

Catena Manual

Catena

Flussonic

© Flussonic 2025

Table of contents

1. Products	3
2. Catena	4
2.1 Управление порталами	6
2.2 Управление менеджерами портала	21
3. Управление контентом	36
3.1 Управление телевизионными каналами	36
3.2 Управление пакетами каналов	44
3.3 Управление источниками EPG	54
4. Управление абонентами	66
4.1 Управление абонентами	66
4.2 Управление подписками	78
5. Мониторинг	93
5.1 Мониторинг сеансов воспроизведения	93
5.2 Журнал операций	106
6. Клиентские приложения	123
6.1 Руководство пользователя Android-приложения Catena	123

1. Products

2. Catena

2.0.1 Catena

Catena - это комплексное решение для управления IPTV-сервисами, состоящее из серверного ПО и клиентских приложений для Smart TV, мобильных устройств и приставок. Система предназначена для доставки платного и бесплатного телевидения конечным абонентам.

Клиентские приложения Catena доступны для следующих платформ:

- **Samsung Tizen** — приложение для Smart TV Samsung
- **LG WebOS** — приложение для Smart TV LG
- **Android** — приложение для Android TV, мобильных устройств и планшетов

Какой-либо контент в поставку Catena не входит — он всегда предоставляется оператором видеостримингового сервиса.

Для кого предназначена Catena

Catena разработана для **операторов видеостриминговых сервисов**, которые хотят запустить собственный IPTV-сервис. Это может быть:

- **Телекоммуникационные операторы** — провайдеры интернет-услуг, желающие предложить своим абонентам услугу интерактивного телевидения
- **OTT-операторы** — компании, предоставляющие видеоконтент через интернет
- **Кабельные операторы** — традиционные ТВ-операторы, переходящие на IP-технологии
- **Корпоративные клиенты** — организации, создающие собственные внутренние ТВ-сервисы для сотрудников или клиентов
- **Медиакомпании** — издатели и вещатели, желающие создать прямую связь со своей аудиторией

Основные возможности Catena

Catena предоставляет полный набор инструментов для управления IPTV-сервисом:

УПРАВЛЕНИЕ КОНТЕНТОМ

- **Управление телевизионными каналами** — создание, настройка и организация каналов вещания. Каждый канал имеет уникальное имя для стриминга, отображаемое название для пользователей, логотип и привязку к источнику EPG
- **Управление пакетами каналов** — формирование тарифных пакетов из групп каналов. Пакеты используются для продажи наборов каналов абонентам
- **Управление программой передач (EPG)** — подключение и синхронизация источников электронной программы передач, отображение расписания программ для каждого канала

УПРАВЛЕНИЕ АБОНЕНТАМИ

- **Управление абонентами** — регистрация подписчиков, управление их данными (имя, телефон), генерация токенов для воспроизведения контента
- **Управление подписками** — назначение пакетов каналов абонентам, управление доступом к контенту, поддержка как платных, так и бесплатных пакетов

МОНИТОРИНГ И АНАЛИТИКА

- **Сессии воспроизведения** — просмотр текущих и исторических сессий просмотра: какие каналы смотрят абоненты, с каких устройств, сколько данных передано
- **Журнал операций** — детальная история всех действий в системе (создание/удаление абонентов, изменение подписок и т.д.) с возможностью фильтрации и аудита

АДМИНИСТРИРОВАНИЕ

- **Настройки портала** — конфигурация параметров сервиса: название, домен, брендинг (логотип, описание), управление API-ключами, настройка бесплатных пакетов
- **Управление менеджерами** — создание учетных записей администраторов с разными уровнями доступа: управление инфраструктурой, управление абонентами, управление контентом

Веб-интерфейс и API

Catena предоставляет два способа управления системой:

ВЕБ-ИНТЕРФЕЙС (UI)

Через веб-интерфейс администратор может:

- **Визуально управлять всеми сущностями** — создавать, редактировать и удалять каналы, пакеты, абонентов через удобный графический интерфейс
- **Просматривать статистику в реальном времени** — видеть активные сессии просмотра, количество абонентов, популярность каналов
- **Загружать логотипы и изображения** — добавлять визуальные элементы для каналов и портала
- **Управлять программой передач** — просматривать и обновлять EPG, связывать каналы с источниками телепрограммы
- **Администрировать доступ** — создавать менеджеров с разными правами, управлять API-ключами
- **Отслеживать историю операций** — просматривать журнал всех действий для аудита и контроля

ТИПИЧНЫЕ СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

Оператор малого масштаба:

- Использует преимущественно веб-интерфейс для ручного управления небольшим количеством абонентов (до нескольких тысяч)
- Вручную создает каналы и пакеты
- Регистрирует абонентов через UI при обращении в службу поддержки

Оператор среднего и крупного масштаба:

- Использует API для автоматической синхронизации с биллингом и CRM
- Автоматизирует создание/блокировку абонентов при изменении статуса оплаты
- Использует веб-интерфейс для мониторинга, аналитики и ручных операций
- Настраивает автоматическое обновление EPG через API

2.1 Управление порталами

Портал — это независимый брендированный домен с собственным набором абонентов, каналов и пакетов в системе Catena. Порталы позволяют управлять несколькими IPTV-сервисами с различными брендами на одной инфраструктуре.

2.1.1 Что такое портал

Портал в Catena — это изолированное пространство для отдельного IPTV-сервиса со своими настройками, абонентами и контентом. Каждый портал представляет собой независимый брендированный сервис с собственным доменом, логотипом и визуальным оформлением.

Ключевая концепция:

```

Catena инфраструктура
├── Портал "Netflix-подобный сервис" (myiptv.com)
│   ├── Абоненты: 10,000
│   ├── Каналы: 200
│   └── Брендинг: красный логотип, современный дизайн
├── Портал "Региональный провайдер" (region-tv.ru)
│   ├── Абоненты: 5,000
│   ├── Каналы: 150
│   └── Брендинг: синий логотип, классический стиль
└── Портал "Спортивный сервис" (sport-tv.com)
    ├── Абоненты: 3,000
    ├── Каналы: 50 (только спорт)
    └── Брендинг: зелёный логотип, динамичный дизайн
  
```

Основные характеристики портала:

- **Независимый домен** — каждый портал доступен по своему URL
- **Собственный брендинг** — логотип, название, описание, визуальное оформление
- **Изолированные данные** — абоненты одного портала не видны в другом
- **Отдельный контент** — свой набор каналов, пакетов, EPG
- **Брендированные приложения** — возможность создать mobile app для портала
- **Общая инфраструктура** — все порталы работают на одних серверах

Зачем нужны порталы:

1. **Мультибрендинг** — управление несколькими IPTV-брендами
2. **White-label решения** — предоставление сервиса под брендом клиента
3. **Географическое разделение** — разные порталы для разных регионов
4. **Сегментация аудитории** — premium и budget сервисы на одной платформе
5. **Партнёрские проекты** — отдельные порталы для B2B партнёров
6. **Тестовая среда** — отдельный портал для тестирования нововведений

2.1.2 Управление несколькими порталами

Концепция мультитенантности

Одна инфраструктура — множество порталов:

- Все порталы используют одни и те же streaming-серверы
- Контент может быть как общим, так и уникальным для портала
- Один менеджер может управлять несколькими порталами
- Billing-система может обслуживать все порталы

Преимущества:

- **Экономия ресурсов** — один сервер для всех порталов
- **Централизованное управление** — единая панель администратора
- **Общий контент** — одни каналы для разных брендов
- **Гибкий маркетинг** — разные стратегии для каждого портала
- **Масштабируемость** — легко добавлять новые порталы

Один менеджер — несколько порталов

Сценарий: Владелец IPTV-бизнеса управляет тремя брендами

```
Менеджер: admin@company.com
├─ Портал 1: myiptv.com (owner, полный доступ)
├─ Портал 2: premium-tv.com (owner, полный доступ)
└─ Портал 3: budget-tv.com (content admin, управление контентом)
```

Как работает доступ:

- Менеджер создаётся отдельно для каждого портала
- Один email может использоваться в разных порталах
- При входе система показывает список доступных порталов
- Менеджер выбирает портал для работы
- API ключ привязан к конкретному portalу

Типы прав менеджера:

- **isAdmin** — управление инфраструктурой (создание порталов, серверов)
- **canManage** — полное управление порталом (owner)
- **canManageSubscribers** — управление абонентами и подписками
- **canManageContent** — управление каналами, пакетами, EPG

2.1.3 Основные параметры портала

Технические параметры**Идентификатор портала (Portal ID)**

- Автоматически генерируется при создании портала
- Формат: base64-кодированный Snowflake ID с заменой `+/=` на `-_.`
- Пример: `pK19SW3AAAE.`
- Используется во всех API запросах
- Связывает все сущности (абонентов, каналы, пакеты) с порталом

Внутреннее имя (Name)

- Техническое название портала в системе
- Видно только администраторам и владельцу портала
- Используется для идентификации в логах и панели управления
- Должно быть уникальным в системе
- Примеры: `catena-netflix`, `my-iptv-service`, `test-portal`

Домен (Domain)

- Доменное имя, под которым доступен портал
- Указывается владельцем портала как заявка на домен
- Реальная привязка DNS выполняется администратором системы
- Используется для брендированных мобильных приложений
- Пример: `myiptv.com`, `tv.example.org`

ID владельца (Owner ID)

- Идентификатор менеджера — владельца портала
- Владелец имеет полный доступ ко всем настройкам
- Может назначать других менеджеров
- Устанавливается при создании портала

API ключ (API Key)

- Уникальный ключ для доступа к Management API портала
- Генерируется автоматически при создании портала
- Используется для аутентификации всех API запросов
- Может быть регенерирован через `/portal/reset_api_key`
- Должен храниться в безопасности

Параметры брендинга**Логотип (Logo)**

- URL или base64-encoded изображение логотипа портала
- Отображается в мобильных приложениях и веб-интерфейсе
- Видят конечные пользователи (абоненты)
- Рекомендуемый формат: PNG с прозрачностью
- Рекомендуемый размер: 512x512px или выше

Название (Title)

- Публичное название портала для конечных пользователей
- Отображается в приложениях, на сайте, в уведомлениях
- Примеры: "Моё IPTV", "Premium TV Service", "Спорт ТВ"

Описание (Description)

- Краткое описание сервиса для пользователей
- Используется в app stores, на лендингах
- Может содержать слоган или краткое описание преимуществ
- Пример: "Лучшее IPTV для всей семьи. 200+ каналов в HD качестве"

Бесплатные пакеты (Free Packages)**Концепция:**

- Список пакетов, доступных всем абонентам портала автоматически
- Не требуется создавать подписку для каждого абонента
- Используется для базового контента, демо-каналов, trial периода

Применение:

- **Базовые каналы** — федеральные, общедоступные каналы
- **Пробный доступ** — первый месяц для всех новых абонентов
- **Промо контент** — рекламные и информационные каналы
- **Loyalty программа** — бонусные каналы для всех клиентов

Управление:

```
# Добавить пакет в список бесплатных
POST /portal/free-packages/{packageId}

# Удалить пакет из списка бесплатных
DELETE /portal/free-packages/{packageId}
```

2.1.4 Получение информации о портале

Через веб-интерфейс

1. **Войдите в панель управления Catena**
2. **Выберите портал** (если у вас доступ к нескольким)
3. **Откройте раздел "Настройки портала"**
4. **Просмотрите параметры:**
 5. Основная информация (название, домен)
 6. Брендинг (логотип, описание)
 7. API ключ
 8. Список бесплатных пакетов
 9. Права доступа

Через Management API**Получить информацию о текущем портале:**

```
curl -X GET https://your-catena-domain.com/tv-management/api/v1/portal \
-H "X-Auth-Token: your-api-key"
```

Ответ:

```
{
  "portalId": "pK19SW3AAAE.",
  "name": "my-iptv-service",
  "domain": "myiptv.com",
  "freePackages": ["basicK19SW3AAAE.", "trialK19SW3AAAE."],
  "branding": {
    "logo": "https://myiptv.com/logo.png",
    "title": "My IPTV Service",
    "description": "Premium IPTV streaming for everyone"
  },
  "apiKey": "secret_api_key_1234567890",
  "ownerId": "mK19SW3AAAE.",
  "createdAt": "2024-01-15T10:00:00Z",
  "updatedAt": "2024-10-16T14:30:00Z",
  "flags": {
    "canManage": true,
    "canManageSubscribers": true,
    "canManageContent": true
  }
}
```

Поля ответа:

- **portalId** — уникальный идентификатор портала
- **name** — внутреннее техническое название
- **domain** — доменное имя портала
- **freePackages** — массив ID бесплатных пакетов
- **branding** — параметры брендинга
- **apiKey** — API ключ для аутентификации
- **ownerId** — ID владельца портала
- **createdAt/updatedAt** — даты создания и обновления
- **flags** — права доступа текущего менеджера

2.1.5 Редактирование портала

Через веб-интерфейс

1. Откройте раздел **"Настройки портала"**
2. Нажмите **"Редактировать"**
3. Измените параметры:
4. Название для пользователей (Title)
5. Описание сервиса (Description)
6. URL логотипа (Logo)
7. Сохраните изменения

Важно: Технические параметры (name, domain, portalId) обычно не редактируются после создания.

