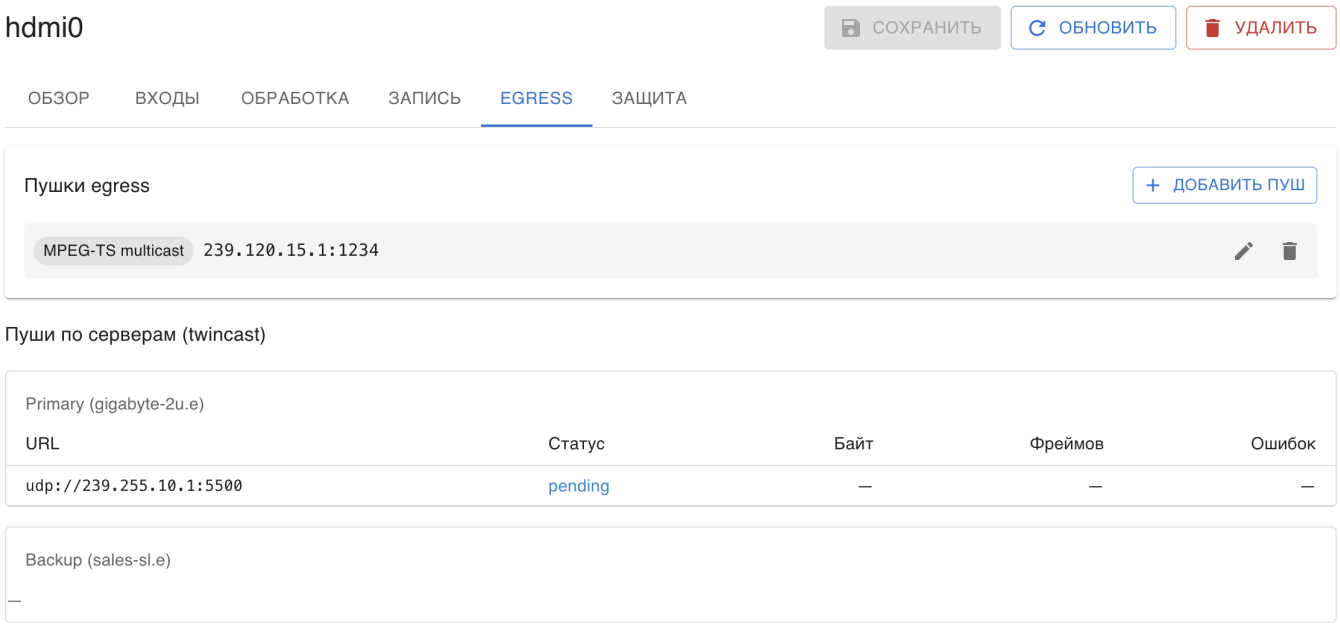
В глобальных настройках транскодера задается параметр `GOP`.

`GOP` влияет на поток следующим образом:

- при увеличении `gop size` снижается битрейт при том же визуальном качестве;
- при увеличении `GOP` растет задержка проигрывания относительно реального времени;
- также увеличивается задержка старта просмотра.

Практическая рекомендация для локальной сети - использовать `GOP`, соответствующий `2` секундам. Это считается хорошей рабочей нормой для корпоративной среды.

Исходящие публикации с помощью Agora



Исходящие публикации определяют, куда Agora отправляет уже подготовленный поток после приема и обработки. Через этот раздел оператор настраивает egress-направления для доставки сигнала в корпоративную сеть, во внешние системы или на соседние узлы платформы.

ПРАКТИЧЕСКИЙ СМЫСЛ ПУШЕЙ

Исходящие публикации (пуши) позволяют отправить поток в те системы, которые не могут сами запросить видео, а именно:

- публикация в соцсети
- публикация во внешние видеостриминговые хостинги
- отправка через multicast

ЧТО МОЖНО НАСТРОИТЬ

На странице исходящих публикаций оператор может:

- добавить новую исходящую публикацию;
- изменить параметры существующей публикации;
- удалить публикацию;
- просматривать краткую сводку по каждому push;
- видеть базовую статистику по уже работающим публикациям.

ПОДДЕРЖИВАЕМЫЕ ТИПЫ ПУБЛИКАЦИЙ

В текущем интерфейсе Agora можно настроить следующие типы исходящих публикаций:

- `MPEG-TS multicast`;
- `SRT`;
- `RTMP`.

Каждый тип публикации имеет собственный набор параметров.

MPEG-TS multicast

Для `MPEG-TS multicast` указываются:

- multicast IP;
- порт;
- номер программы, если он используется.

Этот тип публикации подходит для распространения потока по локальной сети предприятия в сценариях multicast-доставки.

SRT

Для `SRT`-публикации настраиваются:

- host;
- порт;
- passphrase, если используется защищенное соединение.

Такой режим подходит для надежной доставки потока в удаленный сегмент сети, на резервный сервер или на внешний приемник.

RTMP

Для `RTMP` задается URL публикации.

Этот режим используется в тех случаях, когда downstream-система принимает поток по `RTMP`.

РАБОТА СО СПИСКОМ ПУБЛИКАЦИЙ

Для каждой исходящей публикации интерфейс показывает:

- тип исходящей публикации;
- краткую сводку параметров;
- действия редактирования;
- действие удаления.

Если по публикации уже есть статистика, в кратком виде также отображаются:

- объем переданных данных;
- количество кадров;
- количество ошибок.

Это позволяет оператору быстро понять, какие публикации работают штатно, а какие требуют дополнительной проверки.

СТАТИСТИКА ИСХОДЯЩИХ ПУБЛИКАЦИЙ

Agora показывает отдельную статистику по исходящим публикациям потока.

Для каждого push могут отображаться:

- URL или адрес публикации;
- текущий статус;
- количество переданных байт;
- количество кадров;
- количество ошибок.

Статусы публикации позволяют понять текущее состояние egress-направления, например:

- активна ли публикация;
- находится ли она в ожидании;
- выполняется ли повторная попытка;
- произошла ли ошибка.

СТАТИСТИКА В РЕЖИМЕ TWINCAST

Если поток работает в режиме `Twincast`, статистика исходящих публикаций показывается раздельно для:

- `primary`;
- `backup`.

Это особенно важно для диагностики, когда оператор должен понять, с какого тракта идет публикация и есть ли различия между двумя сторонами `Twincast`.

2.4 Security

2.4.1 Управление пользователями в Agora

В Agora реализовано управление пользователями и их правами доступа.

Концепция пользователей существует только для разграничения прав доступа, никакие персональные данные самих пользователей не хранятся, не обрабатываются.

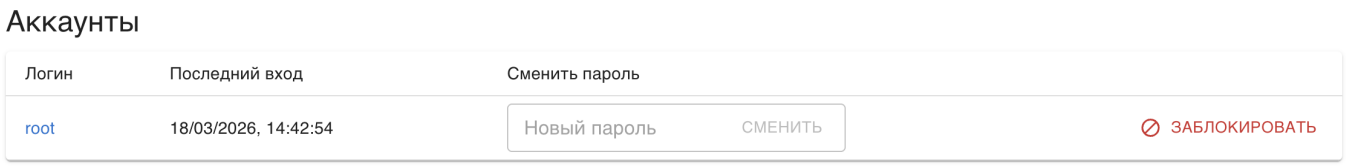
Управление пользователями в Agora сосредоточено в security-разделе административного интерфейса. В текущем интерфейсе доступны список учетных записей, карточка отдельного пользователя, смена пароля, просмотр статуса блокировки и переход к связанным security-данным.

Первичное создание администратора

Необходимо на сервере выполнить команду:

```
./create-account -u USERNAME -p PASSWORD
```

Список пользователей



На странице списка пользователей администратор может:

- просматривать все учетные записи;
- создавать новые учетные записи;
- видеть логин пользователя и его роль;
- видеть время последнего входа;
- быстро сменить пароль прямо из списка;
- перейти в карточку конкретного пользователя;
- заблокировать пользователя.

Если учетная запись уже заблокирована, это отображается прямо в строке пользователя.

СМЕНА ПАРОЛЯ ИЗ СПИСКА

Для быстрой смены пароля:

1. Найдите нужного пользователя в списке.
2. Введите новый пароль в поле `New password`.
3. Нажмите кнопку подтверждения в конце поля или клавишу `Enter`.

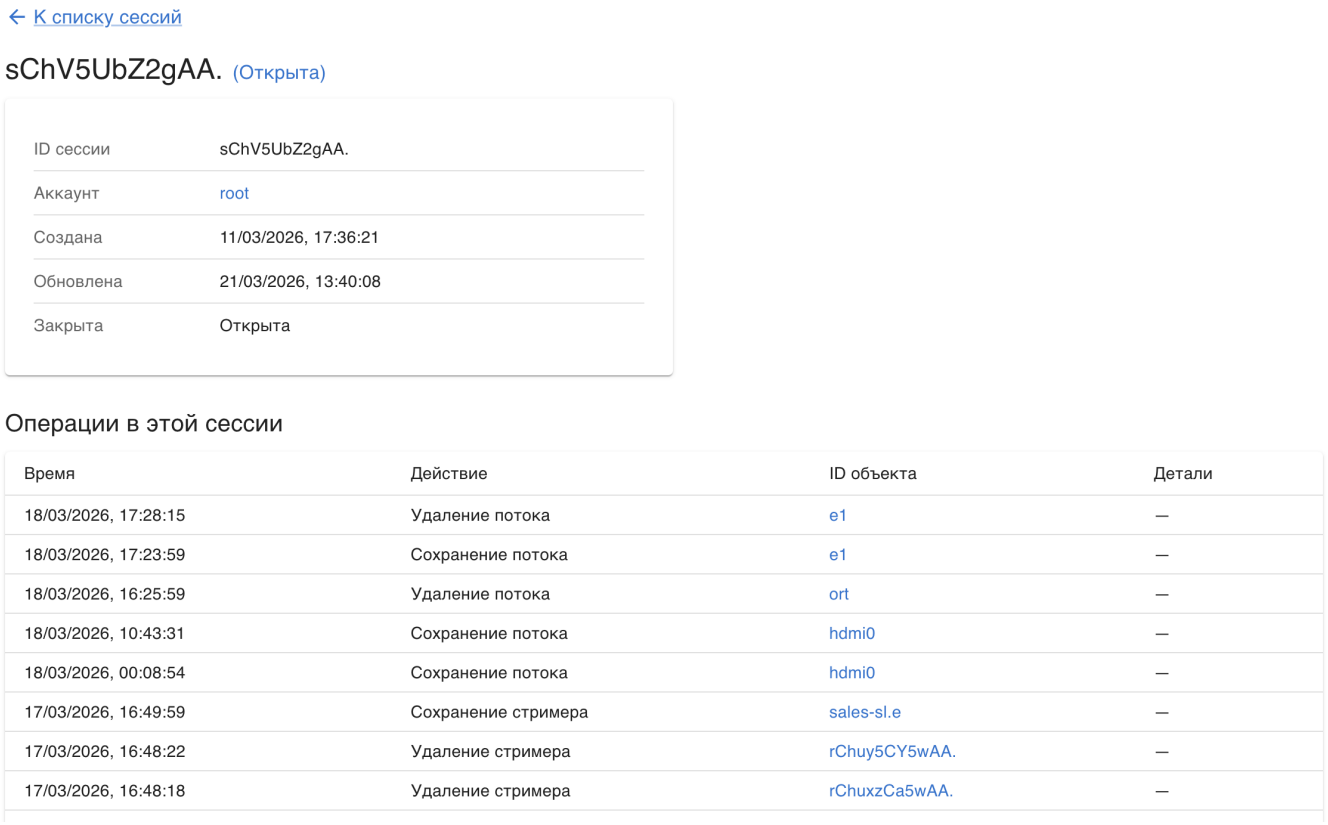
После успешного изменения интерфейс показывает уведомление об успешной смене пароля.

БЛОКИРОВКА ИЗ СПИСКА

При блокировке пользователя немедленно будут заблокированы все его активные сессии.

Разблокировка возможна на странице аккаунта.

Карточка пользователя



Из списка можно открыть карточку отдельного пользователя. В ней доступны:

- `account_id`;
- логин;
- `external_account_id` для возможности связать учетную пользователя с внешней системой;
- дата создания учетной записи;
- дата последнего изменения;
- время последнего входа;
- время блокировки, если пользователь уже заблокирован.
- список текущих открытых сессий и старых, уже закрытых.

Карточка также позволяет:

- сменить пароль для выбранной учетной записи;
- заблокировать пользователя;
- разблокировать пользователя;
- вернуться к общему списку учетных записей.

Если в форме смены пароля введены данные, интерфейс предупреждает о несохраненных изменениях при попытке покинуть страницу.

СМЕНА ПАРОЛЯ В КАРТОЧКЕ ПОЛЬЗОВАТЕЛЯ

Для смены пароля в карточке пользователя:

1. Откройте страницу нужной учетной записи.
2. Введите новый пароль.
3. Подтвердите изменение.

При ошибке интерфейс показывает текст ошибки прямо под полем пароля.

БЛОКИРОВКА ПОЛЬЗОВАТЕЛЯ

Как и в списке пользователей, в карточке есть кнопка блокировки и подтверждающий диалог.

Блокировка пользователя немедленно завершает все его открытые сессии.

Связанные security-функции

Управление пользователями связано с другими security-разделами Agora:

- журнал аудита позволяет просматривать действия пользователей;
- раздел сессий позволяет просматривать открытые и завершенные сессии;
- для открытых сессий предусмотрено принудительное завершение через logout.

Все действия, связанные с пользователями, должны отражаться в audit log и использоваться для последующего контроля и расследования событий безопасности.

Таким образом, даже при неполной реализации блокировки пользователей в API оператор безопасности уже может контролировать активность учетных записей и при необходимости завершать открытые сессии.