Через Management API**Обновить параметры портала:**

```
curl -X PUT https://your-catena-domain.com/tv-management/api/v1/portal \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "name": "my-iptv-service",
  "branding": {
    "logo": "https://myiptv.com/new-logo.png",
    "title": "My IPTV - Новое название",
    "description": "Обновлённое описание сервиса"
  }
}'
```

Ответ:

Обновлённый объект Portal с новыми значениями.

Что можно изменить:

- Параметры брендинга (logo, title, description)
- Список бесплатных пакетов (через отдельные endpoints)

Что нельзя изменить:

- portalId — генерируется автоматически
- name — устанавливается при создании
- domain — устанавливается при создании
- ownerId — изменяется отдельно администратором

2.1.6 Управление API ключом

Безопасность API ключа

API ключ – это секретный токен для доступа к portalу. Обращайтесь с ним осторожно:

- Храните в безопасном месте (переменные окружения, secret manager)
- Не коммитьте в Git репозитории
- Не передавайте третьим лицам
- Регулярно обновляйте (каждые 6-12 месяцев)
- Немедленно обновите при подозрении на компрометацию

Регенерация API ключа

Когда нужно регенерировать:

- API ключ случайно попал в публичный репозиторий
- Подозрение на несанкционированный доступ
- Увольнение сотрудника, имевшего доступ к ключу
- Плановое обновление по политике безопасности
- Смена интеграции или биллинговой системы

Через веб-интерфейс:

1. **Откройте "Настройки портала"**
2. **Перейдите в раздел "Безопасность"**
3. **Нажмите "Сгенерировать новый API ключ"**
4. **Подтвердите действие**
5. **Скопируйте новый ключ** (старый перестанет работать немедленно)
6. **Обновите ключ во всех интеграциях**

Через Management API:

```
curl -X POST https://your-catena-domain.com/tv-management/api/v1/portal/reset_api_key \
-H "X-Auth-Token: current-api-key"
```

Ответ:

```
{
  "portalId": "pK19SW3AAAE.",
  "name": "my-iptv-service",
  "apiKey": "new_secret_api_key_0987654321",
  "branding": { ... },
  ...
}
```

Важно:

- Старый API ключ перестаёт работать немедленно
- Все текущие интеграции с старым ключом начнут получать ошибку 401
- Обновите ключ во всех местах: биллинг, monitoring, скрипты
- Сохраните новый ключ в безопасном месте

2.1.7 Управление бесплатными пакетами

Добавление бесплатного пакета

Через веб-интерфейс:

1. Откройте "Настройки портала"
2. Перейдите в раздел "Бесплатные пакеты"
3. Нажмите "Добавить пакет"
4. Выберите пакет из списка доступных
5. Подтвердите добавление

Все абоненты портала немедленно получают доступ к каналам этого пакета.

Через Management API:

```
curl -X POST https://your-catenas-domain.com/tv-management/api/v1/portal/free-packages/basicK19SW3AAAE. \
-H "X-Auth-Token: your-api-key"
```

Ответ: HTTP 201 Created

Удаление бесплатного пакета

Через веб-интерфейс:

1. Откройте "Настройки портала"
2. Перейдите в раздел "Бесплатные пакеты"
3. Найдите пакет в списке
4. Нажмите "Удалить"
5. Подтвердите удаление

Абоненты без явной подписки на этот пакет потеряют доступ к его каналам.

Через Management API:

```
curl -X DELETE https://your-catenas-domain.com/tv-management/api/v1/portal/free-packages/basicK19SW3AAAE. \
-H "X-Auth-Token: your-api-key"
```

Ответ: HTTP 201 Created

Важно:

- Если у абонента есть явная подписка на пакет, он сохранит доступ
- Удаление из бесплатных не удаляет сам пакет
- Изменения вступают в силу немедленно

2.1.8 Типичные сценарии использования

Сценарий 1: Несколько брендов на одной платформе

Задача: Компания управляет тремя IPTV-брендами

Структура:

```
Компания "IPTV Group"
├── Бренд "Premium TV" (premium-tv.com)
│   ├── Целевая аудитория: премиум сегмент
│   ├── Контент: 300 HD/4K каналов
│   ├── Цена: от $20/месяц
│   └── Брендинг: золотой логотип, элегантный дизайн
└── Бренд "Family TV" (family-tv.com)
```

```

├── Целевая аудитория: семьи с детьми
├── Контент: 150 каналов (кино, детские, общие)
├── Цена: от $10/месяц
├── Брендинг: яркие цвета, дружелюбный дизайн
└──
├── Бренд "Sport TV" (sport-tv.com)
├── Целевая аудитория: спортивные фанаты
├── Контент: 50 спортивных каналов
├── Цена: от $15/месяц
└── Брендинг: динамичный, энергичный стиль

```

Преимущества:

- Один streaming сервер для всех брендов
- Централизованное управление контентом
- Разные маркетинговые стратегии
- Изолированные базы абонентов
- Экономия на инфраструктуре

Реализация:

1. Создать 3 портала в Catena
2. Настроить брендинг для каждого
3. Распределить каналы по порталам
4. Создать пакеты с разной ценовой политикой
5. Интегрировать с единой биллинговой системой
6. Развернуть брендированные mobile apps

Сценарий 2: White-label решение для партнёров

Задача: Предоставить IPTV платформу партнёрам под их брендом

Бизнес-модель:

- Вы – провайдер инфраструктуры и контента
- Партнёры – владельцы абонентской базы и брендов
- Каждый партнёр получает свой портал
- Партнёр платит за количество абонентов или фиксированную плату

Пример структуры:

```

Ваша платформа: catena-platform.com
├── Партнёр 1: regional-provider.ru
│   └── 5,000 абонентов
├── Партнёр 2: city-tv.com
│   └── 3,000 абонентов
└── Партнёр 3: corporate-tv.net
    └── 1,000 абонентов (корпоративное ТВ)

```

Что получает партнёр:

- Собственный портал с уникальным доменом
- Полный контроль над брендингом
- Доступ к вашему каталогу каналов
- Брендированное мобильное приложение
- API для интеграции со своим биллингом
- Техподдержку от вашей команды

Что делаете вы:

- Создаёте портал для партнёра
- Предоставляете доступ к каналам
- Поддерживаете инфраструктуру
- Обновляете EPG
- Обеспечиваете стабильную работу
- Выставляете счета партнёру

Workflow создания портала для партнёра:

1. Партнёр регистрируется в вашей системе
2. Вы создаёте портал с его доменом
3. Партнёр настраивает брендинг (логотип, цвета, название)
4. Вы подключаете каналы согласно тарифу
5. Партнёр получает API ключ для интеграции
6. Вы создаёте брендированное mobile app для партнёра
7. Партнёр начинает привлекать абонентов

Сценарий 3: Географическое разделение

Задача: Предоставить IPTV в разных странах/регионах

Почему нужны отдельные порталы:

- Разный контент из-за лицензионных ограничений
- Разные языки интерфейса
- Разные валюты и способы оплаты
- Локальные каналы для каждого региона
- Соответствие местному законодательству

Пример:

```
Международный IPTV сервис
├── Портал "IPTV Russia" (iptv.ru)
│   ├── Контент: российские каналы + международные
│   ├── Язык: русский
│   ├── Валюта: рубли
│   └── 50,000 абонентов
├── Портал "IPTV Europe" (iptv.eu)
│   ├── Контент: европейские каналы
│   ├── Языки: английский, немецкий, французский
│   ├── Валюта: евро
│   └── 30,000 абонентов
└── Портал "IPTV USA" (iptv.com)
    ├── Контент: американские каналы
    ├── Язык: английский
    ├── Валюта: доллары
    └── 20,000 абонентов
```

Сценарий 4: Тестовая среда

Задача: Безопасно тестировать новые функции

Решение:

- Создать отдельный портал `test.myiptv.com`
- Использовать для внутреннего тестирования
- Тестировать новые каналы, пакеты, функции
- Не влиять на production порталы

Преимущества:

- Полная изоляция от продакшн данных
- Возможность экспериментировать
- Тестирование интеграций
- Обучение новых сотрудников

2.1.9 Брендируемые мобильные приложения

Концепция брендируемых приложений

Для каждого портала можно создать отдельное мобильное приложение с уникальным брендом.

Что включает брендируемое приложение:

- Логотип портала в качестве иконки app
- Название портала в App Store / Google Play
- Цветовая схема портала в интерфейсе
- Уникальный Bundle ID / Package Name
- Подключение к API портала через API ключ

Платформы:

- **iOS** — Swift/SwiftUI приложение для iPhone/iPad
- **Android** — Kotlin/Java приложение
- **Android TV** — версия для Smart TV
- **Apple TV** — версия для Apple TV

Процесс создания приложения

Типичный workflow:

1. Вы предоставляете параметры портала:

2. Доменное имя (domain)
3. Логотип (logo)
4. Название (title)
5. Цветовая схема
6. API endpoint

7. Разработчик создаёт приложение:

8. Брендирует интерфейс согласно дизайну
9. Интегрирует с Catena API
10. Настраивает авторизацию через SMS
11. Реализует плеер для просмотра

12. Публикация в store:

13. Регистрация в Apple Developer / Google Play Console
14. Подготовка скриншотов и описания
15. Прохождение модерации
16. Публикация приложения

17. Абоненты скачивают:

18. Находят ваше приложение в store
19. Устанавливают на устройство
20. Входят через SMS
21. Смотрят каналы

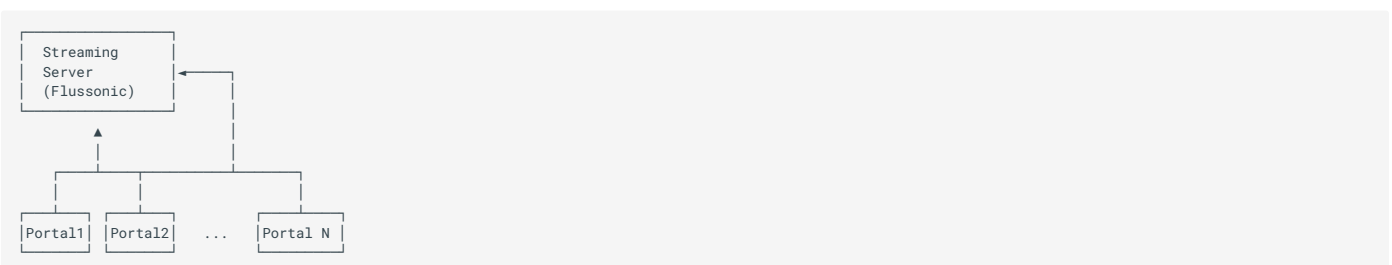
Важно:

- Для iOS требуется Apple Developer аккаунт (\$99/год)
- Для Android требуется Google Play Console (\$25 один раз)
- Приложение должно соответствовать правилам store
- Обновления приложения проходят модерацию

2.1.10 Общая инфраструктура для порталов

Общие streaming-серверы

Все порталы используют одни и те же серверы для доставки контента:



Преимущества:

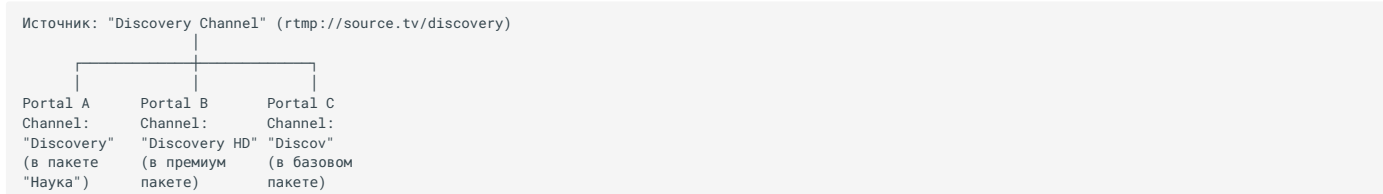
- Один source для канала → N порталов
- Экономия трафика и CPU
- Централизованное управление stream
- Единая точка мониторинга

Разграничение доступа:

- Streaming server проверяет `playback_token` абонента
- Token содержит информацию о `portal_id`
- Абонент может смотреть только каналы своего портала
- Технически возможно предоставить один канал нескольким порталам

Общие каналы для нескольких порталов

Сценарий: Один source канала для разных брендов

Пример:**Как работает:**

1. Канал добавляется в каждый портал отдельно
2. У каждого портала свой `channelId`
3. Но source URL одинаковый
4. Streaming server кеширует поток
5. Все порталы получают поток из кеша

Преимущества:

- Один source → многократное использование
- Экономия на лицензировании (зависит от договора)
- Централизованное обновление EPG
- Единая точка мониторинга качества

2.1.11 Лучшие практики

Именованние порталов**Внутреннее имя (name):**

- Используйте понятные технические названия
- Примеры: `company-premium`, `partner-acme`, `test-portal`
- Избегайте пробелов и специальных символов
- Держите консистентность: `brand-segment` или `client_name`

Публичное название (title):

- Используйте привлекательное маркетинговое название
- Примеры: "Premium TV", "Семейное Телевидение", "СпортТВ+"
- Учитывайте целевую аудиторию
- Проверьте доступность названия (trademark)

Организация контента**Стратегии распределения каналов:**

1. **Полное дублирование** — все порталы имеют одинаковый контент
2. Проще в управлении
3. Подходит для white-label без сегментации
4. **Сегментированный контент** — разный контент для разных порталов
5. Премиум портал: эксклюзивные каналы
6. Базовый портал: стандартный набор
7. Тематический портал: только спорт/кино/новости
8. **Общая база + уникальный контент**
9. Базовые каналы доступны везде
10. Премиум каналы только в дорогих порталах
11. Локальные каналы в региональных порталах

Безопасность**Защита API ключей:**

```
# ПЛОХО - ключ в коде
api_key = "secret_key_123456"

# ХОРОШО - ключ в переменной окружения
api_key = os.getenv('CATENA_API_KEY')

# ЛУЧШЕ - ключ в secret manager
api_key = secrets_manager.get('catena_api_key')
```

Права доступа менеджеров:

- Выдавайте минимально необходимые права
- Content admin не нужен доступ к абонентам
- Support не нужен доступ к API ключу
- Регулярно проверяйте список менеджеров
- Удаляйте доступ уволенных сотрудников

Мониторинг**Что отслеживать для каждого портала:**

- Количество активных абонентов
- Количество одновременных сессий
- Популярные каналы
- Ошибки при входе (failed SMS, invalid tokens)
- API requests count и latency
- Storage usage для каждого портала

Инструменты:

- Grafana dashboards с разбивкой по порталам
- Prometheus метрики с label `portal_id`
- Алерты при аномалиях (резкое падение абонентов)
- Регулярные отчёты для владельцев порталов

2.1.12 Устранение проблем

Абоненты не могут войти**Возможные причины:**

- Указан неверный domain в приложении
- API ключ устарел после регенерации
- Portal временно недоступен
- SMS-шлюз не настроен для портала

Решение:

1. Проверьте domain в настройках портала
2. Убедитесь, что API ключ актуален
3. Проверьте статус сервисов (API, SMS gateway)
4. Просмотрите логи ошибок авторизации
5. Протестируйте вход из другого приложения/браузера

Каналы не воспроизводятся**Возможные причины:**

- Проблемы со streaming-сервером
- Канал не добавлен в портал
- Абонент не подписан на пакет с каналом
- Проблемы с сетью у абонента

Решение:

1. Проверьте работу канала на другом портале
2. Убедитесь, что канал существует в этом портале
3. Проверьте подписки абонента
4. Просмотрите логи streaming-сервера
5. Проверьте playback_token абонента

API возвращает 401 Unauthorized**Возможные причины:**

- Неверный API ключ
- API ключ был регенерирован
- Ключ передаётся в неверном формате
- Ключ от другого портала

Решение:

1. Проверьте актуальность API ключа: `GET /portal`
2. Убедитесь, что ключ в заголовке: `X-Auth-Token: your-key`
3. Проверьте, что используется ключ нужного портала
4. При необходимости регенерируйте ключ

Два портала видят абонентов друг друга

Проблема: Нарушена изоляция данных между порталами

Это не должно происходить по дизайну системы. Если происходит:

1. Немедленно сообщите в техподдержку
2. Проверьте, что используете правильный API ключ
3. Проверьте, что не путаете порталы в коде
4. Просмотрите логи API запросов

Причины (редко):

- Баг в системе (требует исправления)
- Неправильная интеграция (используется один ключ для разных порталов)
- Кеширование на стороне клиента

2.1.13 См. также

- **Управление менеджерами** — создание пользователей для управления порталами
- **Управление абонентами** — абоненты привязаны к конкретному portalу
- **Управление пакетами** — пакеты создаются в рамках портала
- **Управление каналами** — каналы добавляются в порталы
- **Управление подписками** — бесплатные пакеты портала

2.2 Управление менеджерами портала

Менеджеры – это пользователи, которые имеют доступ к панели управления портала в Catena. Система менеджеров позволяет предоставлять различные уровни доступа разным сотрудникам для управления контентом, абонентами и настройками портала.

2.2.1 Что такое менеджер

Менеджер в Catena – это учётная запись пользователя с правами доступа к управлению порталом. В отличие от абонентов (которые смотрят каналы), менеджеры администрируют систему.

Ключевые особенности:

- **Аутентификация по email и паролю** – вход в панель управления
- **Система прав доступа** – гибкая настройка разрешений для каждого менеджера
- **Множественные порталы** – один email может управлять несколькими порталами
- **Роли и полномочия** – от просмотра статистики до полного администрирования
- **Изоляция по порталам** – менеджер видит только данные своих порталов

Типичная структура команды:

```
Портал "My IPTV Service"
├─ Владелец портала (ownerId)
│   └─ Назначается снаружи системы, полный доступ
├─ Главный администратор (isAdmin: true)
│   └─ Управление инфраструктурой
├─ Контент-менеджер (isContentAdmin: true)
│   └─ Управление каналами, пакетами, EPG
├─ Менеджер по абонентам (isSubscriberAdmin: true)
│   └─ Работа с абонентами и подписками
├─ Оператор поддержки (только чтение)
│   └─ Просмотр данных, без изменений
```

2.2.2 Владелец портала vs Менеджеры

Владелец портала (Owner)

Важно: Владелец портала – это особая роль, которая управляется **вне этого Management API**.

Характеристики владельца:

- Устанавливается при создании портала на уровне инфраструктуры
- **Не может быть изменён** через Management API данного портала
- Имеет полный и неограниченный доступ к portalу
- Права доступа к нему **не применяются** (isAdmin, isContentAdmin и т.д.)
- Может создавать, изменять и удалять других менеджеров
- Может назначать любые права другим менеджерам

Поле ownerId в портале:

```
{
  "portalId": "pK19SW3AAAE.",
  "ownerId": "mK19SW3AAAE.", // ID владельца
  "name": "my-portal"
}
```

Смена владельца:

- Выполняется администратором системы Catena
- Не доступна через обычный Management API
- Требуется обращения в техническую поддержку
- Используется при передаче портала другому лицу

Обычные менеджеры**Менеджеры, создаваемые через API:**

- Создаются владельцем портала или другими администраторами
- Имеют ограниченные права согласно настройкам
- Могут быть изменены или удалены владельцем
- Подчиняются системе прав доступа

Ключевое различие:

Характеристика	Владелец (Owner)	Обычный менеджер
Создание	Вне Management API	Через Management API
Изменение прав	Не применимо	Настраивается владельцем
Удаление	Только администратором системы	Владельцем портала
Доступ	Полный всегда	Согласно правам

2.2.3 Система прав доступа

Уровни доступа

isAdmin — администратор инфраструктуры

- Управление техническими настройками портала
- Создание и удаление других менеджеров
- Изменение критичных параметров
- Доступ к серверным настройкам
- **Не даёт автоматически доступ к контенту или абонентам**

isContentAdmin — администратор контента

- Управление каналами (создание, редактирование, удаление)
- Управление пакетами каналов
- Управление источниками EPG
- Настройка связей каналов с пакетами
- **Не имеет доступа к абонентам**

isSubscriberAdmin — администратор абонентов

- Управление абонентами (создание, редактирование, удаление)
- Управление подписками на пакеты
- Просмотр сеансов воспроизведения
- Просмотр журнала операций
- **Не может изменять каналы и пакеты**

Комбинирование прав:

Менеджер может иметь несколько прав одновременно:

```
{
  "isAdmin": true,
  "isContentAdmin": true,
  "isSubscriberAdmin": true
}
```

Это даст полный доступ ко всем функциям портала (кроме смены владельца).

Примеры распределения прав**Сценарий 1: Небольшая компания**

```
Владелец: owner@company.com
└─ Все права по умолчанию

Технический специалист: tech@company.com
└─ isAdmin: true, isContentAdmin: true, isSubscriberAdmin: false

Поддержка: support@company.com
└─ isAdmin: false, isContentAdmin: false, isSubscriberAdmin: true
```

Сценарий 2: Крупный оператор

```
Владелец: ceo@operator.com
└─ Главный владелец бизнеса

CTO: cto@operator.com
└─ isAdmin: true (техническая инфраструктура)

Контент-директор: content@operator.com
└─ isContentAdmin: true (закупка и настройка каналов)

Руководитель поддержки: support-head@operator.com
└─ isSubscriberAdmin: true (управление клиентской базой)

Операторы поддержки: support1@, support2@, ...
└─ isSubscriberAdmin: true (только работа с абонентами)
```

2.2.4 Основные параметры менеджера

Идентификация

managerId — уникальный идентификатор менеджера

- Формат: base64-кодированный Snowflake ID
- Пример: `mK19SW3AAAE.`
- Генерируется при создании

portalId — идентификатор портала

- К какому portalу привязан менеджер
- Один менеджер создаётся для одного портала
- Пример: `pK19SW3AAAE.`

email — email адрес менеджера

- Используется для входа в систему
- **Уникален в рамках портала**
- **Может повторяться в разных порталах** (один человек управляет несколькими)
- Пример: `admin@company.com`

name — отображаемое имя

- ФИО или идентификатор менеджера
- Используется в интерфейсе и логах
- Пример: "Иван Петров", "Admin"

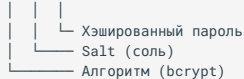
Аутентификация

password — пароль менеджера

- **Write-only поле** — передаётся только при создании/обновлении
- Не возвращается в GET запросах (в целях безопасности)
- Хранится в БД в зашифрованном виде (bcrypt, \$2a\$)
- Формат хэша: \$2a\$12\$salt\$hashedpassword
- Минимальные требования: 8+ символов, сложность (настраивается)

Хранение пароля в БД:

```
passwordEncrypted: "$2a$12$abcdefghijklmnopqrstuvxyz..."
```



Временные метки

createdAt — дата создания менеджера

- ISO 8601 format
- Пример: 2024-01-15T10:00:00Z

updatedAt — дата последнего изменения

- Обновляется при изменении любых параметров
- Пример: 2024-10-16T14:30:00Z

2.2.5 Создание менеджера

Через веб-интерфейс

1. **Откройте раздел "Менеджеры"** в панели управления
2. **Нажмите "Создать менеджера"**
3. **Заполните обязательные поля:**
4. **Email** — адрес электронной почты
5. **Name** — имя или идентификатор
6. **Password** — пароль для входа
7. **Настройте права доступа:**
8. ☒ isAdmin — администратор инфраструктуры
9. ☒ isContentAdmin — управление контентом
10. ☒ isSubscriberAdmin — управление абонентами
11. **Сохраните менеджера**

Важно: Только владелец портала и менеджеры с правом `isAdmin: true` могут создавать других менеджеров.

Через Management API

Создать нового менеджера:

```
curl -X POST https://your-caten-a-domain.com/tv-management/api/v1/managers \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "email": "content@company.com",
  "name": "Контент-менеджер",
  "password": "SecurePassword123!",
  "isAdmin": false,
  "isContentAdmin": true,
  "isSubscriberAdmin": false
}'
```

Ответ:

```
{
  "managerId": "mK19SW3AAAB.",
  "portalId": "pK19SW3AAAE.",
  "email": "content@company.com",
  "name": "Контент-менеджер",
  "isAdmin": false,
  "isContentAdmin": true,
  "isSubscriberAdmin": false,
  "createdAt": "2024-10-16T15:00:00Z",
  "updatedAt": "2024-10-16T15:00:00Z"
}
```

Обратите внимание: Поле `password` не возвращается в ответе.

2.2.6 Просмотр списка менеджеров

Через веб-интерфейс

В разделе "Менеджеры" отображается таблица:

- **Имя** — отображаемое имя менеджера
- **Email** — адрес электронной почты
- **Права** — иконки или badges с активными правами
- **Последний вход** — когда менеджер заходил в систему
- **Создан** — дата создания учётной записи
- **Действия** — кнопки редактирования и удаления

Через Management API

Получить список всех менеджеров портала:

```
curl -X GET https://your-caten-a-domain.com/tv-management/api/v1/managers \
-H "X-Auth-Token: your-api-key"
```

Ответ:

```
{
  "managers": [
    {
      "managerId": "mK19SW3AAAE.",
      "portalId": "pK19SW3AAAE.",
      "email": "owner@company.com",
      "name": "Владелец",
      "isAdmin": true,
      "isContentAdmin": true,
      "isSubscriberAdmin": true,
      "createdAt": "2024-01-15T10:00:00Z",
      "updatedAt": "2024-01-15T10:00:00Z"
    },
    {
      "managerId": "mK19SW3AAAB.",
      "portalId": "pK19SW3AAAE.",
      "email": "content@company.com",
      "name": "Контент-менеджер",
      "isAdmin": false,
      "isContentAdmin": true,

```

```

    "isSubscriberAdmin": false,
    "createdAt": "2024-10-16T15:00:00Z",
    "updatedAt": "2024-10-16T15:00:00Z"
  }
}
}

```

Примечание: Список включает всех менеджеров портала, но **не показывает, кто из них владелец**. Эта информация доступна в настройках портала через поле `ownerId`.

2.2.7 Получение информации о менеджере

Через Management API

```

curl -X GET https://your-catena-domain.com/tv-management/api/v1/managers/mKl9SW3AAAB. \
-H "X-Auth-Token: your-api-key"

```

Ответ: Аналогичен объекту `manager` из списка.

2.2.8 Редактирование менеджера

Через веб-интерфейс

1. **Откройте список менеджеров**
2. **Найдите нужного менеджера** и нажмите "Редактировать"
3. **Измените параметры:**
4. Имя
5. Email (с осторожностью)
6. Права доступа
7. Пароль (если нужно сменить)
8. **Сохраните изменения**

Ограничения:

- Владелец портала не может изменить свои права (они управляются снаружи)
- Менеджер не может изменить собственные права (только владелец или admin)
- Нельзя убрать последнего администратора

Через Management API

Обновить данные менеджера:

```

curl -X PUT https://your-catena-domain.com/tv-management/api/v1/managers/mKl9SW3AAAB. \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "email": "content@company.com",
  "name": "Старший контент-менеджер",
  "isAdmin": false,
  "isContentAdmin": true,
  "isSubscriberAdmin": true
}'

```

Смена пароля:

```

curl -X PUT https://your-catena-domain.com/tv-management/api/v1/managers/mKl9SW3AAAB. \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "email": "content@company.com",
  "name": "Контент-менеджер",
  "password": "NewSecurePassword456!",
  "isAdmin": false,
  "isContentAdmin": true,
  "isSubscriberAdmin": false
}'

```

Важно:

- Если передать поле `password`, оно будет обновлено
- Если не передавать `password`, старый пароль сохранится
- При смене пароля менеджер должен будет войти заново

2.2.9 Удаление менеджера

Через веб-интерфейс

1. **Откройте список менеджеров**
2. **Найдите менеджера** для удаления
3. **Нажмите "Удалить"**
4. **Подтвердите удаление**

Предупреждение:

- Нельзя удалить владельца портала
- Нельзя удалить последнего администратора
- Менеджер немедленно потеряет доступ к системе

Через Management API

```
curl -X DELETE https://your-catena-domain.com/tv-management/api/v1/managers/mK19SW3AAAB. \
-H "X-Auth-Token: your-api-key"
```

Ответ: HTTP 201 - менеджер удалён

2.2.10 Вход в систему

Процесс аутентификации**Менеджеры входят через email и пароль:**

```
curl -X POST https://your-catena-domain.com/tv-management/api/v1/login \
-H "Content-Type: application/json" \
-d '{
  "email": "admin@company.com",
  "password": "password123"
}'
```

Ответ:

```
{
  "portals": [
    {
      "portalId": "pK19SW3AAAE.",
      "portalName": "My IPTV Service",
      "sessionId": "sessionK19SW3AAAE."
    },
    {
      "portalId": "pK19SW3AAAB.",
      "portalName": "Premium TV",
      "sessionId": "sessionK19SW3AAAB."
    }
  ]
}
```

Множественные порталы:

- Если email используется в нескольких порталах, возвращается список всех доступных
- Менеджер выбирает портал для работы
- Каждый портал имеет свой `sessionId` для дальнейшей работы

Использование sessionId:

После получения sessionId, используйте его как API ключ для доступа к portalу:

```
curl -X GET https://your-catena-domain.com/tv-management/api/v1/channels \
-H "X-Auth-Token: sessionKl9SW3AAAE."
```

2.2.11 Типичные сценарии использования

Сценарий 1: Разделение обязанностей

Задача: Распределить работу между специалистами

Команда:

1. **Владелец портала** (owner@company.com)

2. Стратегические решения

3. Назначение менеджеров

4. Контроль финансов

5. **Технический директор** (tech@company.com)

6. isAdmin: true

7. Настройка серверов

8. Мониторинг инфраструктуры

9. Создание других менеджеров

10. **Контент-менеджер** (content@company.com)

11. isContentAdmin: true

12. Добавление новых каналов

13. Создание пакетов

14. Настройка EPG

15. **Менеджер клиентского сервиса** (support@company.com)

16. isSubscriberAdmin: true

17. Работа с абонентами

18. Управление подписками

19. Решение проблем клиентов

Создание команды:

```
#!/bin/bash

# Владелец создаёт остальных менеджеров

# Технический директор
curl -X POST $API_URL/managers -H "X-Auth-Token: $OWNER_KEY" -d '{
  "email": "tech@company.com",
  "name": "Технический директор",
  "password": "SecurePass1!",
  "isAdmin": true,
  "isContentAdmin": false,
  "isSubscriberAdmin": false
}'

# Контент-менеджер
curl -X POST $API_URL/managers -H "X-Auth-Token: $OWNER_KEY" -d '{
  "email": "content@company.com",
  "name": "Контент-менеджер",
  "password": "SecurePass2!",
  "isAdmin": false,
  "isContentAdmin": true,
  "isSubscriberAdmin": false
}'

# Менеджер поддержки
curl -X POST $API_URL/managers -H "X-Auth-Token: $OWNER_KEY" -d '{
  "email": "support@company.com",
```

```
"name": "Клиентский сервис",
"password": "SecurePass3!",
"isAdmin": false,
"isContentAdmin": false,
"isSubscriberAdmin": true
}'
```

Сценарий 2: Управление несколькими порталами

Задача: Один человек управляет тремя IPTV-брендами

Решение:

Создайте менеджера с одним email в каждом портале:

```
import requests

portals = [
    {"id": "portal1_id", "api_key": "portal1_key", "name": "Premium TV"},
    {"id": "portal2_id", "api_key": "portal2_key", "name": "Family TV"},
    {"id": "portal3_id", "api_key": "portal3_key", "name": "Sport TV"}
]

manager_data = {
    "email": "admin@company.com",
    "name": "Главный администратор",
    "password": "SecurePassword123!",
    "isAdmin": True,
    "isContentAdmin": True,
    "isSubscriberAdmin": True
}

for portal in portals:
    response = requests.post(
        f"{API_URL}/managers",
        headers={"X-Auth-Token": portal['api_key']},
        json=manager_data
    )

    print(f"Создан менеджер в портале {portal['name']}")
```

При входе:

```
curl -X POST $API_URL/login -d '{
  "email": "admin@company.com",
  "password": "SecurePassword123!"
}'
```

Результат:

```
{
  "portals": [
    {"portalId": "portal1_id", "portalName": "Premium TV", "sessionId": "session1"},
    {"portalId": "portal2_id", "portalName": "Family TV", "sessionId": "session2"},
    {"portalId": "portal3_id", "portalName": "Sport TV", "sessionId": "session3"}
  ]
}
```

Менеджер может выбрать любой портал для работы.

Сценарий 3: Временный доступ

Задача: Предоставить временный доступ подрядчику

Решение:

1. Создать менеджера с ограниченными правами
2. После завершения работ удалить учётную запись

```
# Создать временного менеджера
curl -X POST $API_URL/managers -H "X-Auth-Token: $OWNER_KEY" -d '{
  "email": "contractor@external.com",
  "name": "Временный контент-менеджер",
  "password": "TempPass123!",
  "isAdmin": false,
  "isContentAdmin": true,
  "isSubscriberAdmin": false
}'
```

```
# После завершения работ (через месяц)
curl -X DELETE $API_URL/managers/mK19SW3AAAB. \
-H "X-Auth-Token: $OWNER_KEY"
```

Рекомендации:

- Устанавливайте напоминание об удалении
- Используйте описательное имя ("Временный...")
- Предоставляйте минимально необходимые права
- Смените пароли после удаления временного доступа

2.2.12 Лучшие практики

Безопасность паролей

Требования к паролям:

```
import re

def validate_password(password):
    """Валидация пароля"""

    if len(password) < 8:
        return False, "Минимум 8 символов"

    if not re.search(r'[A-Z]', password):
        return False, "Требуется хотя бы одна заглавная буква"

    if not re.search(r'[a-z]', password):
        return False, "Требуется хотя бы одна строчная буква"

    if not re.search(r'[0-9]', password):
        return False, "Требуется хотя бы одна цифра"

    if not re.search(r'[@#%&*(),.?":{}|<>]', password):
        return False, "Требуется хотя бы один спецсимвол"

    return True, "Пароль надёжный"

# Примеры
print(validate_password("password")) # False
print(validate_password("Password!")) # True
```

Политика паролей:

- Минимум 8 символов
- Заглавные и строчные буквы
- Цифры и спецсимволы
- Не использовать общие пароли (password123, admin, qwerty)
- Менять каждые 90 дней
- Не использовать один пароль для всех порталов

Управление правами

Принцип минимальных привилегий:

НЕ давайте больше прав, чем нужно для работы

Плохо:

```
{
  "email": "intern@company.com",
  "isAdmin": true,           // Слишком много
  "isContentAdmin": true,    // Слишком много
  "isSubscriberAdmin": true  // Слишком много
}
```

Хорошо:

```
{
  "email": "intern@company.com",
  "isAdmin": false,
  "isContentAdmin": false,
  "isSubscriberAdmin": true    // Только то, что нужно
}
```

Аудит прав:

Регулярно проверяйте, кому какие права назначены:

```
# Получить всех менеджеров с admin правами
curl -X GET $API_URL/managers -H "X-Auth-Token: $API_KEY" \
  | jq '.managers[] | select(.isAdmin == true)'

# Проверить, нет ли избыточных прав
curl -X GET $API_URL/managers -H "X-Auth-Token: $API_KEY" \
  | jq '.managers[] | select(.isAdmin == true and .isContentAdmin == true and .isSubscriberAdmin == true)'
```

Удаление уволенных сотрудников**Чек-лист при увольнении:**

1. ☒ Немедленно удалить учётную запись менеджера
2. ☒ Проверить, не был ли он владельцем портала
3. ☒ Сменить API ключи портала (если имел доступ)
4. ☒ Проверить журнал операций на подозрительные действия
5. ☒ Уведомить команду об изменениях в доступах

```
#!/bin/bash
# offboard-manager.sh

MANAGER_EMAIL="$1"
MANAGER_ID=$(curl -s -X GET $API_URL/managers -H "X-Auth-Token: $API_KEY" \
  | jq -r ".managers[] | select(.email == \"$MANAGER_EMAIL\") | .managerId")

if [ -z "$MANAGER_ID" ]; then
  echo "Менеджер не найден"
  exit 1
fi

# Удалить менеджера
curl -X DELETE $API_URL/managers/$MANAGER_ID \
  -H "X-Auth-Token: $API_KEY"

echo " Менеджер $MANAGER_EMAIL удалён"

# Проверить, не был ли он владельцем
PORTAL=$(curl -s -X GET $API_URL/portal -H "X-Auth-Token: $API_KEY")
OWNER_ID=$(echo $PORTAL | jq -r '.ownerId')

if [ "$OWNER_ID" == "$MANAGER_ID" ]; then
  echo "⚠ ВНИМАНИЕ: Этот менеджер был владельцем портала!"
  echo "Обратитесь к администратору системы для назначения нового владельца"
fi
```

Мониторинг активности менеджеров**Отслеживание действий:**

Используйте журнал операций для аудита:

```
# Найти все операции за последние 7 дней
curl -X GET "https://your-catena-domain.com/tv-management/api/v1/operations?created_at_gte=$(date -d '7 days ago' '+%Y-%m-%d')" \
  -H "X-Auth-Token: your-api-key" \
  | jq '.operations[] | {type, createdAt, payload}'
```

Метрики для отслеживания:

- Количество операций каждым менеджером
- Время последнего входа
- Подозрительные действия (массовые удаления)
- Неудачные попытки входа

2.2.13 Типичные сценарии настройки прав

Только просмотр данных

Задача: Дать доступ аналитику для просмотра статистики

Решение:

```
{
  "email": "analyst@company.com",
  "name": "Аналитик",
  "password": "SecurePass123!",
  "isAdmin": false,
  "isContentAdmin": false,
  "isSubscriberAdmin": false
}
```

Примечание: Без специальных прав менеджер может только просматривать данные, но не изменять.