Ролевая модель

В Agora есть несколько ролей пользователей:

- администратор, который может управлять стримерами, устройствами и внешними настройками;
- контент-менеджер, который может управлять стримами и `VOD`-файлами;
- наблюдатель, который может следить за мониторингом, но не может редактировать ни стримы, ни настройки;
- сотрудник безопасности, который может читать audit log и при необходимости блокировать пользователей и открытые сессии.

Эта ролевая модель задает целевое разграничение доступа между административными, контентными и security-задачами. Подробные права каждой роли будут описаны на отдельных страницах документации по безопасности и управлению пользователями.

2.4.2 Использование OIDC для провижнинга пользователей и авторизации

Agora позволяет интегрировать свою базу пользователей в корпоративный каталог по протоколу OIDC.

Локальная база данных пользователей отключается (это опционально и при желании можно её оставить) и становится полностью управляемой внешней системой (как правило центральной корпоративной системой, далее IdP (identity provider)). Права пользователя настраиваются во внешней системе, создание, блокировка и удаление пользователя происходит во внешней системе.

Важно: **Agora не обращается к IdP при каждом API-запросе** — она **проверяет JWT** (подпись по **JWKS** или локальному ключу) и из этого токена достает права доступа для этого запроса.

«Провижнинг» пользователя в базу (**JIT**) выполняется при первом успешном запросе с валидным токеном IdP **только если** в токене есть **маппируемая роль** (см. `OIDC_ROLE_CLAIM` / `OIDC_ROLE_MAP`). Запасной роли «по умолчанию» **нет**; без маппируемой роли учётку **не** создают, запрос отклоняют (**403**). Если вы оставляете локальный вход включённым, встроенные учётные записи и выдача JWT через `POST /login` продолжают работать параллельно с IdP (гибридный режим).

Вся коммуникация с IdP происходит по протоколу HTTP(S). Поддержка протоколов LDAP и т.п. происходит через внешнюю систему (через сам IdP).

OIDC не является единственно возможным режимом, по умолчанию Agora работает с встроенной базой пользователей и позволяет самостоятельно управлять пользователями. При этом по сути Agora будет самостоятельно себе выписывать JWT токены.

Настройка OIDC может быть достаточно комплексной, требует настроить много разных деталей и учитывая вариативность корпоративных инсталяций, может быть достаточно громоздкой.

Дальше в документе описываются шаги, позволяющие повторить у себя в окружении базовые вещи и потом настроить сценарий посложнее:

1. Как запустить OIDC вместе с локально запущенным Authentik
2. Какие поля в JWT используются у нас
3. Какие опции настройки OIDC есть в Agora
4. Глоссарий с терминами, используемыми в этом описании

1. Как запустить OIDC вместе с Authentik

1.1. РАЗВЁРТЫВАНИЕ AUTHENTIK

Полная установка Authentik (Docker Compose, Kubernetes, интеграция с LDAP и т.д.) описана в официальной документации Authentik.

В этом мануале мы запустим Agora и Authentik на сервере в локальной сети с помощью Docker Compose для того, чтобы получить общее представление о конфигурации OIDC. При переносе этой конфигурации в продакшн, подход к настройке не изменится.

Для начала запустим Authentik в локальной конфигурации:

```
echo "PG_PASS=$(openssl rand -base64 36 | tr -d '\n')" >> .authentik-env
echo "AUTHENTIK_SECRET_KEY=$(openssl rand -base64 60 | tr -d '\n')" >> .authentik-env
curl -o authentik-compose.yml https://docs.goauthentik.io/compose.yml
docker compose -f authentik-compose.yml --env-file .authentik-env -p authentik up -d
```

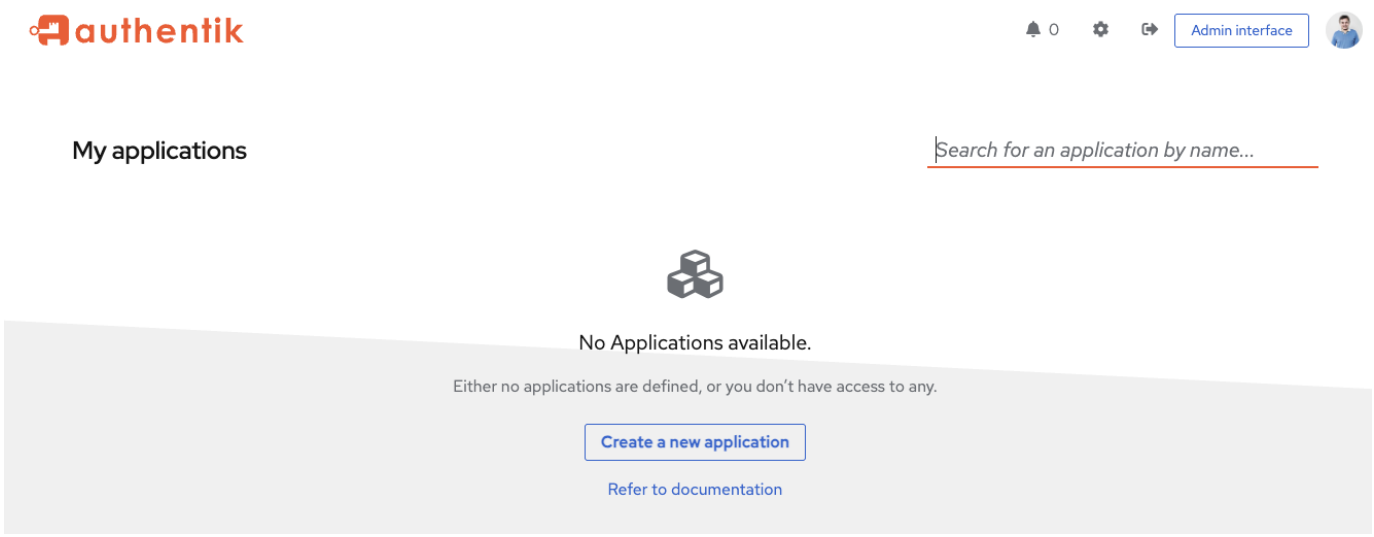
Если что-то пойдет не так, всегда можно остановить полностью удалить authentik:

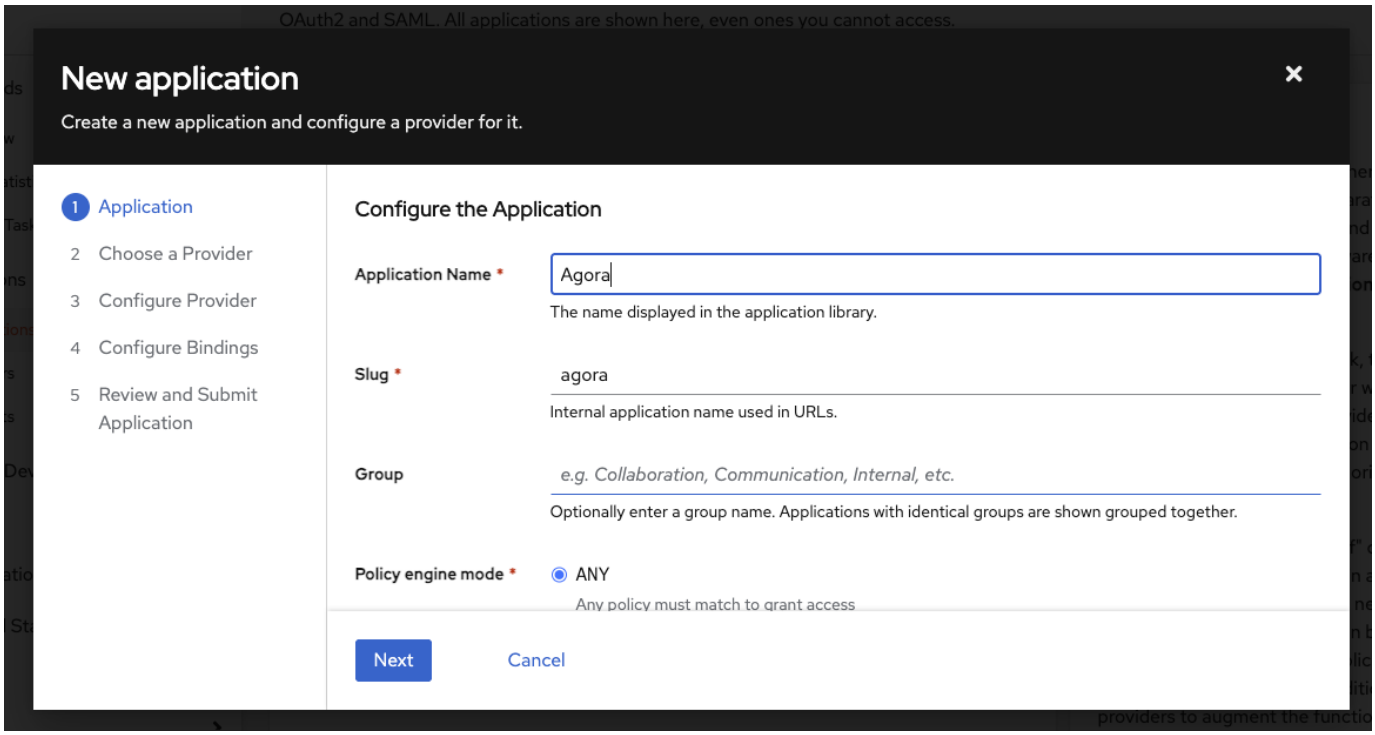
```
docker compose -f authentik-compose.yml --env-file .authentik-env -p authentik down -v
```

Теперь пройдем по адресу http://authentik.local:9000/if/flow/initial-setup/ Здесь authentik.local - адрес сервера, на котором вы запустили этот IdP. В простом варианте это может быть localhost, но главное чтобы на него мог перейти браузер. В этом интерфейсе надо создать администратора для Authentik. Совершенно неважно, что тут указать, это временная инсталяция.

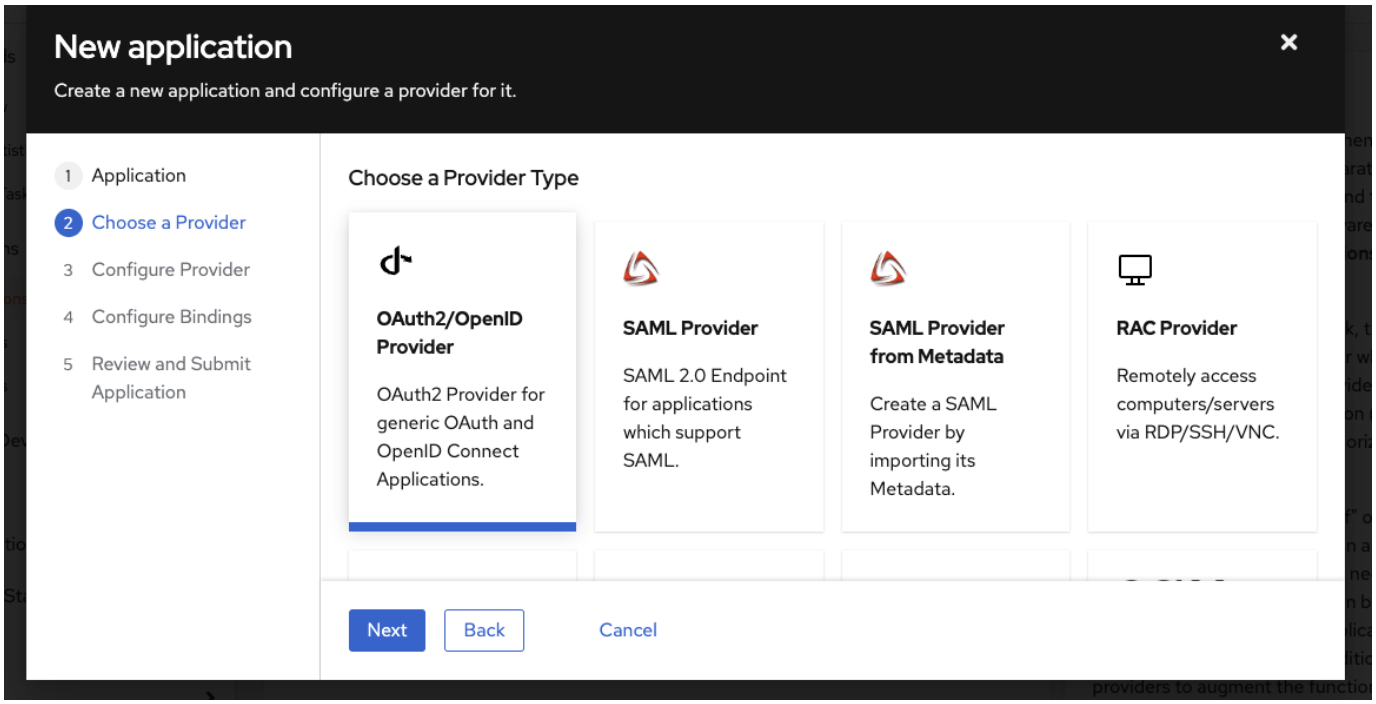
1.2. НАСТРАИВАЕМ AUTHENTIK

1. Создайте в Authentik **Application** (приложение) с именем Agora (slug впишется автоматически).

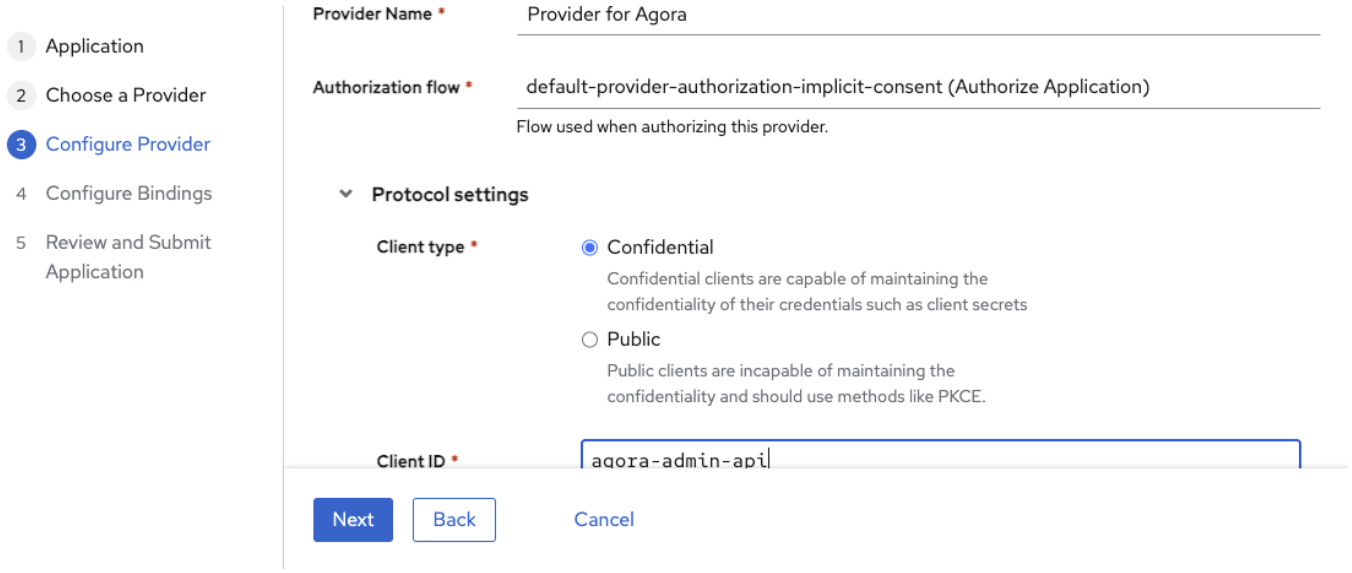




1. Создайте OAuth2/OIDC Provider (просто выберите его в списке)

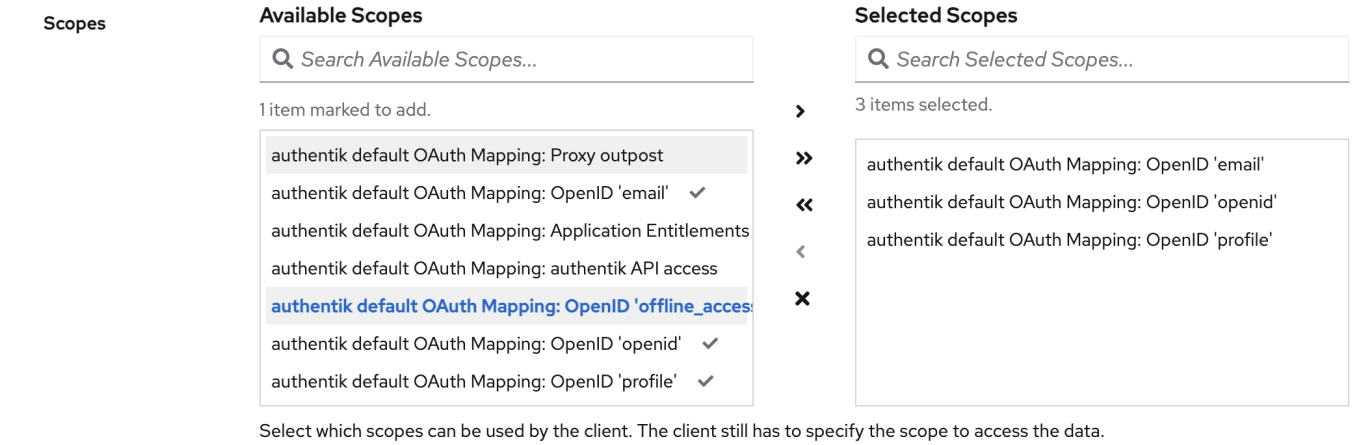


1. Выберите **Authorization flow** implicit (default-provider-authorization-implicit-consent)



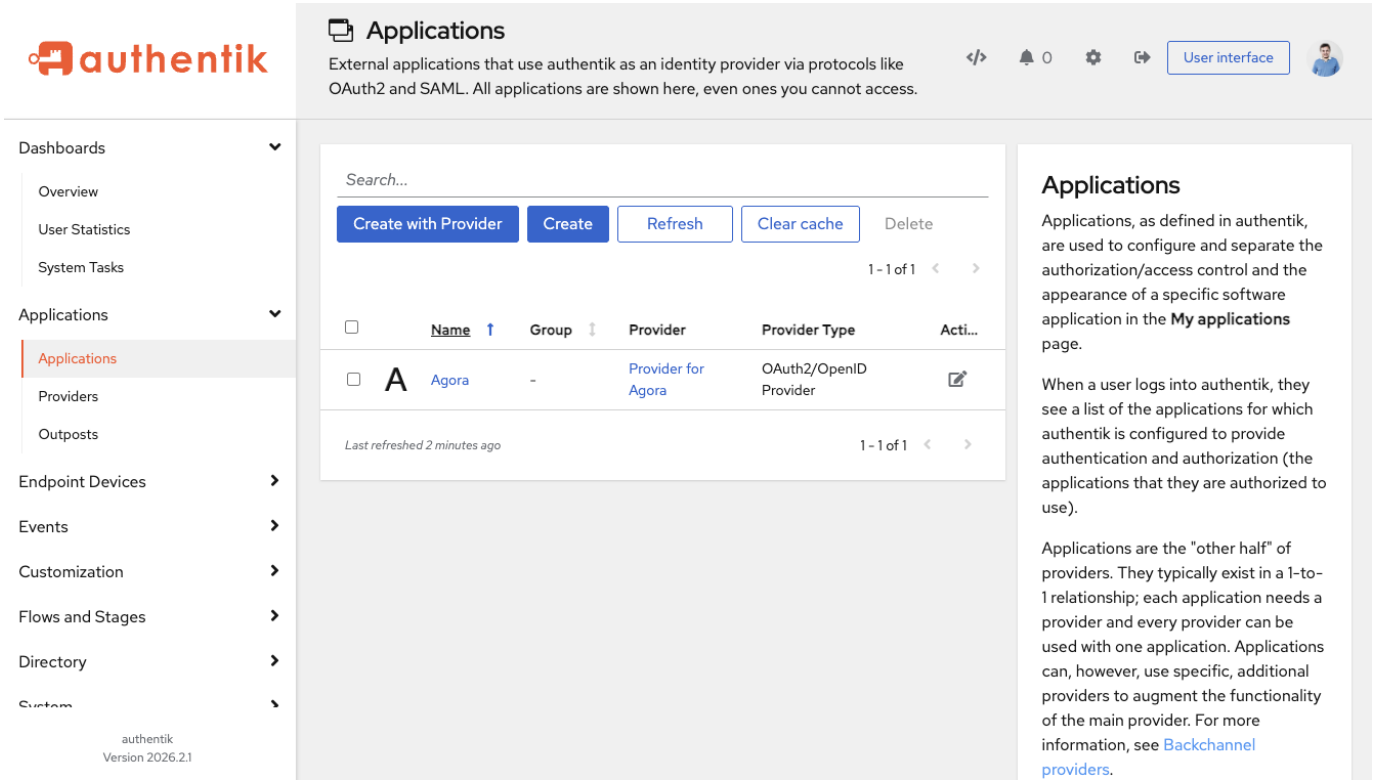
1. На этой странице надо поменять Client ID на `agora-admin-api`. Так будет проще потом. Скопируйте с этой страницы Client Secret. Их надо будет поместить в опции `OIDC_CLIENT_ID` и `OIDC_CLIENT_SECRET` (см ниже раздел настроек Agora)

Здесь же надо зайти в расширенные настройки Advanced protocol settings и добавить `offline_access` к доступным Scopes.

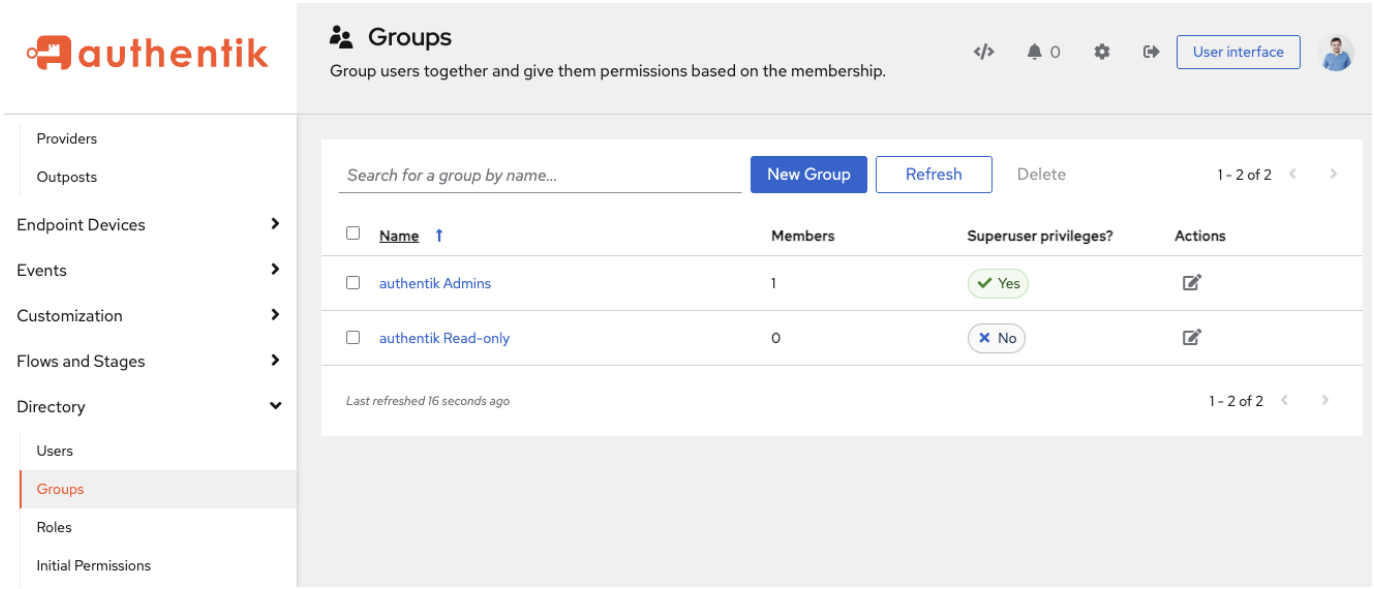


1. На следующей странице пропускаем добавление групп, потому что их ещё нет

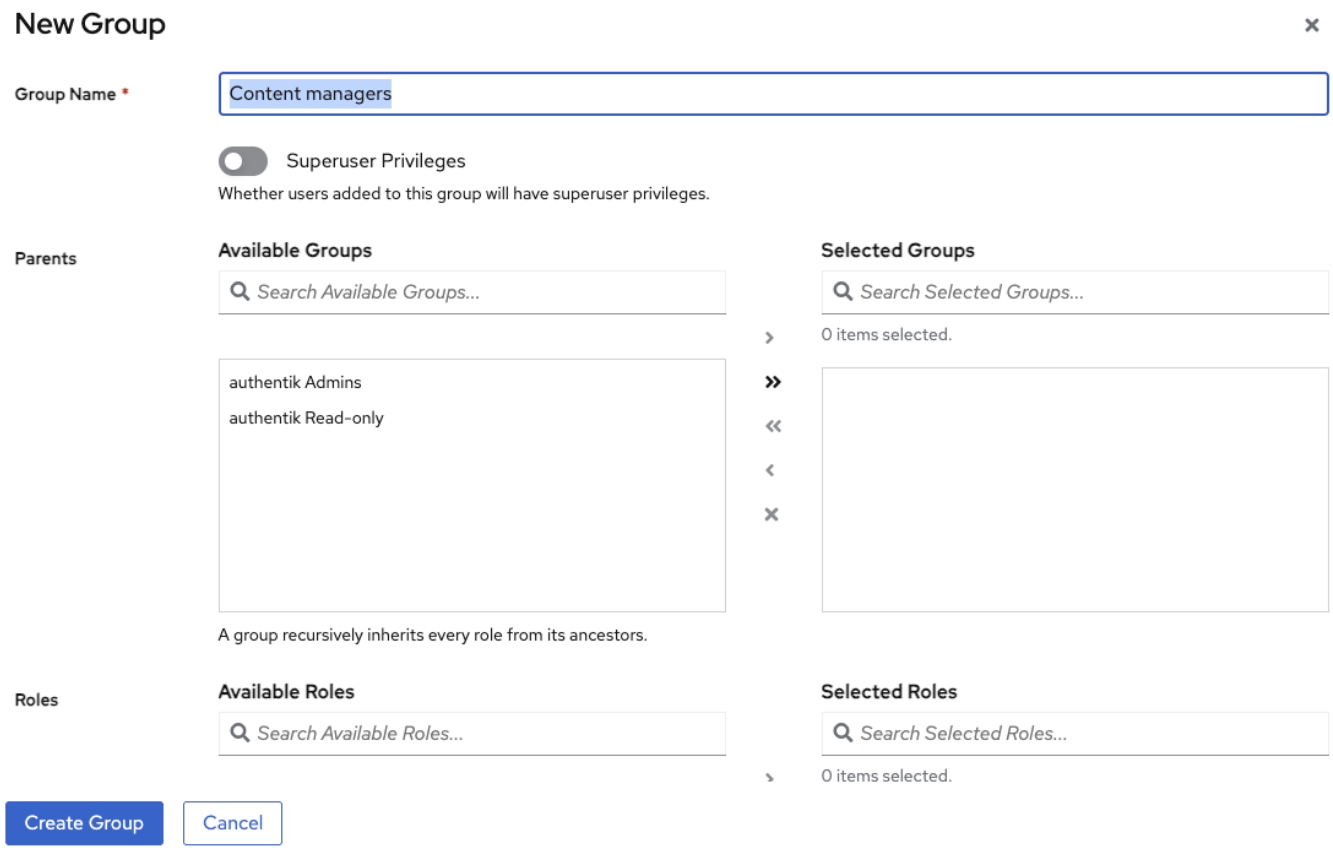
2. На последней странице выбираем Submit и приложение создано