Контент-команда

Задача: Команда из 3 человек управляет контентом

```
#!/bin/bash

CONTENT_TEAM=(
  "content-lead@company.com:Руководитель контента"
  "content-editor1@company.com:Контент-редактор 1"
  "content-editor2@company.com:Контент-редактор 2"
)

for MEMBER in "${CONTENT_TEAM[@]"; do
  EMAIL=$(echo $MEMBER | cut -d: -f1)
  NAME=$(echo $MEMBER | cut -d: -f2)

  curl -X POST $API_URL/managers -H "X-Auth-Token: $OWNER_KEY" -d "{
    \"email\": \"$EMAIL\",
    \"name\": \"$NAME\",
    \"password\": \"$TempPassword123!\",
    \"isAdmin\": false,
    \"isContentAdmin\": true,
    \"isSubscriberAdmin\": false
  }"

  echo "Создан: $NAME ($EMAIL)"
done
```

Горячая линия поддержки

Задача: 10 операторов работают с абонентами поочередно

Решение:

1. Создать 10 учётных записей с правом `isSubscriberAdmin`
2. Каждый оператор входит под своим email
3. Все действия логируются в журнал операций
4. Можно отследить, кто что делал

```
import requests

def create_support_team(count=10):
    """Создать команду поддержки"""

    for i in range(1, count + 1):
```

```

requests.post(
    f"{API_URL}/managers",
    headers={"X-Auth-Token": OWNER_KEY},
    json={
        "email": f"support{i}@company.com",
        "name": f"Оператор поддержки {i}",
        "password": generate_secure_password(),
        "isAdmin": False,
        "isContentAdmin": False,
        "isSubscriberAdmin": True
    }
)

print(f"Создано {count} операторов поддержки")

create_support_team(10)

```

2.2.14 Устранение проблем

Не могу войти в систему

Возможные причины:

1. Неверный email или пароль
2. Учётная запись удалена
3. Учётная запись заблокирована (если есть такая функция)
4. Email указан для другого портала

Решение:

1. Проверьте правильность email и пароля
2. Убедитесь, что вводите email в нижнем регистре
3. Попросите владельца проверить наличие вашей учётной записи
4. Попробуйте сбросить пароль (если есть функция)

Не вижу нужный раздел в интерфейсе

Проблема: Менеджер не видит раздел "Абоненты" или "Каналы"

Причина: Недостаточно прав

Решение:

1. Проверьте свои права через GET /managers/{id}
2. Попросите владельца или администратора назначить нужные права
3. Для доступа к абонентам нужен `isSubscriberAdmin: true`
4. Для доступа к каналам нужен `isContentAdmin: true`

Не могу создать другого менеджера

Проблема: API возвращает ошибку при попытке создать менеджера

Причины:

- У вас нет права `isAdmin: true`
- Вы не являетесь владельцем портала
- Email уже используется в этом портале

Решение:

1. Проверьте свои права
2. Только владелец и администраторы могут создавать менеджеров
3. Используйте уникальный email для каждого менеджера в портале

Не могу изменить владельца портала

Проблема: Хочу сменить ownerId, но API не позволяет

Это правильное поведение:

- Владелец портала управляется **вне Management API**
- Смена владельца – критичная операция
- Требуется обращения к администратору системы Catena
- Не может быть выполнена самостоятельно

Процедура смены владельца:

1. Обратитесь в техническую поддержку Catena
2. Предоставьте:
3. Portal ID
4. Текущий owner ID
5. Новый owner ID (или создайте нового менеджера заранее)
6. Обоснование смены
7. Администратор выполнит смену на уровне системы
8. Новый владелец получит полный доступ

2.2.15 Безопасность

Двухфакторная аутентификация (2FA)

Рекомендация: Если ваша установка Catena поддерживает 2FA, включите её для всех менеджеров, особенно с правами `isAdmin`.

Аудит доступа**Регулярно проверяйте:**

```
#!/bin/bash
# audit-managers.sh

echo "=== Аудит менеджеров портала ==="
echo ""

# Получить всех менеджеров
MANAGERS=$(curl -s -X GET $API_URL/managers -H "X-Auth-Token: $API_KEY")

# Подсчитать по правам
ADMINS=$(echo $MANAGERS | jq ' [.managers[] | select(.isAdmin == true)] | length')
CONTENT_ADMINS=$(echo $MANAGERS | jq ' [.managers[] | select(.isContentAdmin == true)] | length')
SUBSCRIBER_ADMINS=$(echo $MANAGERS | jq ' [.managers[] | select(.isSubscriberAdmin == true)] | length')
TOTAL=$(echo $MANAGERS | jq ' .managers | length')

echo "Всего менеджеров: $TOTAL"
echo "Администраторов: $ADMINS"
echo "Контент-менеджеров: $CONTENT_ADMINS"
echo "Менеджеров абонентов: $SUBSCRIBER_ADMINS"
echo ""

# Список администраторов
echo "Администраторы (полный доступ):"
echo $MANAGERS | jq -r ' .managers[] | select(.isAdmin == true) | " - \(.name) (\(.email))"'
```

Ротация паролей

Политика:

- Менять пароли каждые 90 дней
- Не использовать предыдущие 5 паролей
- Уведомлять менеджеров за неделю до истечения

Скрипт напоминания:

```
from datetime import datetime, timedelta

def check_password_expiry():
    """Проверить устаревшие пароли"""

    managers = get_managers()
    warnings = []

    for manager in managers:
        # Время последнего обновления = время смены пароля
        updated = datetime.fromisoformat(manager['updatedAt'].replace('Z', '+00:00'))
        days_since_update = (datetime.now(updated.tzinfo) - updated).days

        if days_since_update > 90:
            warnings.append({
                'manager': manager['email'],
                'days': days_since_update,
                'severity': 'expired'
            })
        elif days_since_update > 83: # За неделю
            warnings.append({
                'manager': manager['email'],
                'days': days_since_update,
                'severity': 'expiring_soon'
            })

    # Отправить уведомления
    for warning in warnings:
        if warning['severity'] == 'expired':
            send_email(
                to=warning['manager'],
                subject="⚠ Пароль устарел",
                body=f"Ваш пароль не менялся {warning['days']} дней. Смените его немедленно."
            )
        else:
            send_email(
                to=warning['manager'],
                subject="i Напоминание о смене пароля",
                body=f"Ваш пароль устареет через {90 - warning['days']} дней."
            )

    return warnings
```

2.2.16 См. также

- **Управление порталами** — владелец портала и его роль
- **Журнал операций** — аудит действий менеджеров
- **Управление абонентами** — что могут делать менеджеры с правом isSubscriberAdmin
- **Управление каналами** — что могут делать менеджеры с правом isContentAdmin

3. Управление контентом

3.1 Управление телевизионными каналами

Телевизионные каналы — это основная единица контента в системе Catena. Каждый канал представляет собой отдельный поток видеоконтента, который доставляется абонентам через клиентские приложения.

3.1.1 Что такое канал в Catena

Канал в Catena — это сущность, которая объединяет:

- **Технические параметры** — уникальный идентификатор и имя для стриминг-сервера
- **Визуальное представление** — отображаемое название и логотип для пользователей
- **Программу передач** — связь с источником EPG (Electronic Program Guide)
- **Тарификацию** — включение в пакеты каналов для продажи абонентам

Важно понимать, что **Catena не занимается непосредственной доставкой видеопотока** — это задача стриминг-сервера (например, Flussonic Media Server). Catena управляет метаданными каналов и правами доступа к ним.

3.1.2 Основные параметры канала

Технические параметры

Идентификатор канала (Channel ID)

- Автоматически генерируется при создании канала
- Формат: base64-кодированный Snowflake ID с заменой символов `+/=` на `-_.`
- Пример: `aK19SW3AAAE.`
- Используется для программного доступа через API
- Не редактируется после создания

Имя для стриминга (Name)

- Уникальное техническое имя канала в рамках портала
- Используется стриминг-сервером для идентификации потока
- Требования:
 - Только латинские буквы, цифры, дефис и подчёркивание: `[a-zA-Z0-9_-]`
 - Длина от 2 до 20 символов
 - Должно быть уникальным в рамках вашего портала
- Примеры: `sport1`, `news-hd`, `first_channel`

Параметры отображения

Название для пользователей (Title)

- Локализованное название канала, которое видят зрители
- Может содержать любые символы, включая кириллицу
- Примеры: `Первый канал`, `Спорт HD`, `Новости 24`

Логотип (Logo)

- Изображение канала для отображения в клиентских приложениях
- Формат: PNG, рекомендуется прозрачный фон
- Загружается как binary data (base64)
- Доступен через отдельный эндпоинт: `GET /channels/{channelId}/logo`
- Оптимальный размер: 300x300 пикселей

Интеграция с EPG

Имя источника EPG (EPG Source Name)

- Название источника электронной программы передач
- Ссылается на ранее созданный EPG Source в системе
- Один источник EPG может использоваться для множества каналов

Имя канала в EPG (EPG Channel Name)

- Идентификатор канала внутри источника EPG
- Используется для сопоставления вашего канала с программой передач из EPG-файла
- Должен точно соответствовать названию канала в XML EPG

Пример связи: Если в вашем EPG-файле канал называется `perviy-kanal`, то в поле EPG Channel Name нужно указать именно `perviy-kanal`, даже если в Catena у вас канал называется `first_channel`.

3.1.3 Создание канала

Через веб-интерфейс

1. **Откройте раздел "Каналы"** в панели управления Catena
2. **Нажмите кнопку "Создать канал"**
3. **Заполните обязательные поля:**
4. **Name** — техническое имя для стриминг-сервера (латиницей)
5. **Title** — название для отображения пользователям
6. **Заполните дополнительные поля (опционально):**
7. **Логотип** — загрузите изображение канала (PNG)
8. **EPG Source Name** — выберите источник программы передач
9. **EPG Channel Name** — укажите имя канала в EPG-источнике
10. **Сохраните канал**

После создания канал получит уникальный ID и будет доступен для добавления в пакеты.

Через Management API

```
curl -X POST https://your-catena-domain.com/tv-management/api/v1/channels \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "name": "sport1",
  "title": "Спорт HD",
  "epgSourceName": "main-epg",
  "epgChannelName": "sport-channel-1"
}'
```

Ответ:

```
{
  "channelId": "aK19SW3AAAE.",
  "portalId": "pK19SW3AAAE.",
  "name": "sport1",
  "title": "Спорт HD",
  "epgSourceName": "main-epg",
  "epgChannelName": "sport-channel-1",
  "packages": []
}
```

3.1.4 Просмотр списка каналов

Через веб-интерфейс

В разделе "Каналы" отображается таблица со всеми каналами портала:

- **Логотип** — миниатюра логотипа канала
- **Название** — отображаемое название (Title)
- **Техническое имя** — имя для стриминга (Name)
- **Пакеты** — список пакетов, в которые входит канал
- **EPG** — информация о подключённом источнике программы передач
- **Действия** — кнопки редактирования и удаления

Через Management API

Получить список всех каналов:

```
curl -X GET https://your-catena-domain.com/tv-management/api/v1/channels \
-H "X-Auth-Token: your-api-key"
```

Ответ:

```
{
  "channels": [
    {
      "channelId": "aK19SW3AAAE.",
      "portalId": "pK19SW3AAAE.",
      "name": "sport1",
      "title": "Спорт HD",
      "packages": ["basic", "premium"],
      "epgSourceName": "main-epg",
      "epgChannelName": "sport-channel-1"
    }
  ],
  "next": "cursor-for-next-page"
}
```

Пагинация: Для получения следующей страницы используйте параметр `cursor`:

```
curl -X GET "https://your-catena-domain.com/tv-management/api/v1/channels?cursor=cursor-for-next-page" \
-H "X-Auth-Token: your-api-key"
```

3.1.5 Редактирование канала

Через веб-интерфейс

1. **Откройте список каналов**
2. **Найдите нужный канал** и нажмите кнопку "Редактировать"
3. **Измените параметры:**
4. Title — можно изменить отображаемое название
5. Логотип — загрузить новое изображение
6. EPG Source Name / EPG Channel Name — изменить связь с программой передач

7. Сохраните изменения

Важно: Поле `name` (техническое имя) нельзя изменить после создания канала. Если нужно изменить техническое имя, создайте новый канал и удалите старый.