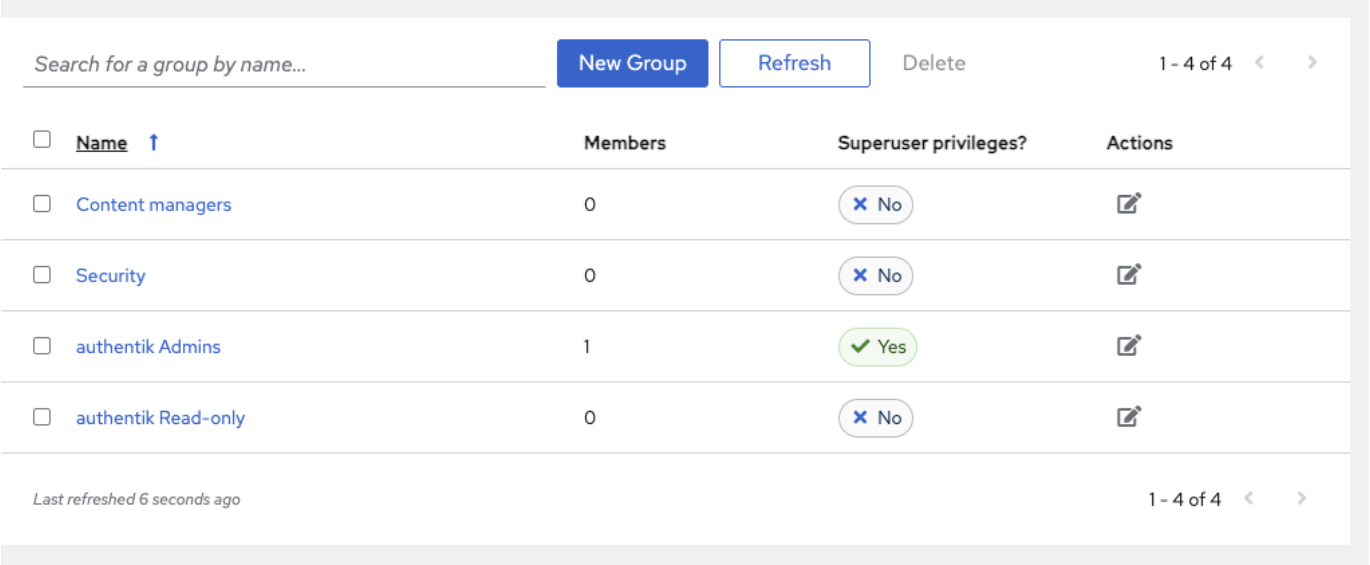
1. Далее надо зайти в раздел групп



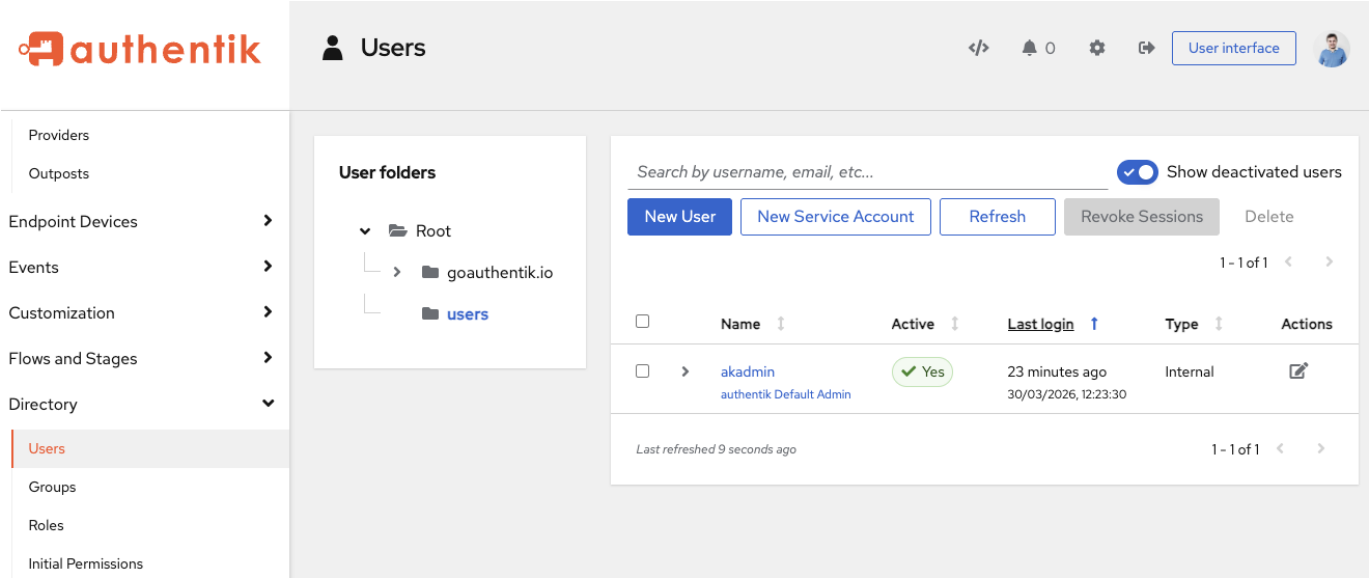
1. Создать две новых группы, например "Content managers" и "Security"



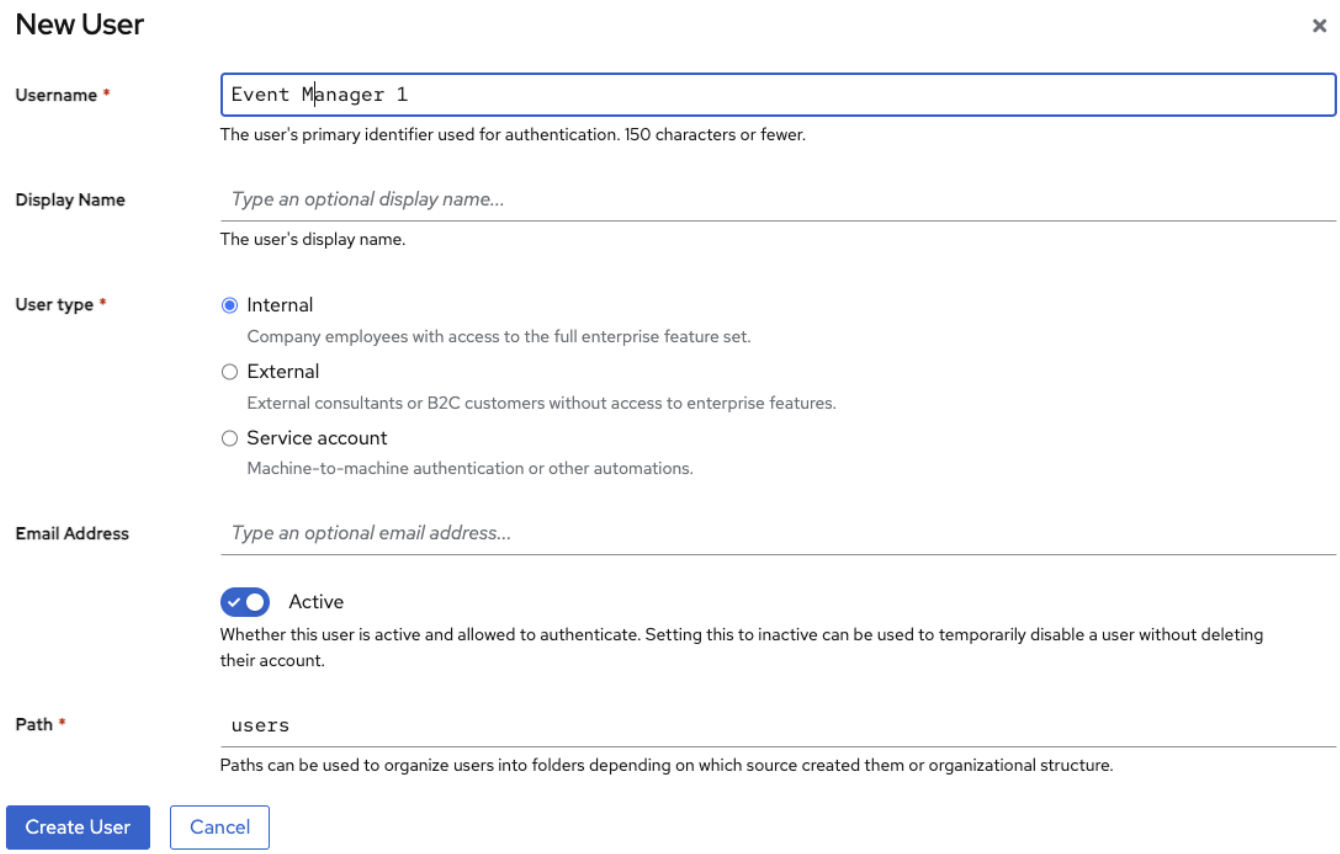
1. Получится список из нескольких групп



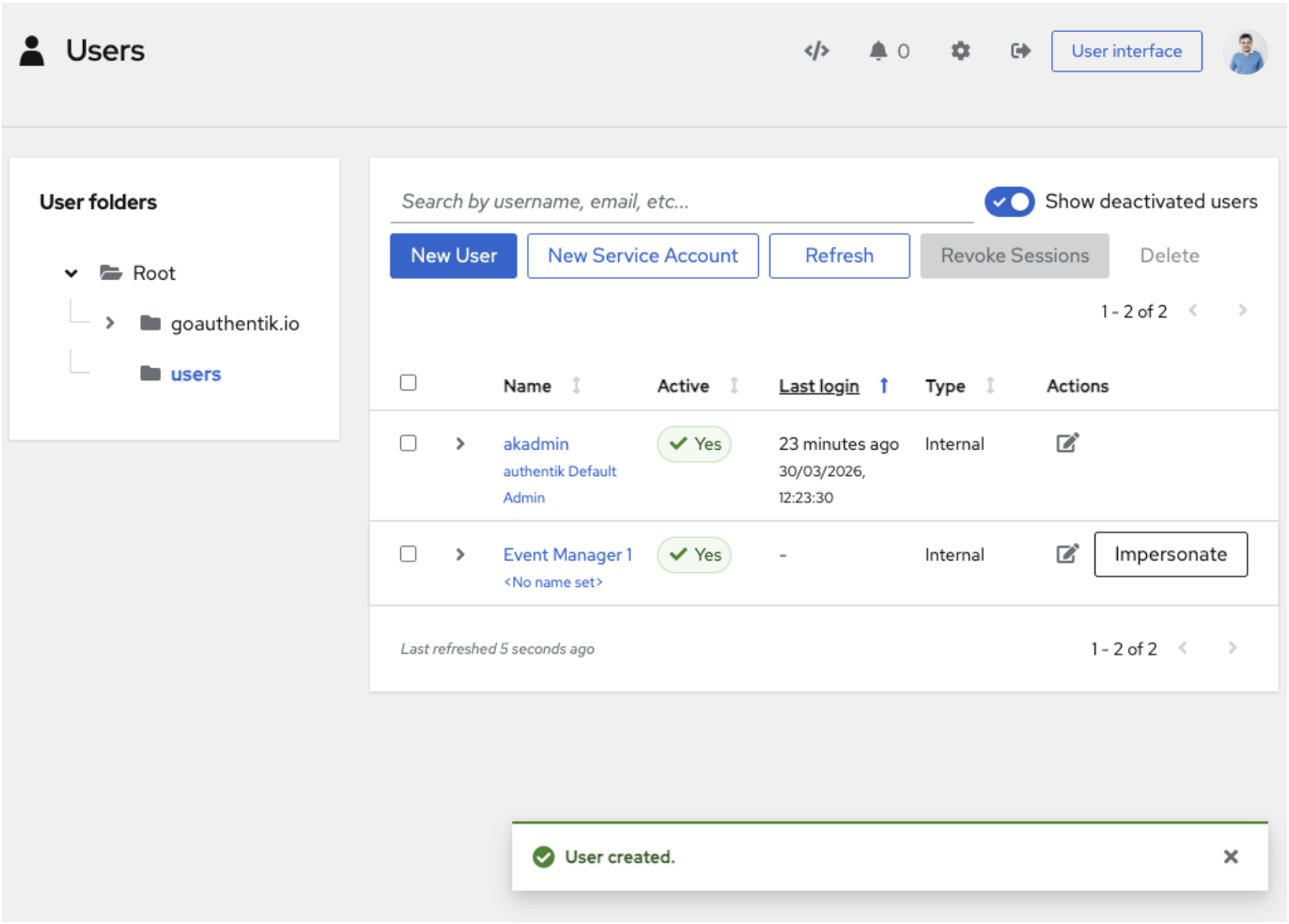
1. Теперь идем в раздел пользователей создавать нового для теста



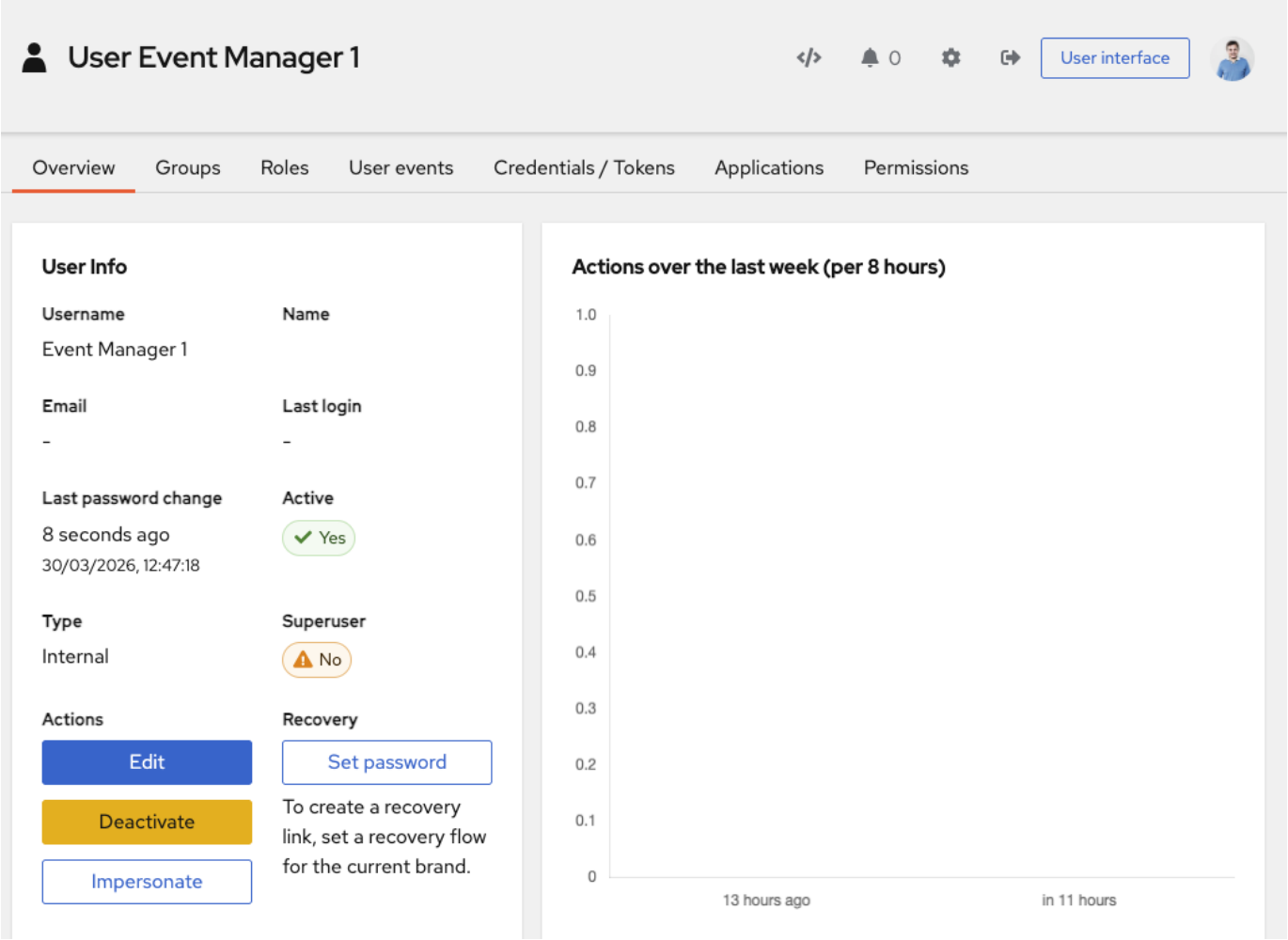
1. Создаем нового, назовем его "Event Manager 1" (возможно, друзья зовут его как-то по-другому, но нам это сейчас не принципиально)



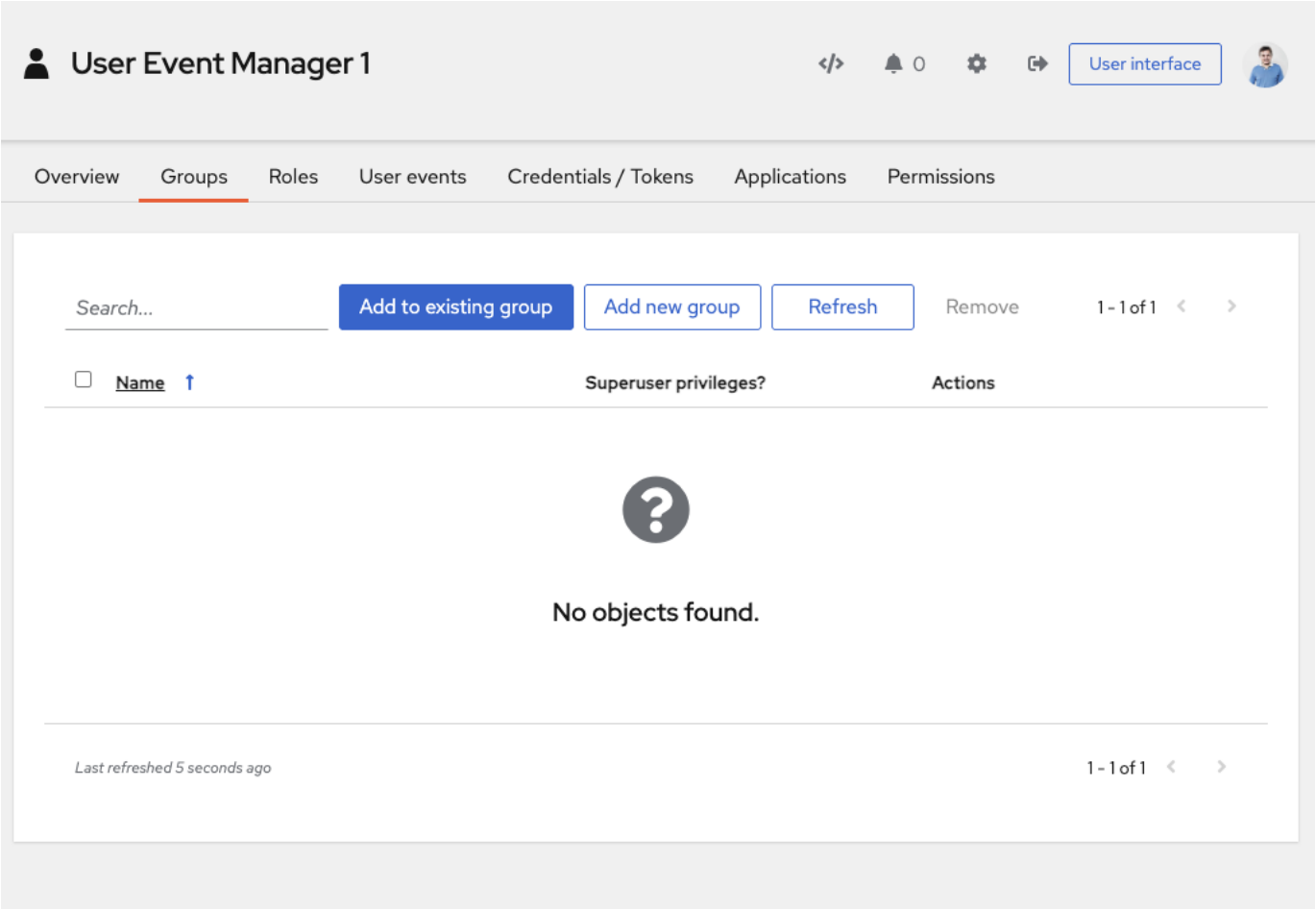
1. Пользователь создан



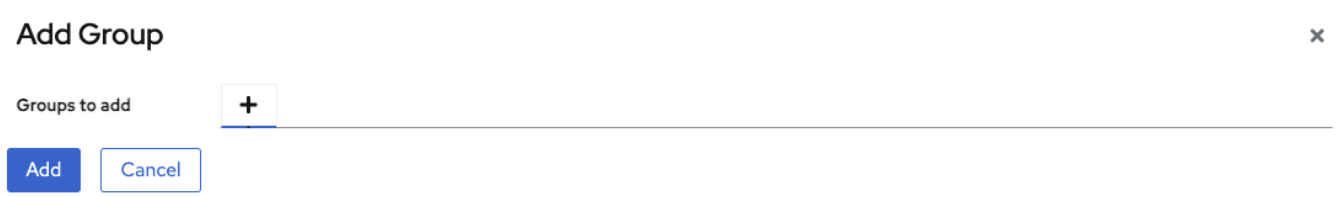
1. И можно перейти на его карточку



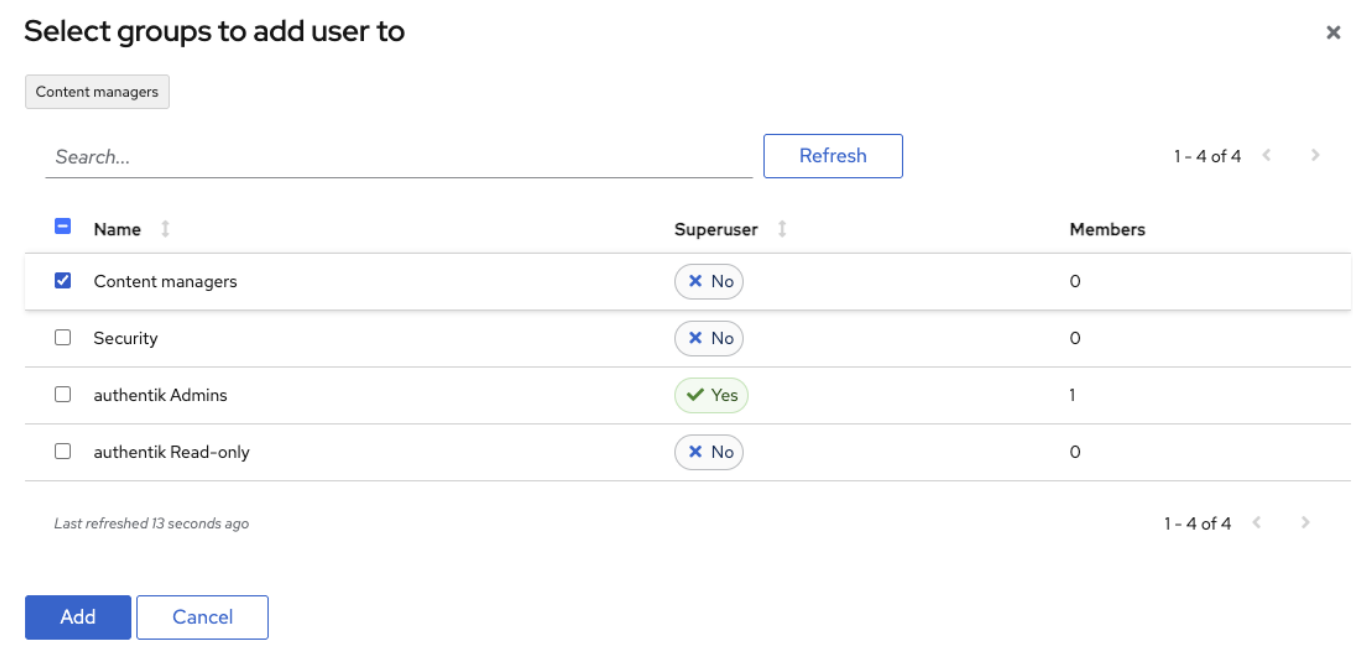
1. Идем во вкладку "Groups"



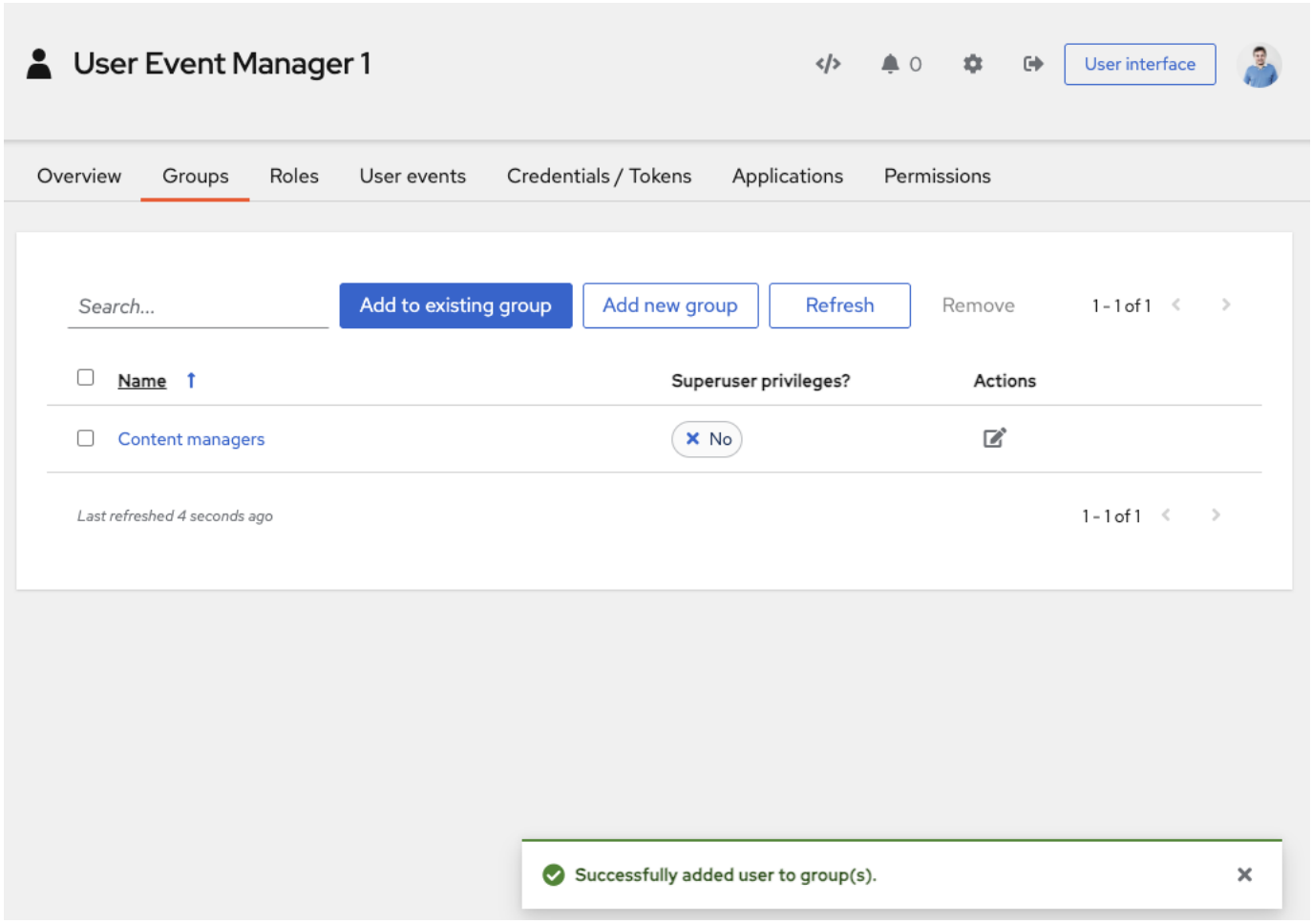
1. Добавляем группу



1. Выбираем имеющуюся группу "Content managers"