Через Management API

```
curl -X PUT https://your-caten-a-domain.com/tv-management/api/v1/channels/aKl9SW3AAAE. \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "name": "sport1",
  "title": "Спорт Full HD",
  "epgSourceName": "new-epg-source",
  "epgChannelName": "sport-hd-channel"
}'
```

3.1.6 Загрузка и получение логотипа

Загрузка логотипа через API

При создании или обновлении канала логотип передаётся в поле `logo` как base64-строка:

```
curl -X PUT https://your-caten-a-domain.com/tv-management/api/v1/channels/aKl9SW3AAAE. \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "name": "sport1",
  "title": "Спорт HD",
  "logo": "data:image/png;base64,iVBORw0KGgoAAAANSUUEUgAA..."
}'
```

Получение логотипа

Логотип доступен через отдельный эндпоинт:

```
curl -X GET https://your-caten-a-domain.com/tv-management/api/v1/channels/aKl9SW3AAAE./logo \
-H "X-Auth-Token: your-api-key" \
--output channel-logo.png
```

Этот URL можно использовать в клиентских приложениях для отображения логотипа.

3.1.7 Удаление канала

Через веб-интерфейс

1. **Откройте список каналов**
2. **Найдите канал** для удаления
3. **Нажмите кнопку "Удалить"**
4. **Подтвердите удаление**

Предупреждение: При удалении канала он будет автоматически удалён из всех пакетов, в которые был включён.

Через Management API

```
curl -X DELETE https://your-caten-a-domain.com/tv-management/api/v1/channels/aKl9SW3AAAE. \
-H "X-Auth-Token: your-api-key"
```

3.1.8 Связь каналов с пакетами

Канал сам по себе не доступен абонентам. Чтобы предоставить доступ к каналу, его нужно включить в [пакет каналов](#).

Поле `packages` в канале (read-only): При получении информации о канале поле `packages` содержит список имён пакетов, в которые включён канал. Это поле только для чтения и обновляется автоматически при добавлении/удалении канала в пакеты через API [управления связями каналов и пакетов](#).

Пример:

```
{
  "channelId": "aK19SW3AAAE.",
  "name": "sport1",
  "title": "Спорт HD",
  "packages": ["basic", "premium", "sport-package"]
}
```

3.1.9 Связь каналов с EPG

Как работает интеграция с EPG

1. **Создайте источник EPG** в разделе [EPG Sources](#)
2. **Укажите URL XML**-файла с программой передач
3. **Запустите синхронизацию** EPG-данных
4. **В настройках канала укажите:**
5. `epgSourceName` — имя созданного источника EPG
6. `epgChannelName` — имя канала, как оно указано в XML EPG

Сопоставление каналов

Пример структуры EPG XML:

```
<tv>
  <channel id="perviy-kanal">
    <display-name>Первый канал</display-name>
  </channel>
  <programme start="20241015120000" stop="20241015130000" channel="perviy-kanal">
    <title lang="ru">Новости</title>
  </programme>
</tv>
```

В Catena укажите:

- EPG Source Name: `main-epg` (имя источника, который вы создали)
- EPG Channel Name: `perviy-kanal` (значение атрибута `id` или `display-name` из XML)

После этого программа передач будет автоматически доступна для этого канала в клиентских приложениях.

3.1.10 Типичные сценарии использования

Запуск нового канала

Задача: Добавить новый спортивный канал в сервис

Шаги:

1. Создайте канал в Catena с именем `sport-premium`
2. Загрузите логотип канала
3. Настройте связь с EPG (если доступна программа передач)
4. Добавьте канал в один или несколько пакетов
5. Настройте соответствующий поток на стриминг-сервере с именем `sport-premium`

Массовое добавление каналов

Задача: Добавить 50 каналов из нового контент-провайдера

Решение через API:

1. Подготовьте CSV или JSON с данными каналов
2. Создайте скрипт для автоматического создания каналов через API
3. Загрузите логотипы для каждого канала
4. Настройте EPG сопоставление
5. Сгруппируйте каналы в тематические пакеты

Обновление EPG для каналов

Задача: Сменить источник EPG для группы каналов

Шаги:

1. Создайте новый EPG Source с актуальными данными
2. Обновите каналы, указав новый `epgSourceName`
3. Проверьте корректность сопоставления `epgChannelName`
4. Запустите обновление EPG

Ребрендинг канала

Задача: Изменить название и логотип канала

Шаги:

1. Откройте редактирование канала
2. Обновите поле `title` с новым названием
3. Загрузите новый логотип
4. Сохраните изменения
5. Изменения автоматически появятся в клиентских приложениях при следующем обновлении данных

3.1.11 Лучшие практики

Именованние каналов

- **Name (техническое имя):**
 - Используйте короткие, понятные имена: `sport1`, `news`, `movies-hd`
 - Избегайте спецсимволов, кроме дефиса и подчёркивания
 - Добавляйте суффиксы для HD/SD версий: `sport-hd`, `sport-sd`
- **Title (отображаемое название):**
 - Используйте полные, понятные названия: "Спорт HD", "Новости 24"
 - Можно использовать любые символы и эмодзи
 - Укажите качество в названии: "4K", "HD", "SD" (если важно)

Организация каналов

- **Группируйте каналы логически** по темам, используя пакеты
- **Используйте единый стиль** для логотипов (размер, фон, формат)
- **Поддерживайте актуальность** EPG-данных
- **Документируйте** техническое сопоставление каналов

Управление логотипами

- **Размер:** оптимально 300x300 пикселей
- **Формат:** PNG с прозрачным фоном
- **Вес файла:** не более 100 КБ для быстрой загрузки
- **Единый стиль:** используйте одинаковый стиль для всех логотипов

Интеграция со стриминг-сервером

Помните, что **техническое имя канала в Catena должно совпадать с именем потока на стриминг-сервере:**

- Catena: `name: "sport1"`
- Flussonic: поток должен называться `sport1`

Это обеспечит корректную работу токенов доступа и аналитики просмотров.

3.1.12 Устранение проблем

Канал не отображается в приложении

Возможные причины:

- Канал не включён ни в один пакет
- У абонента нет подписки на пакет с этим каналом
- Не настроено соответствующее вещание на стриминг-сервере

Решение:

1. Проверьте, что канал добавлен в пакет
2. Убедитесь, что абонент подписан на этот пакет
3. Проверьте, что стриминг-сервер отдаёт поток с соответствующим именем

EPG не отображается для канала

Возможные причины:

- Неверно указано `epgChannelName`
- EPG Source не обновлялся или содержит ошибки
- В EPG XML нет данных для этого канала

Решение:

1. Откройте XML EPG и найдите правильный идентификатор канала
2. Обновите `epgChannelName` в настройках канала
3. Запустите принудительное обновление EPG
4. Проверьте логи обновления EPG на наличие ошибок

Ошибка "Name must be unique"

Причина: Канал с таким техническим именем уже существует в вашем портале

Решение:

- Используйте другое техническое имя
- Или удалите существующий канал с таким именем (если он больше не нужен)

Не загружается логотип**Возможные причины:**

- Файл слишком большой
- Неподдерживаемый формат изображения
- Ошибка кодирования в base64

Решение:

- Используйте PNG формат
- Сожмите изображение до размера < 100 КБ
- Проверьте корректность base64-кодирования

3.1.13 См. также

- **Управление пакетами каналов** — группировка каналов для продажи
- **Управление EPG** — настройка электронной программы передач
- **Управление абонентами** — предоставление доступа к каналам
- **API Reference** — полная документация по API каналов

3.2 Управление пакетами каналов

Пакеты каналов — это группы телевизионных каналов, объединённые для продажи абонентам. Пакеты позволяют создавать различные тарифные планы и монетизировать ваш IPTV-сервис.

3.2.1 Что такое пакет каналов

Пакет каналов в Catena — это именованная группа телевизионных каналов, которую можно назначить абонентам. Пакеты позволяют:

- **Создавать тарифные планы** — группировать каналы по темам (спорт, кино, новости) или уровням доступа (базовый, премиум)
- **Монетизировать сервис** — продавать доступ к пакетам каналов абонентам
- **Управлять доступом** — предоставлять разным абонентам доступ к разным наборам каналов
- **Упрощать администрирование** — назначать пакеты вместо управления доступом к каждому каналу отдельно

Важная концепция: Канал сам по себе не доступен абонентам. Доступ к каналу предоставляется только через пакеты, на которые подписан абонент.

3.2.2 Основные параметры пакета

Технические параметры

Идентификатор пакета (Package ID)

- Автоматически генерируется при создании пакета
- Формат: base64-кодированный Snowflake ID с заменой символов `+/=` на `-_.`
- Пример: `aK19SW3AAAE.`
- Используется для программного доступа через API
- Не редактируется после создания

Имя пакета (Name)

- Уникальное техническое имя пакета в рамках портала
- Используется в системе для идентификации пакета
- Требования:
 - Только латинские буквы, цифры, дефис и подчёркивание: `[a-zA-Z0-9_-]`
 - Длина от 2 до 20 символов
 - Должно быть уникальным в рамках вашего портала
- Примеры: `basic`, `premium`, `sport-pack`, `movies_hd`

ID портала (Portal ID)

- Идентификатор портала, к которому принадлежит пакет
- Автоматически устанавливается при создании

Параметры отображения

Описание (Description)

- Текстовое описание пакета для администраторов
- Может содержать информацию о содержании пакета, ценообразовании, целевой аудитории
- Не обязательное поле
- Примеры: "Базовый пакет из 30 каналов", "Премиум спортивные каналы в HD качестве"

Состав пакета

Список каналов (Channels)

- Поле только для чтения (read-only)
- Содержит имена (name) всех каналов, включённых в пакет
- Обновляется автоматически при добавлении/удалении каналов через API управления связями
- Пример: ["sport1", "sport2", "news", "movies-hd"]

3.2.3 Создание пакета

Через веб-интерфейс

1. **Откройте раздел "Пакеты"** в панели управления Catena
2. **Нажмите кнопку "Создать пакет"**
3. **Заполните обязательные поля:**
4. **Name** — техническое имя пакета (латиницей)
5. **Description** — описание пакета (опционально)
6. **Сохраните пакет**
7. **Добавьте каналы в пакет** через раздел управления составом

После создания пакет получит уникальный ID и будет доступен для назначения абонентам.

Через Management API

```
curl -X POST https://your-catena-domain.com/tv-management/api/v1/packages \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "name": "premium",
  "description": "Премиум пакет с HD каналами"
}'
```

Ответ:

```
{
  "packageId": "pK19SW3AAAE.",
  "portalId": "portal123",
  "name": "premium",
  "description": "Премиум пакет с HD каналами",
  "channels": []
}
```

3.2.4 Просмотр списка пакетов

Через веб-интерфейс

В разделе "Пакеты" отображается таблица со всеми пакетами портала:

- **Название** — имя пакета (Name)
- **Описание** — текстовое описание пакета
- **Количество каналов** — сколько каналов входит в пакет
- **Абонентов** — количество абонентов с этим пакетом
- **Действия** — кнопки редактирования и удаления

Через Management API

Получить список всех пакетов:

```
curl -X GET https://your-catena-domain.com/tv-management/api/v1/packages \
-H "X-Auth-Token: your-api-key"
```

Ответ:

```
{
  "packages": [
    {
      "packageId": "pK19SW3AAAE.",
      "portalId": "portal123",
      "name": "premium",
      "description": "Премиум пакет с HD каналами",
      "channels": ["sport1", "sport2", "news-hd", "movies-4k"]
    },
    {
      "packageId": "bK19SW3AAAE.",
      "portalId": "portal123",
      "name": "basic",
      "description": "Базовый пакет каналов",
      "channels": ["news", "general"]
    }
  ],
  "next": "cursor-for-next-page"
}
```

Пагинация:

Для получения следующей страницы используйте параметр `cursor`:

```
curl -X GET "https://your-catena-domain.com/tv-management/api/v1/packages?cursor=cursor-for-next-page" \
-H "X-Auth-Token: your-api-key"
```

3.2.5 Получение информации о пакете

Через Management API

Получить пакет по ID:

```
curl -X GET https://your-catena-domain.com/tv-management/api/v1/packages/pK19SW3AAAE. \
-H "X-Auth-Token: your-api-key"
```

Ответ:

```
{
  "packageId": "pK19SW3AAAE.",
  "portalId": "portal123",
  "name": "premium",
  "description": "Премиум пакет с HD каналами",
  "channels": ["sport1", "sport2", "news-hd", "movies-4k"]
}
```

3.2.6 Редактирование пакета

Через веб-интерфейс

1. **Откройте список пакетов**
2. **Найдите нужный пакет** и нажмите кнопку "Редактировать"
3. **Измените параметры:**
4. Name — техническое имя (лучше не менять после создания)
5. Description — описание пакета
6. **Сохраните изменения**

Примечание: Изменение состава каналов в пакете выполняется отдельно через управление связями каналов и пакетов.