1. Сохраняем и возвращаемся во вкладку групп пользователя



1. Последний шаг: добавляем пароль пользователю

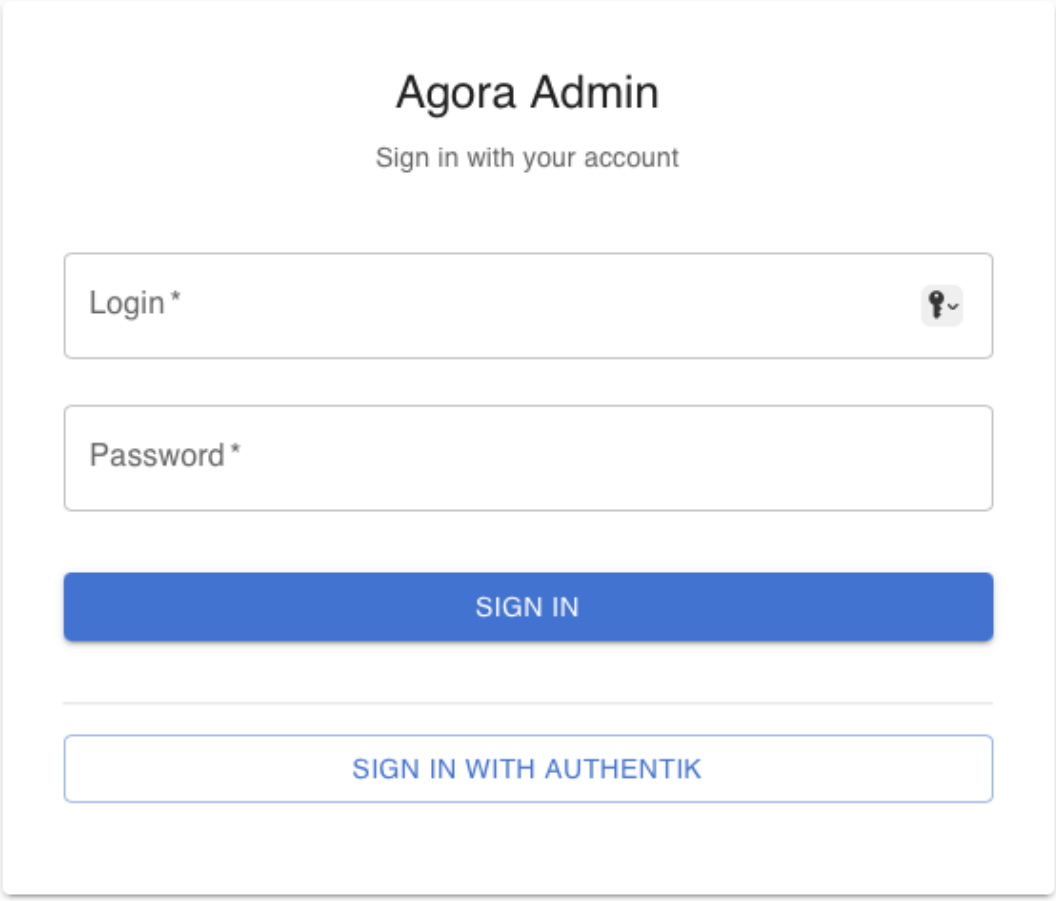
1.2. НАСТРОЙКА AGORA

Для того, чтобы Agora могла работать по OIDC с IdP надо настроить следующие переменные окружения:

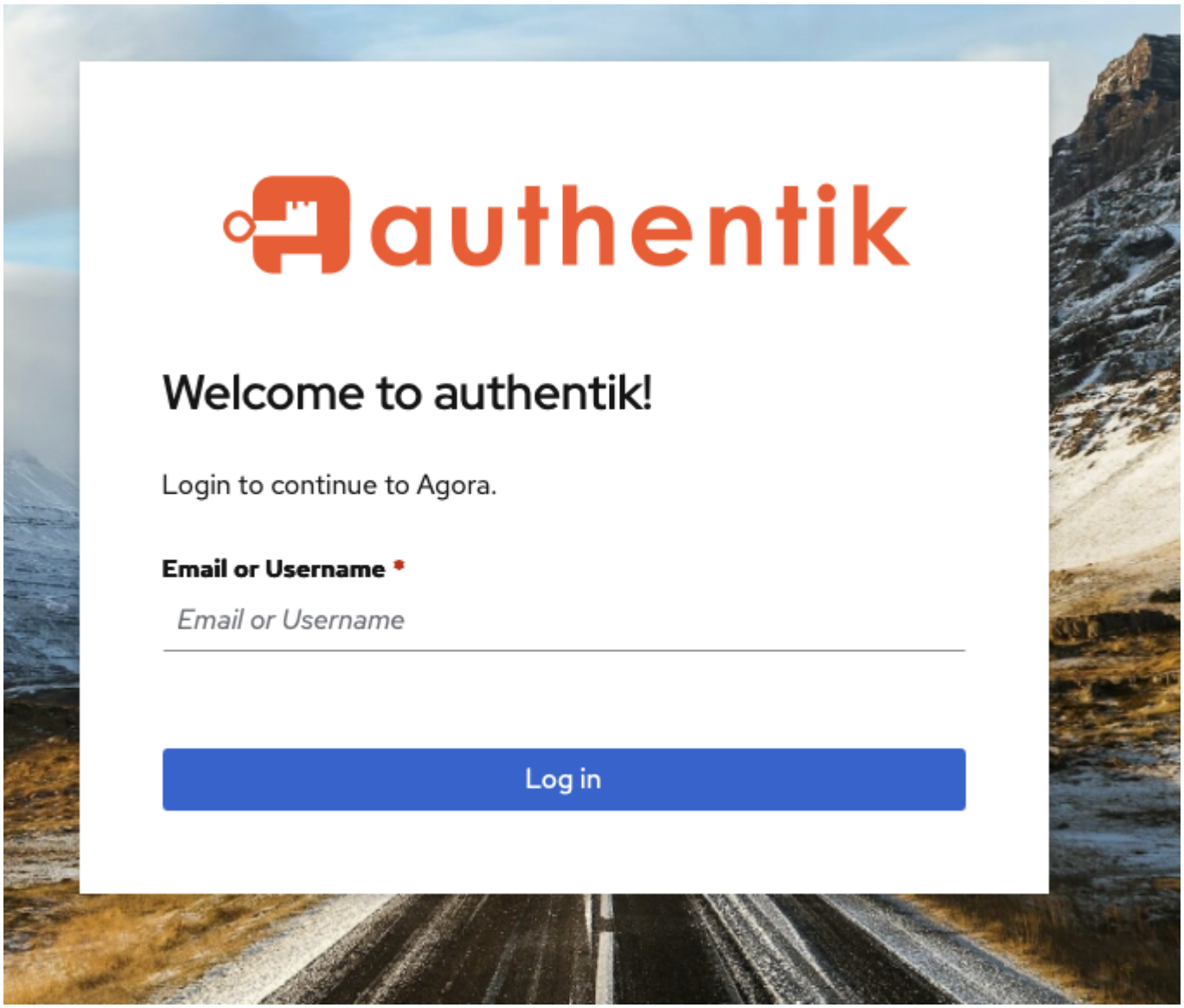
- `OIDC_AUDIENCE=agora-admin-api` - этот параметр мы ожидаем в **aud** клейме внутри JWT, пришедшего от IdP
- `OIDC_CLIENT_ID=agora-admin-api` - здесь мы списываем по сути логин от записи о нашем приложении в IdP
- `OIDC_CLIENT_SECRET=oCSarud3XmFrYyjVl....ko1VW8n2O3SJLHfwCoTiy` это пароль от записи о нашем приложении
- `OIDC_SCOPES="openid profile offline_access"` эта настройка опциональная, здесь выписаны те дефолты, которые уже есть в Agora. offline_access нужен для того, чтобы можно было обновлять авторизационный токен
- `OIDC_ISSUER=http://authentik.local:9000/application/o/agora/` - надо вписать адрес вашего сервера, например того authentik, который мы настроили в примере выше. Важно, что в этом пути `agora` - это соглашение самого authentik, на то, что Slug приложения будет участвовать в пути к Issuer ресурсу
- `OIDC_TITLE=Authentik` - сюда нужно вписать что захотите, это то сообщение, которое будет выводиться пользователям. Без этой опции OIDC не включается.
- `OIDC_ROLE_CLAIM=groups` - опциональная настройка, указывает в каком поле внутри JWT прийдет список групп пользователя
- `OIDC_ROLE_MAP='{"authentik Admins":"administrator","Content managers":"content_manager","Sysops":"monitoring"}'` - важное и сложное поле, без него ничего не заработает. В нём указывается, как превращаются группы из IdP в роли внутри Agora (см дальше список ролей)
- `LOCAL_LOGIN_ENABLED=false` - можно опционально выключить авторизацию по логину-паролю и оставить только через IdP

1.3. ШАГИ ПОЛЬЗОВАТЕЛЯ СИСТЕМЫ

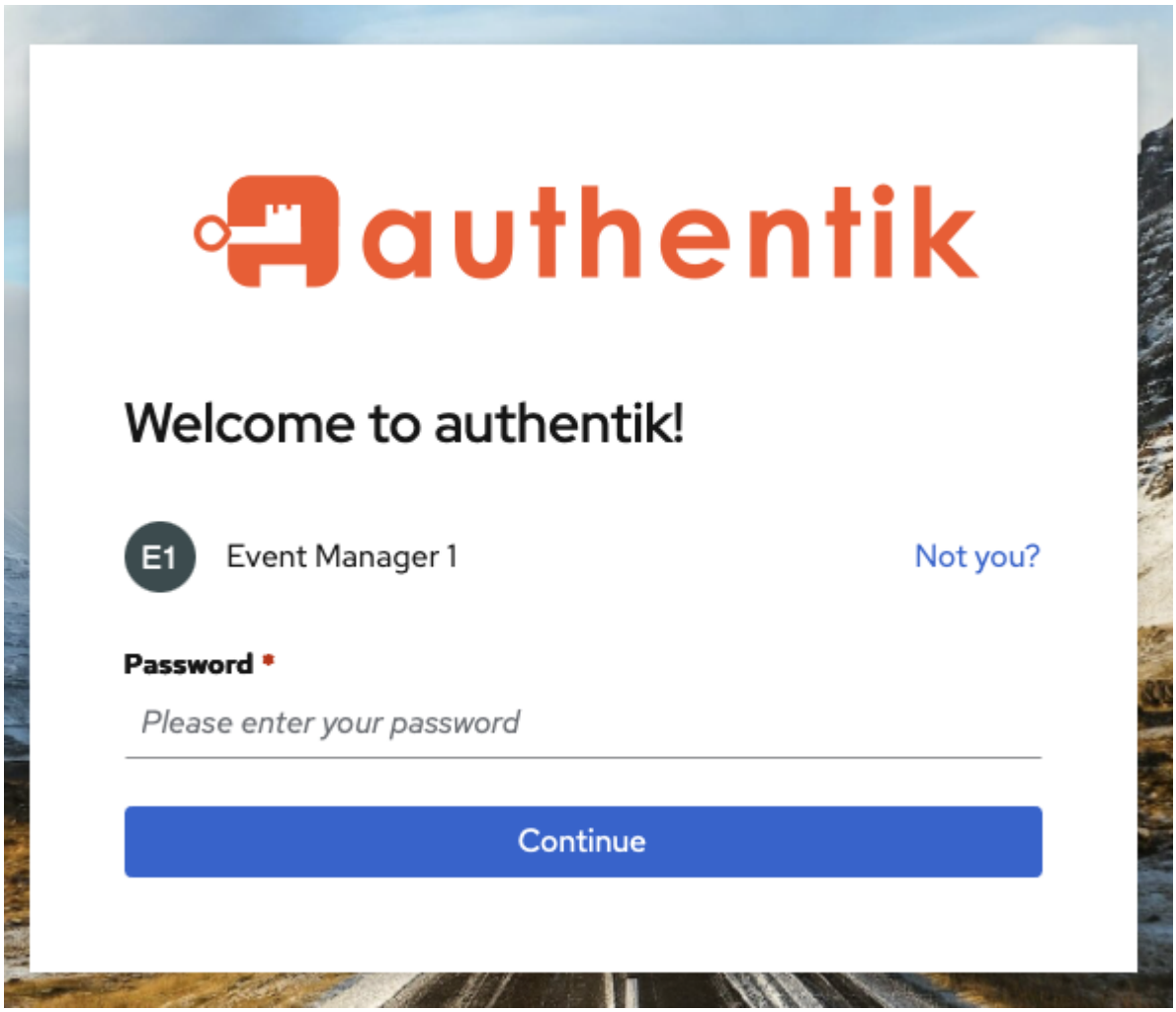
1. Пользователь заходит на экран логина и видит два варианта авторизации: локальный и через OIDC



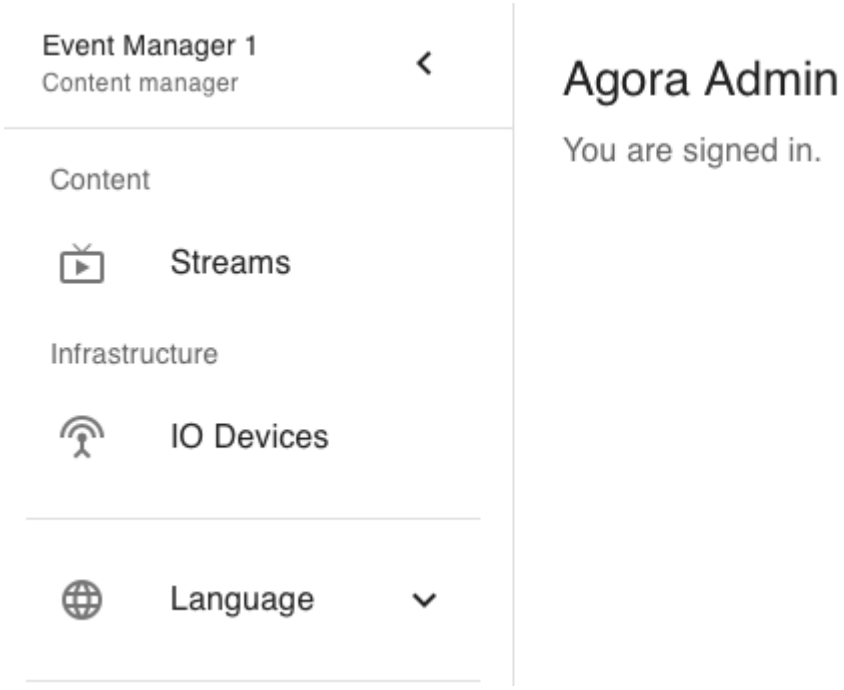
1. Выбирает вход через IdP и попадает в приложение Authentik



1. Вводит свой логин (конкретно в этом примере "Event Manager 1") и переходит на экран ввода пароля



1. После успешного ввода пароля пользователь возвращается на экран Agora и автоматически логинится в приложение



Пользователь успешно создан в системе Agora.

2. Детали настройки

2.1. РОЛИ В IDP И СОПОСТАВЛЕНИЕ С AGORA

Ниже практический алгоритм для администратора, как настроить сопоставление ролей IdP -> Agora.

Agora берет значение роли из **access token**:

- имя claim (поля в JWT токене) задает переменная окружения `OIDC_ROLE_CLAIM`;
- сопоставление строк из токена с ролями Agora задает `OIDC_ROLE_MAP`.

Допустимые роли в Agora:

- `administrator`;
- `content_manager`;
- `monitoring`;
- `security`.

Если для нового пользователя не найдено ни одного совпадения в `OIDC_ROLE_MAP`, вход завершается `403`, учетная запись по JIT схеме не создается.

Если учетная запись уже существует и в новом токене нет маппируемой роли, текущая `Account.role` не меняется.

Пошаговая настройка

1. Определите, из какого claim будете брать роль.

Чаще всего это `groups` (массив строк), реже отдельный строковый claim.

1. Настройте IdP, чтобы нужный claim попадал в **access token**.

Для Authentik обычно делают Scope/Property Mapping и подключают его к нужному OAuth2/OpenID Provider, но из коробки всё настроено на `groups` и ничего делать не надо.

1. Убедитесь, что значения в claim стабильны и удобны для маппинга.

Пример значений групп: `agora-administrator`, `agora-content-manager`, `agora-monitoring`, `agora-security`. Так же можно использовать более человеко-читаемые, как в примере выше: `Content managers`

1. Настройте переменные Agora.

```
OIDC_ROLE_CLAIM=groups
OIDC_ROLE_MAP='{"agora-administrator":"administrator","agora-content-manager":"content_manager","agora-monitoring":"monitoring","agora-security":"security"}'
```

1. Перезапустите Agora, выполните новый login через IdP и проверьте результат.

Проверка и отладка

В логах Agora вы увидите декодированный токен:


```
{"level":"info","token_endpoint":"http://localhost:9000/application/o/token/","has_refresh_token":true,"expires_in":300,"access_token_payload":"{\"iss\":
\"http://localhost:9000/application/o/agora/\",\"sub\":\"4b34f838bad570b486a2168b5353666fc19a77a230f9df03aff8c1bf7c1baf77\",\"aud\":\"agora-admin-api\",
\"exp\":1775037446,\"iat\":1775037146,\"auth_time\":1775037103,\"acr\":\"goauthentik.io/providers/oauth2/default\",\"amr\":[\"pwd\"],\"sid\":
\"1799899e3648369f24a47c987310fd96a5c5678dce14bbdcf61ce67c4e61b430\",\"jti\":\"oUgCDmUr9jtxiBn2zN4N0ExsZV4rptZ8eUuajmOy\",\"name\":\"\",\"given_name\":\"\",
\"preferred_username\":\"Event Manager 1\",\"nickname\":\"Event Manager 1\",\"groups\":[\"Content managers\"],\"azp\":\"agora-admin-api\",\"uid\":
\"yfgs9bVO0Lg2Okc3c0HdDjN5OxxGG1SlHen4b1l4\",\"scope\":\"openid profile offline_access\"}","time":1775037146,"message":"oidc: token response received"}
{"level":"info","jwks_url":"http://localhost:9000/application/o/agora/jwks/","keys":1,"ttl":3600000,"time":1775037146,"message":"oidc: jwks cache refreshed"}
{"level":"info","issuer":"http://localhost:9000/application/o/
agora/","subject":"4b34f838bad570b486a2168b5353666fc19a77a230f9df03aff8c1bf7c1baf77","jti":"oUgCDmUr9jtxiBn2zN4N0ExsZV4rptZ8eUuajmOy","exp":1775037446,"time":
1775037146,"message":"oidc: access token verified"}
```


Можно проверить, что именно приходит в поле `groups`

Проверьте логи backend:

- при проблеме маппинга будет `oidc login: no mappable role for JIT user`;
- при успешном маппинге увидите успешный `POST /login/oidc` и создание/вход в учетку.

Для shell-конфигураций используйте `OIDC_ROLE_MAP` как одну JSON-строку в кавычках (как в примере выше), чтобы избежать искажения формата при `source .env`.

1.6. СЕССИИ В AGORA И ОТЗЫВ

Agora работает по паттерну **BFF (Backend for Frontend)**: браузер не хранит и не отправляет JWT IdP в API. Браузер использует обычный session token Agora (opaque), а backend сам работает с access/refresh token IdP.

Подход соответствует рекомендациям IETF для browser-based приложений: OAuth 2.0 for Browser-Based Applications (BFF).

Идентификатор сессии Agora (`session_id`) для OIDC-логина инициализируется из первого `jti` access token IdP и дальше остается стабильным для этой локальной сессии.

- **`GET /sessions`**, **`GET /sessions/{id}`** — **`{id}`** это **`jti`**.
- **`session_logout`** — завершение по **`jti`**.

Для OIDC-сессии logout закрывает локальную сессию Agora. Если у IdP объявлен `revocation_endpoint`, backend дополнительно отправляет revoke для refresh token (best effort).

После закрытия сессии Agora отвечает **401** по этому session token.

1.7. ПРОВЕРКА ПОСЛЕ СВЯЗКИ С IDP

1. **`OIDC_ISSUER`**, **`OIDC_AUDIENCE`**, JWKS совпадают с реальным access token.
2. Логин через OIDC в UI и успешный вход в админку.
3. JIT: новый **`sub`** и маппируемая роль в токене → учётка в MongoDB, верная роль; без маппируемой роли → **403**, учётки нет.
4. Отзыв сессии по **`jti`**.

2. Какие поля в access token IdP использует backend Agora

Ниже — поля **access token IdP**, которые backend Agora проверяет и использует в OIDC-flow. Браузер в API Agora JWT не отправляет.

Claim / аспект	Назначение для Agora
`iss`	Должен совпадать с **`OIDC_ISSUER`**.
`aud`	Должен содержать **`agora-admin-api`** (строка или элемент массива).
`sub`	Стабильный идентификатор пользователя у IdP; связывается с **`external_account_id`** в MongoDB.
`jti`	**Обязателен**; идентификатор сессии в API и в аудите (**`session_id`**). Без **`jti`** — **401**.
`exp`	Срок действия токена; проверка с **leeway 120 с** (допуск по часам).
`nbf`	Не обязателен; если есть — «не раньше» с тем же **leeway 120 с**.
`iat`	Стандартное время выдачи; может использоваться вспомогательно.
Claim из **`OIDC_ROLE_CLAIM`**	Строка или массив строк (например группы); маппинг в роль Agora через **`OIDC_ROLE_MAP`**.
`preferred_username`, **`email`**	Участвуют в выборе **логина** при JIT (вместе с **`sub`**).

Дополнительно Agora использует **заголовок** JWT **`kid`** при выборе ключа из JWKS и кэшировании ключей (**`OIDC_JWKS_CACHE_TTL`**, повторная загрузка JWKS при сбое проверки подписи — по логике вашей сборки).

4. Глоссарий

- **OIDC** — OpenID Connect. Протокол входа и выдачи **JWT** без постоянных запросов приложения к каталогу при каждом действии пользователя.
- **JWT** — JSON Web Token. Подписанное сообщение с полями (**claims**): кто пользователь, до когда токен действителен, для кого он выдан. Содержимое видно, подделать без ключа нельзя.
- **IdP** — Identity Provider. Центральный корпоративный каталог, хранящий пользователей и политики доступа; выдаёт токены клиенту после входа.
- **Claim** — одно именованное поле внутри JWT (`sub` , `aud` , `groups` и т.д.).
- **Access token** — JWT IdP, который backend Agora использует для валидации личности и ролей. Браузер этот токен в API Agora не отправляет.
- **ID token** — JWT «о факте входа» в браузере. Для админского API Agora **не** используется (другой `aud` и назначение).
- **JWKS** — JSON Web Key Set. Публичные ключи **IdP** (отдельным URL); по ним Agora проверяет подпись access token. Например этот список запрашивается у IdP.
- **RS256** — подпись JWT алгоритмом RSA-SHA256. Типично у access token, который выдает IdP.
- **OAuth2 / OpenID Provider** — часть IdP, которая по протоколу OAuth2/OIDC отдаёт клиенту **access** (и при необходимости **ID**) токены.
- **Discovery** — стандартный JSON по адресу `{issuer}/.well-known/openid-configuration` : оттуда берут `issuer` , `jwks_uri` и прочие конечные точки.
- **Scope** — набор прав/данных, который **клиент запрашивает** при авторизации; от настроек scope зависит, какие **claims** попадут в токен.
- **Audience (`aud`)** — в JWT указано, **для какого приложения** выдан токен. Для Agora в access token должно быть `agora-admin-api` (или это значение в списке `aud`).
- **Bearer** — схема в HTTP: `Authorization: Bearer <токен>` . В нашей схеме браузер передает Bearer session token Agora, а не JWT IdP.
- **JIT** (Just-In-Time) — первый успешный запрос к Agora с валидным токеном IdP **создаёт** учётную запись в базе, **если** её ещё не было **и** в токене есть **маппируемая роль**; иначе учётку **не** создают (**403**).
- **Property Mapping** (в Authentik) — правило «какие поля добавить в **access token**» (например список групп); без маппинга группы могут не попасть в JWT.
- `iss` **(issuer)** — кто выдал JWT (URL **IdP**). Agora по `iss` выбирает нужный **JWKS** для проверки подписи.
- `sub` **(subject)** — неизменяемый идентификатор пользователя у **IdP**; в Agora хранится как связь с внешним аккаунтом (`external_account_id`).
- `jti` **(JWT ID)** — уникальный номер выпуска access token IdP. Для OIDC-сессии Agora использует `jti` первого успешного токена как стабильный `session_id` .
- `exp` / `nbf` — до какого момента токен действителен и (если есть) с какого момента его уже можно принимать.
- **SSO** — single sign-on: пользователь один раз аутентифицируется в **IdP**, приложения используют выданные токены.
- **End session** — URL у **IdP** для завершения сессии в браузере («выйти везде»), в отличие от простого сброса токена на стороне клиента.

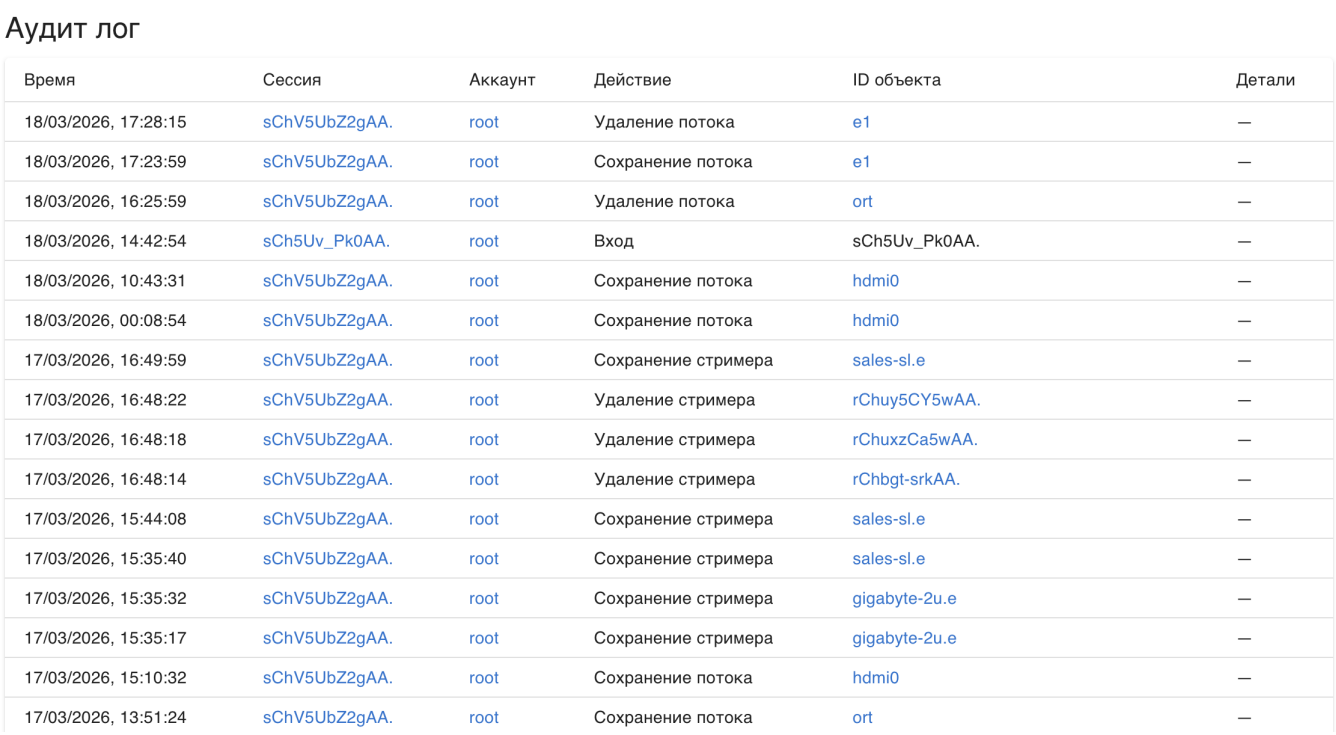
2.4.3 Журнал безопасности

Журнал аудита в Agora используется для контроля административных действий, расследования инцидентов и последующего анализа событий безопасности. Через этот раздел можно просматривать историю операций, выполненных пользователями в системе.