Через Management API

```
curl -X PUT https://your-catena-domain.com/tv-management/api/v1/packages/pK19SW3AAAE. \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "name": "premium",
  "description": "Премиум пакет с HD и 4K каналами"
}'
```

3.2.7 Удаление пакета

Через веб-интерфейс

1. **Откройте список пакетов**
2. **Найдите пакет** для удаления
3. **Нажмите кнопку "Удалить"**
4. **Подтвердите удаление**

Предупреждение: При удалении пакета:

- Все абоненты потеряют доступ к каналам из этого пакета (если у них нет других пакетов с этими каналами)
- Связи между пакетом и каналами будут удалены
- Связи между пакетом и абонентами будут удалены

Через Management API

```
curl -X DELETE https://your-catena-domain.com/tv-management/api/v1/packages/pK19SW3AAAE. \
-H "X-Auth-Token: your-api-key"
```

3.2.8 Управление составом пакета

Добавление канала в пакет

Через Management API:

```
curl -X POST https://your-catena-domain.com/tv-management/api/v1/channels-packages \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "channelId": "cK19SW3AAAE.",
  "packageId": "pK19SW3AAAE.",
  "portalId": "portal123"
}'
```

Ответ:

```
{
  "channelId": "cK19SW3AAAE.",
```

```
"packageId": "pK19SW3AAAE.",
"portalId": "portal123"
}
```

После добавления:

- Канал будет отображаться в поле `channels` пакета
- Пакет будет отображаться в поле `packages` канала
- Все абоненты с этим пакетом получают доступ к добавленному каналу

Удаление канала из пакета

Через Management API:

```
curl -X DELETE https://your-catenal-domain.com/tv-management/api/v1/channels-packages \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "channelId": "cK19SW3AAAE.",
  "packageId": "pK19SW3AAAE.",
  "portalId": "portal123"
}'
```

Важно: Удаление канала из пакета не удаляет сам канал из системы, только разрывает связь между каналом и пакетом.

Массовое управление каналами

Для добавления нескольких каналов в пакет используйте последовательные вызовы API:

```
# Добавить канал 1
curl -X POST https://your-catenal-domain.com/tv-management/api/v1/channels-packages \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{"channelId": "channel1", "packageId": "premium", "portalId": "portal123"}'

# Добавить канал 2
curl -X POST https://your-catenal-domain.com/tv-management/api/v1/channels-packages \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{"channelId": "channel12", "packageId": "premium", "portalId": "portal123"}'
```

3.2.9 Назначение пакетов абонентам

Подробнее о назначении пакетов абонентам см. в разделе [Управление подписками](#).

Добавление пакета абоненту

```
curl -X POST https://your-catenal-domain.com/tv-management/api/v1/packages-subscribers \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "packageId": "pK19SW3AAAE.",
  "subscriberId": "sK19SW3AAAE.",
  "portalId": "portal123"
}'
```

После назначения пакета абоненту:

- Абонент получит доступ ко всем каналам из этого пакета
- Пакет отобразится в списке пакетов абонента
- Доступ будет предоставлен во всех клиентских приложениях

Удаление пакета у абонента

```
curl -X DELETE https://your-catenal-domain.com/tv-management/api/v1/packages-subscribers \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "packageId": "pK19SW3AAAE.",
  "subscriberId": "sK19SW3AAAE.",
  "portalId": "portal123"
}'
```

```
"subscriberId": "sK19SW3AAAE.",
"portalId": "portal123"
}'
```

3.2.10 Бесплатные пакеты

Бесплатные пакеты — это пакеты, которые автоматически доступны всем абонентам портала без необходимости явного назначения.

Принцип работы

Бесплатные пакеты настраиваются на уровне портала:

- Администратор добавляет пакет в список бесплатных пакетов портала
- Все абоненты автоматически получают доступ к каналам из бесплатных пакетов
- Не требуется явное назначение пакета каждому абоненту

Типичное использование:

- **Пробный период** — предоставить базовый набор каналов всем новым абонентам
- **Бесплатные каналы** — открытые каналы, доступные всем (новости, общественные каналы)
- **Демонстрационный контент** — показать возможности сервиса до покупки подписки

Добавление пакета в список бесплатных

```
curl -X POST https://your-catenal-domain.com/tv-management/api/v1/portal/free-packages/pK19SW3AAAE. \
-H "X-Auth-Token: your-api-key"
```

После добавления в список бесплатных:

- Все существующие и новые абоненты получают доступ к каналам этого пакета
- Пакет будет отображаться как бесплатный в настройках портала

Удаление пакета из списка бесплатных

```
curl -X DELETE https://your-catenal-domain.com/tv-management/api/v1/portal/free-packages/pK19SW3AAAE. \
-H "X-Auth-Token: your-api-key"
```

Важно: Удаление пакета из бесплатных не удаляет сам пакет, только убирает автоматический доступ для всех абонентов.

3.2.11 Типичные сценарии использования

Создание базовой тарифной сетки

Задача: Создать три уровня подписки: Базовый, Стандарт, Премиум

Шаги:**1. Создайте три пакета:**

2. `basic` — 30 основных каналов
3. `standard` — 50 каналов (базовые + развлекательные)
4. `premium` — 80 каналов (все + спорт и кино в HD)

5. Наполните пакеты каналами:

6. Базовый: новости, общие, музыка
7. Стандарт: базовые + сериалы, документальные
8. Премиум: стандарт + спорт HD, кино 4K, эксклюзив

9. Настройте бесплатный пакет:

10. Создайте пакет `trial` с 5 открытыми каналами
11. Добавьте его в список бесплатных пакетов портала
12. Назначайте пакеты абонентам в зависимости от их подписки

Тематические пакеты

Задача: Создать тематические дополнительные пакеты

Примеры пакетов:

- `sport-package` — все спортивные каналы
- `kids-package` — детские каналы
- `movies-package` — киноканалы
- `news-package` — новостные каналы

Преимущества:

- Абонент может докупить интересующие темы к базовой подписке
- Более гибкая монетизация
- Персонализация предложения

Региональные пакеты

Задача: Предоставлять разные наборы каналов для разных регионов

Решение:

1. Создайте региональные пакеты:
2. `region-moscow` — московские региональные каналы
3. `region-spb` — петербургские каналы
4. `region-south` — каналы юга России
5. Назначайте соответствующий региональный пакет при регистрации абонента
6. Абонент получит базовый пакет + региональный

Временные акции

Задача: Провести промо-акцию с расширенным доступом

Шаги:

1. Создайте временный пакет `promo-may`
2. Добавьте в него премиум-каналы
3. Назначьте пакет всем активным абонентам
4. По окончании акции удалите пакет у всех абонентов

3.2.12 Лучшие практики

Планирование структуры пакетов**Рекомендации по именованию:**

- Используйте понятные технические имена: `basic`, `premium`, `sport-hd`
- Избегайте использования версий в имени: не `premium-v2`, а создайте новый пакет
- Используйте префиксы для группировки: `addon-sport`, `addon-kids`, `addon-movies`

Структура тарифной сетки:

- **Базовый уровень** — минимальный набор для начала использования сервиса
- **Средний уровень** — оптимальное соотношение цена/количество каналов
- **Премиум уровень** — максимальный набор со всеми доступными каналами
- **Дополнительные пакеты** — тематические надстройки к основным пакетам

Управление изменениями**При изменении состава пакета:**

- Информируйте абонентов о добавлении новых каналов
- Заранее предупреждайте об удалении каналов из пакета
- Ведите документацию по составу каждого пакета
- Сохраняйте историю изменений для аналитики

При изменении ценообразования:

- Не меняйте техническое имя пакета при изменении цены
- Используйте систему биллинга для управления ценами
- Обеспечьте плавный переход для существующих абонентов

Мониторинг и аналитика**Отслеживайте метрики:**

- Количество абонентов на каждом пакете
- Популярность каналов внутри пакетов
- Конверсию из бесплатного пакета в платные
- Отток абонентов при изменении состава пакета

Используйте данные для оптимизации:

- Формируйте пакеты на основе популярности каналов
- Тестируйте разные комбинации каналов
- Корректируйте состав пакетов по результатам аналитики

Интеграция с биллингом

Рекомендации:

- Используйте техническое имя пакета (name) как идентификатор в биллинговой системе
- Синхронизируйте назначение/удаление пакетов с платежами через API
- Автоматизируйте блокировку доступа при неоплате
- Настройте webhook для уведомлений об изменении подписок

3.2.13 Устранение проблем

Абонент не видит каналы из пакета

Возможные причины:

- Пакет не назначен абоненту
- В пакете нет каналов (пустой пакет)
- Проблемы с синхронизацией данных в клиентском приложении

Решение:

1. Проверьте список пакетов абонента через API
2. Убедитесь, что пакет содержит каналы
3. Проверьте, что каналы правильно настроены на стриминг-сервере
4. Перезапустите клиентское приложение для обновления данных

Каналы дублируются в списке абонента

Причина: Канал входит в несколько пакетов, назначенных абоненту

Это нормальное поведение:

- Абонент может иметь несколько пакетов
- Канал может входить в несколько пакетов одновременно
- Клиентское приложение должно дедуплицировать список каналов

Решение: Убедитесь, что клиентское приложение корректно обрабатывает дубликаты.

Ошибка при добавлении канала в пакет

Возможные причины:

- Неверный `channelId` или `packageId`
- Канал уже есть в этом пакете
- Канал или пакет принадлежат другому portalу

Решение:

1. Проверьте существование канала и пакета
2. Убедитесь, что `portalId` совпадает для канала, пакета и запроса
3. Проверьте, не добавлен ли уже этот канал в пакет

Пакет не удаляется**Возможные причины:**

- Пакет назначен абонентам
- Пакет находится в списке бесплатных пакетов портала
- Недостаточно прав для удаления

Решение:

1. Сначала удалите пакет у всех абонентов
2. Удалите пакет из списка бесплатных (если он там есть)
3. Затем удалите сам пакет

Бесплатный пакет не работает**Возможные причины:**

- Пакет не добавлен в список бесплатных пакетов портала
- Абонент явно заблокирован или не активирован
- Пакет пустой (не содержит каналов)

Решение:

1. Проверьте список бесплатных пакетов в настройках портала
2. Убедитесь, что абонент активен
3. Проверьте наличие каналов в пакете

3.2.14 См. также

-
- **Управление каналами** — создание и настройка телевизионных каналов
 - **Управление абонентами** — регистрация и управление подписчиками
 - **Управление подписками** — назначение пакетов абонентам
 - **Настройки портала** — управление бесплатными пакетами
 - **API Reference** — полная документация по API пакетов

3.3 Управление источниками EPG

Источники EPG (Electronic Program Guide — электронная программа передач) предоставляют информацию о телепрограмме для каждого канала: названия передач, время начала и окончания, описания, жанры и возрастные рейтинги.

3.3.1 Что такое источник EPG

Источник EPG в Catena — это ссылка на внешний XML-файл с расписанием телепередач. Система периодически загружает этот файл и синхронизирует данные о программах для отображения в клиентских приложениях.