Практический смысл журнала аудита

Журнал аудита нужен для того, чтобы восстановить историю действий в системе, понять кто и когда выполнял операции, а также использовать эти данные для расследования инцидентов, внутреннего контроля и последующего security-анализа.

Что показывает журнал аудита



Аудит лог

Время	Сессия	Аккаунт	Действие	ID объекта	Детали
18/03/2026, 17:28:15	sChV5UbZ2gAA.	root	Удаление потока	e1	—
18/03/2026, 17:23:59	sChV5UbZ2gAA.	root	Сохранение потока	e1	—
18/03/2026, 16:25:59	sChV5UbZ2gAA.	root	Удаление потока	ort	—
18/03/2026, 14:42:54	sCh5Uv_Pk0AA.	root	Вход	sCh5Uv_Pk0AA.	—
18/03/2026, 10:43:31	sChV5UbZ2gAA.	root	Сохранение потока	hdmi0	—
18/03/2026, 00:08:54	sChV5UbZ2gAA.	root	Сохранение потока	hdmi0	—
17/03/2026, 16:49:59	sChV5UbZ2gAA.	root	Сохранение стримера	sales-sl.e	—
17/03/2026, 16:48:22	sChV5UbZ2gAA.	root	Удаление стримера	rChuy5CY5wAA.	—
17/03/2026, 16:48:18	sChV5UbZ2gAA.	root	Удаление стримера	rChuxzCa5wAA.	—
17/03/2026, 16:48:14	sChV5UbZ2gAA.	root	Удаление стримера	rChbgt-srkAA.	—
17/03/2026, 15:44:08	sChV5UbZ2gAA.	root	Сохранение стримера	sales-sl.e	—
17/03/2026, 15:35:40	sChV5UbZ2gAA.	root	Сохранение стримера	sales-sl.e	—
17/03/2026, 15:35:32	sChV5UbZ2gAA.	root	Сохранение стримера	gigabyte-2u.e	—
17/03/2026, 15:35:17	sChV5UbZ2gAA.	root	Сохранение стримера	gigabyte-2u.e	—
17/03/2026, 15:10:32	sChV5UbZ2gAA.	root	Сохранение потока	hdmi0	—
17/03/2026, 13:51:24	sChV5UbZ2gAA.	root	Сохранение потока	ort	—

В списке audit-событий отображаются:

- время выполнения операции;
- идентификатор сессии;
- учетная запись, выполнившая действие;
- тип события;
- объект, к которому относится операция;
- дополнительные данные события в `payload`.

Если событие связано с известным объектом системы, интерфейс позволяет перейти по ссылке:

- к карточке сессии;
- к карточке пользователя;
- к потоку;
- к стримеру.

Это позволяет быстро переходить от отдельной записи аудита к связанному объекту и восстанавливать контекст произошедшего.

Контролы поиска и фильтрации

Для работы с журналом аудита предусмотрены контролы:

- выбора даты или временного диапазона;
- фильтрации по типам событий.

Такие фильтры позволяют:

- быстро выделить события за нужный период;
- анализировать только интересующие типы операций;
- отделять действия пользователей, изменения конфигурации и security-события друг от друга.

Типы событий

В журнале аудита фиксируются разные классы операций, включая:

- вход и выход пользователей;
- создание, изменение и удаление потоков;
- создание, изменение и удаление стримеров;
- операции над учетными записями, включая блокировку.

Состав типов событий может расширяться по мере развития системы и backend API.

Постраничная загрузка журнала

Журнал аудита загружается порциями. Если записей много, интерфейс позволяет последовательно подгружать следующие записи журнала.

Такой подход нужен для:

- работы с большими объемами событий;
- снижения нагрузки на интерфейс;
- удобного последовательного просмотра истории операций.

Связь с сессиями безопасности

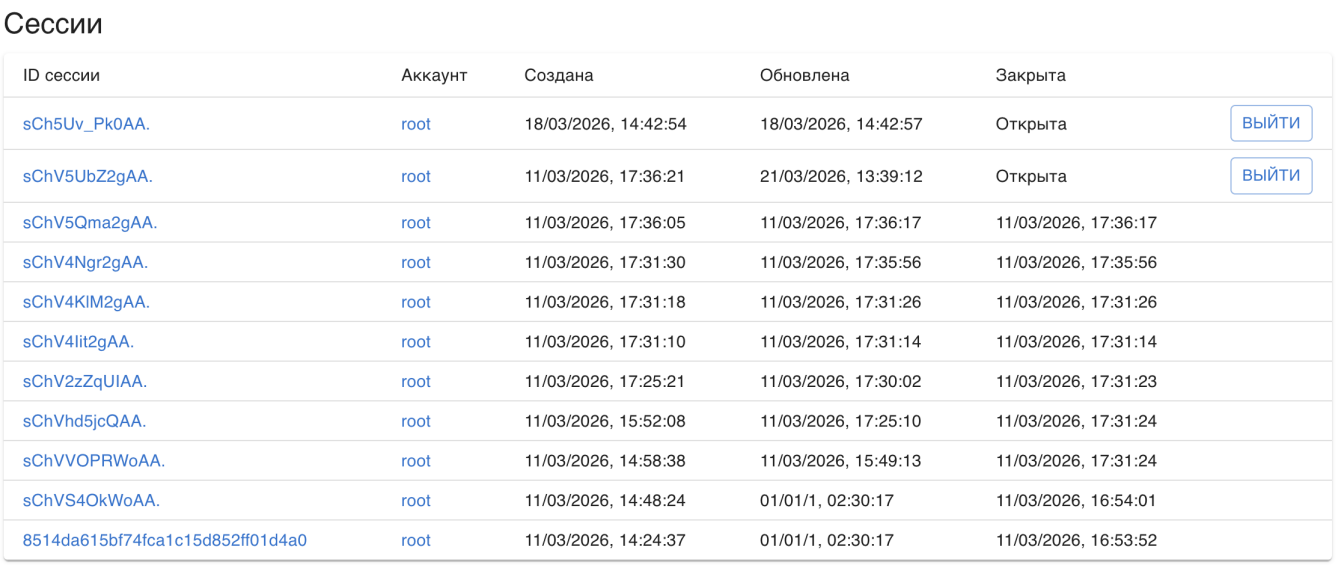
Журнал аудита тесно связан с разделом пользовательских сессий. Для каждой сессии можно просматривать связанные операции, а из журнала аудита можно перейти к конкретной сессии и продолжить анализ уже в ее контексте.

Это особенно полезно при расследовании подозрительных действий, проверке административных изменений и анализе действий конкретного пользователя в рамках одного входа в систему.

2.4.4 Сессии пользователей в Agora

Сессии пользователей в Agora используются для контроля доступа, мониторинга активности и расследования событий безопасности. Через раздел сессий администратор или сотрудник безопасности может просматривать активные и завершенные сеансы работы пользователей, а также завершать открытые сессии при необходимости.

Список сессий



На странице списка сессий отображаются:

- идентификатор сессии;
- пользователь, которому принадлежит сессия;
- время создания сессии;
- время последнего обновления;
- время закрытия сессии или признак того, что сессия еще открыта.

Из списка можно:

- перейти в карточку конкретной сессии;
- перейти в карточку пользователя, которому принадлежит сессия;
- принудительно завершить открытую сессию через logout.

Если сессия уже закрыта, действие logout для нее недоступно.

Завершение открытой сессии

Для принудительного завершения активной сессии:

1. Найдите нужную сессию в списке.
2. Нажмите кнопку `logout`.
3. Дождитесь подтверждения успешного завершения.

После завершения сессия перестает считаться открытой и обновляется в списке.

Все дальнейшие действия с этой с

Карточка сессии

В карточке отдельной сессии отображаются:

- `session_id`;
- пользователь, связанный с сессией;
- время создания;
- время последнего обновления;
- время закрытия или статус открытой сессии.

Из карточки сессии можно перейти обратно к общему списку или открыть карточку соответствующего пользователя.

Аудит событий по сессии

Для каждой сессии Agora показывает связанные audit-события. Это позволяет:

- увидеть действия, выполненные в рамках конкретной сессии;
- сопоставить операции с конкретным пользователем;
- анализировать последовательность действий в рамках одного входа в систему.

Таким образом, страница сессии используется не только для контроля активных логинов, но и как точка входа в расследование инцидентов.

Сбор IP-адресов

В рамках security-контроля Agora собирает IP-адреса пользовательских подключений и сессий.

Эти данные используются для:

- анализа источника входа;
- расследования подозрительной активности;
- сопоставления действий пользователя с конкретным сетевым адресом;
- последующего аудита и реагирования на инциденты безопасности.

2.4.5 Сетевые взаимодействия

Agora проектируется для работы в корпоративной сети с разделением ролей компонентов и сетевых сегментов. В типовой инсталляции сетевые взаимодействия строятся так, чтобы:

- административный трафик был отделен от медиатрафика;
- внешние источники взаимодействовали только с `Ingress сервер`;
- внутренние мастер-узлы не были напрямую доступны из пользовательских сетей;
- доставка контента конечным пользователям выполнялась через `edge сервера`.

Основные сетевые сегменты

В корпоративной инсталляции обычно выделяются следующие сегменты:

- административный сегмент для доступа к `контроллеру`;
- внутренний медиасегмент для `origin`, `vod transcoder`, стримеров и ретрансляторов;
- DMZ для `Ingress сервер`, если система принимает внешние live-источники;
- пользовательский сегмент или сегмент филиалов, где располагаются `edge сервера` и клиентские устройства;
- сегмент хранения для `storage система`.

Такое разделение снижает риск прямого доступа к внутренним медиасерверам и упрощает применение корпоративных политик безопасности.

Основные сетевые взаимодействия

КОНТРОЛЛЕР И СТРИМЕРЫ / РЕТРАНСЛЯТОРЫ

`Контроллер` взаимодействует со стримерами и ретрансляторами по `HTTP` или `HTTPS`.

Обычно используются:

- порт `80` для незащищенного `HTTP`;
- порт `443` для `HTTPS`.

Этот трафик используется для управления, получения статусов, применения конфигурации и мониторинга состояния узлов.

МЕЖДУ СТРИМЕРАМИ

Между стримерами возможны следующие виды взаимодействия:

- `HTTP` / `HTTPS` для служебного и publish-взаимодействия;
- `multicast` для сценариев `Standby Push`.

Для `Standby Push` backup-стример должен получать `multicast` с primary-стримера. Такой режим предполагает локальную сеть с гарантированно низкой задержкой.

МЕЖДУ РЕТРАНСЛЯТОРАМИ И СТРИМЕРАМИ

Ретрансляторы получают видео со стримеров по:

- `HTTP`;
- `HTTPS`.

В зависимости от сценария отказоустойчивости ретранслятор может подключаться к одному или нескольким стримерам.

МЕЖДУ INGRESS СЕРВЕР И ВНЕШНИМИ ИСТОЧНИКАМИ

`Ingress сервер` принимает внешние источники по тем протоколам, которые поддерживаются конкретным сценарием приема. Обычно используются:

- `HTTP`;
- `HTTPS`;
- `RTMP` на порту `1935`;
- `SRT` на согласованных портах;
- `WebRTC` с заранее зафиксированным набором портов, если это требуется политиками сети.

Для `SRT` и `WebRTC` конкретные диапазоны портов определяются при внедрении и должны быть отдельно согласованы с сетевой службой заказчика.

МЕЖДУ INGRESS СЕРВЕР И ВНУТРЕННИМИ СТРИМЕРАМИ

Если роли `Ingress сервер` и внутренних стримеров разнесены, между ними используется:

- `HTTP`;
- `HTTPS`.

Этот трафик должен быть разрешен только между доверенными внутренними узлами, поскольку через него поток попадает из DMZ во внутренний медиасегмент.

Типовые требования к межсетевому экранированию

При настройке сетевых политик обычно требуется:

- разрешить административный доступ к `контроллеру` только из администраторского сегмента;
- разрешить прием внешнего видеотрафика только на `Ingress сервер`;
- запретить прямой доступ внешних источников к `origin` и другим внутренним медиасерверам;
- разрешить внутренние соединения между стримерами, ретрансляторами и `edge сервера` только по согласованным протоколам;
- ограничить доступ к `storage система` только доверенными компонентами платформы.

Практические замечания

При проектировании сетевых взаимодействий важно учитывать:

- для `Twincast` нужны два независимых тракта приема;
- для `Standby Push` требуется локальная сеть с низкой задержкой и доставка `multicast` между стримерами;
- для приема внешних источников в изолированном контуре предпочтительно выносить `Ingress сервер` в DMZ;
- перечень разрешенных портов и протоколов должен фиксироваться в проектной документации внедрения.