Основные возможности:

- **Автоматическая синхронизация** — регулярная загрузка и обновление программы передач
- **Поддержка нескольких источников** — разные источники EPG для разных групп каналов
- **Мониторинг загрузок** — отслеживание статуса и результатов обновления EPG
- **Просмотр программ** — получение расписания через API для интеграции

Типичный workflow:

1. Создать источник EPG с указанием URL XML-файла
2. Настроить период автоматического обновления
3. Привязать каналы к источнику EPG (в настройках каналов)
4. Система автоматически загружает и обновляет программу передач
5. Абоненты видят актуальную телепрограмму в приложениях

3.3.2 Основные параметры источника EPG

Технические параметры

Идентификатор источника (EPG Source ID)

- Автоматически генерируется при создании источника
- Формат: base64-кодированный Snowflake ID с заменой символов `+/=` на `-_.`
- Пример: `aK19SW3AAAE.`
- Используется для программного доступа через API
- Не редактируется после создания

Имя источника (Name)

- Уникальное техническое имя источника EPG
- Используется для идентификации в системе и в настройках каналов
- Примеры: `main-epg`, `sports-epg`, `xmltv-provider`

ID портала (Portal ID)

- Идентификатор портала, к которому принадлежит источник
- Автоматически устанавливается при создании

Параметры загрузки

URL источника (URL)

- Полный URL до XML-файла с программой передач
- Поддерживаемые протоколы: HTTP, HTTPS
- Пример: `https://epg.example.com/epg.xml`

- **Обязательное поле**

Период обновления (Period)

- Интервал в днях для автоматического обновления EPG
- Пример: `7` — обновлять каждые 7 дней
- Минимальное значение: 1 день
- Если не указан, используется значение по умолчанию
- **Система автоматически загружает EPG** согласно указанному периоду без необходимости ручного вмешательства

Результат последней загрузки

Информация о загрузке (Last Fetch Result)

Поле `last_fetch_result` содержит информацию о последнем обновлении EPG:

- `fetched_at` — время последней загрузки (ISO 8601)
- `job_id` — идентификатор задания загрузки
- `status` — статус загрузки:
 - `success` — успешно загружено
 - `error` — ошибка загрузки
 - `timeout` — превышено время ожидания
- `message` — текстовое сообщение о результате
- `channels` — количество обновлённых каналов
- `foundPrograms` — количество найденных программ в XML
- `importedPrograms` — количество импортированных программ
- `deletedPrograms` — количество удалённых старых программ
- `fetchDuration` — время загрузки XML в секундах
- `importDuration` — время импорта данных в секундах

3.3.3 Создание источника EPG

Через веб-интерфейс

1. Откройте раздел "EPG Sources" в панели управления Catena
2. Нажмите кнопку "Создать источник EPG"
3. Заполните обязательные поля:
4. **Name** — техническое имя источника (например, `main-epg`)
5. **URL** — полный URL до XML-файла с EPG
6. Заполните дополнительные поля (опционально):
7. **Period** — период обновления в днях (например, `7`)
8. Сохраните источник
9. Запустите первичную загрузку через кнопку "Обновить сейчас"

После создания источник получит уникальный ID и будет доступен для привязки к каналам.

Через Management API

```
curl -X POST https://your-catena-domain.com/tv-management/api/v1/epg-sources \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "name": "main-epg",
  "url": "https://epg.example.com/xmltv.xml",
  "period": 7
}'
```

Ответ:

```
{
  "epgSourceId": "eK19SW3AAAE.",
  "portalId": "pK19SW3AAAE.",
  "name": "main-epg",
  "url": "https://epg.example.com/xmltv.xml",
  "period": 7,
  "last_fetch_result": null
}
```

3.3.4 Просмотр списка источников EPG

Через веб-интерфейс

В разделе "EPG Sources" отображается таблица со всеми источниками:

- **Название** — имя источника (Name)
- **URL** — адрес XML-файла
- **Период** — интервал обновления в днях
- **Последнее обновление** — дата и время последней загрузки
- **Статус** — результат последней загрузки (success/error/timeout)
- **Программ** — количество импортированных программ
- **Действия** — кнопки обновления, редактирования и удаления

Через Management API

Получить список всех источников EPG:

```
curl -X GET https://your-catena-domain.com/tv-management/api/v1/epg-sources \
-H "X-Auth-Token: your-api-key"
```

Ответ:

```
{
  "epgSources": [
    {
      "epgSourceId": "eK19SW3AAAE.",
      "portalId": "pK19SW3AAAE.",
      "name": "main-epg",
      "url": "https://epg.example.com/xmltv.xml",
      "period": 7,
      "last_fetch_result": {
        "epgSourceId": "eK19SW3AAAE.",
        "fetched_at": "2024-10-16T10:30:00Z",
        "job_id": "job123",
        "status": "success",
        "message": "EPG source fetched successfully",
        "channels": 25,
        "foundPrograms": 5000,
        "importedPrograms": 4950,
        "deletedPrograms": 2100,
        "fetchDuration": 5,
        "importDuration": 12
      }
    }
  ]
}
```

3.3.5 Получение информации об источнике

Через Management API

```
curl -X GET https://your-catena-domain.com/tv-management/api/v1/epg-sources/eK19SW3AAAE. \
-H "X-Auth-Token: your-api-key"
```

Ответ: Аналогичен объекту EPG source из списка.

3.3.6 Редактирование источника EPG

Через веб-интерфейс

1. **Откройте список источников EPG**
2. **Найдите нужный источник** и нажмите кнопку "Редактировать"
3. **Измените параметры:**
4. Name — техническое имя
5. URL — адрес XML-файла
6. Period — период обновления
7. **Сохраните изменения**

Примечание: Изменение URL или period не запускает автоматическое обновление — используйте кнопку "Обновить сейчас" для немедленной загрузки.

Через Management API

```
curl -X PUT https://your-catena-domain.com/tv-management/api/v1/epg-sources/eK19SW3AAAE. \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "name": "main-epg",
  "url": "https://epg.example.com/updated-xmltv.xml",
  "period": 3
}'
```

3.3.7 Принудительное обновление EPG

Вы можете запустить внеплановое обновление EPG в любой момент.

Через веб-интерфейс

1. **Откройте список источников EPG**
2. **Найдите нужный источник**
3. **Нажмите кнопку "Обновить сейчас"**
4. **Дождитесь завершения** — процесс может занять от нескольких секунд до минут

Через Management API

```
curl -X POST https://your-catena-domain.com/tv-management/api/v1/epg-sources/eK19SW3AAAE./update \
-H "X-Auth-Token: your-api-key"
```

Ответ:

```
{
  "jobId": "job456"
}
```

Важно: Обновление выполняется асинхронно. Используйте `jobId` для отслеживания прогресса или проверьте поле `last_fetch_result` через некоторое время.

3.3.8 Удаление источника EPG

Через веб-интерфейс

1. **Откройте список источников EPG**
2. **Найдите источник** для удаления
3. **Нажмите кнопку "Удалить"**
4. **Подтвердите удаление**

Предупреждение: При удалении источника EPG:

- Все программы из этого источника будут удалены
- Каналы, привязанные к этому источнику, потеряют связь с EPG
- Абоненты перестанут видеть программу передач для этих каналов

Через Management API

```
curl -X DELETE https://your-caten-a-domain.com/tv-management/api/v1/epg-sources/eK19SW3AAAE. \
-H "X-Auth-Token: your-api-key"
```

3.3.9 Просмотр программ из источника EPG

Вы можете получить список программ для анализа или отладки.

Получение программ через API

Получить все программы источника:

```
curl -X GET https://your-caten-a-domain.com/tv-management/api/v1/epg-sources/eK19SW3AAAE./programs \
-H "X-Auth-Token: your-api-key"
```

Фильтрация по дате:

```
curl -X GET "https://your-caten-a-domain.com/tv-management/api/v1/epg-sources/eK19SW3AAAE./programs?date=2024-10-16" \
-H "X-Auth-Token: your-api-key"
```

Фильтрация по каналу:

```
curl -X GET "https://your-caten-a-domain.com/tv-management/api/v1/epg-sources/eK19SW3AAAE./programs?epgChannelName=channel1" \
-H "X-Auth-Token: your-api-key"
```

Комбинированная фильтрация:

```
curl -X GET "https://your-caten-a-domain.com/tv-management/api/v1/epg-sources/eK19SW3AAAE./programs?date=2024-10-16&epgChannelName=channel1" \
-H "X-Auth-Token: your-api-key"
```

Ответ:

```
{
  "programs": [
    {
      "programId": "prK19SW3AAAE.",
      "portalId": "pK19SW3AAAE.",
      "epgSourceId": "eK19SW3AAAE.",
      "epgChannelName": "channel1",
      "date": "2024-10-16",
      "start": "2024-10-16T12:00:00Z",
      "end": "2024-10-16T13:00:00Z",
      "title": "Новости",
      "language": "ru",
      "description": "Главные новости дня",
      "genre": "News",
      "rating": "0+"
    }
  ],
  "next": "cursor-for-next-page"
}
```

Пагинация:

```
curl -X GET "https://your-caten-domain.com/tv-management/api/v1/epg-sources/eK19SW3AAAE./programs?cursor=cursor-for-next-page" \
-H "X-Auth-Token: your-api-key"
```

3.3.10 Просмотр истории обновлений EPG

Catena автоматически сохраняет историю всех обновлений EPG, что позволяет отслеживать успешность загрузок, анализировать проблемы и мониторить работу источников.

Автоматическое обновление EPG**Как работает автоматическое обновление:**

- Система автоматически запускает обновление EPG согласно параметру `period` каждого источника
- Обновления выполняются в фоновом режиме без остановки работы системы
- Каждое обновление записывается в историю с полной информацией о результате
- При успешном обновлении новые программы становятся доступны абонентам немедленно

Преимущества автоматического обновления:

- Не требуется ручное вмешательство для поддержания актуальности EPG
- Программа передач всегда остаётся актуальной для абонентов
- Возможность отследить все попытки обновления и выявить проблемы

Просмотр истории через веб-интерфейс

На странице источника EPG отображается:

- **История последних обновлений** — таблица со всеми попытками загрузки
- **Дата и время** каждого обновления
- **Статус** — success/error/timeout
- **Статистика** — количество найденных, импортированных и удалённых программ
- **Длительность** — время загрузки и импорта данных
- **Сообщения об ошибках** (если были)

Получение истории через API**Получить историю всех обновлений EPG:**

```
curl -X GET https://your-caten-domain.com/tv-management/api/v1/epg-fetches \
-H "X-Auth-Token: your-api-key"
```

Фильтрация по конкретному источнику:

```
curl -X GET "https://your-caten-domain.com/tv-management/api/v1/epg-fetches?epgSourceId=eK19SW3AAAE." \
-H "X-Auth-Token: your-api-key"
```

Пагинация:

```
curl -X GET "https://your-caten-domain.com/tv-management/api/v1/epg-fetches?cursor=cursor-for-next-page" \
-H "X-Auth-Token: your-api-key"
```

Ответ:

```
{
  "epgFetches": [
    {
      "epgFetchId": "fK19SW3AAAE.",
      "epgSourceId": "eK19SW3AAAE.",
    }
  ]
}
```

```

    "created_at": "2024-10-16T10:25:00Z",
    "fetched_at": "2024-10-16T10:30:00Z",
    "job_id": "job123",
    "status": "success",
    "message": "EPG source fetched successfully",
    "channels": 25,
    "foundPrograms": 5000,
    "importedPrograms": 4950,
    "deletedPrograms": 2100,
    "fetchDuration": 5,
    "importDuration": 12
  },
  {
    "epgFetchId": "fK19SW3AAAA.",
    "epgSourceId": "eK19SW3AAAE.",
    "created_at": "2024-10-09T10:25:00Z",
    "fetched_at": "2024-10-09T10:29:00Z",
    "job_id": "job122",
    "status": "success",
    "message": "EPG source fetched successfully",
    "channels": 25,
    "foundPrograms": 4800,
    "importedPrograms": 4750,
    "deletedPrograms": 1950,
    "fetchDuration": 4,
    "importDuration": 10
  }
],
"next": "cursor-for-next-page"
}

```

Поля истории обновлений

Основные поля записи истории:

- **epgFetchId** — уникальный идентификатор попытки обновления
- **epgSourceId** — идентификатор источника EPG
- **created_at** — время создания задачи на обновление (ISO 8601)
- **fetched_at** — время фактического завершения загрузки (ISO 8601)
- **job_id** — идентификатор фонового задания
- **status** — статус обновления:
 - **success** — успешно загружено и импортировано
 - **error** — произошла ошибка
 - **timeout** — превышено время ожидания
- **message** — текстовое описание результата
- **channels** — количество обновлённых каналов
- **foundPrograms** — всего программ найдено в XML файле
- **importedPrograms** — программ успешно импортировано в базу
- **deletedPrograms** — устаревших программ удалено
- **fetchDuration** — время загрузки XML файла (секунды)
- **importDuration** — время обработки и импорта данных (секунды)

Анализ истории обновлений

Использование истории для мониторинга:

1. **Отслеживание стабильности** — проверка, что обновления проходят регулярно
2. **Выявление проблем** — поиск записей со статусом error или timeout
3. **Анализ производительности** — отслеживание времени загрузки и импорта
4. **Контроль качества данных** — сравнение количества программ между обновлениями

Примеры использования:

```
# Проверить последние обновления с ошибками
curl -X GET "https://your-catena-domain.com/tv-management/api/v1/epg-fetches" \
-H "X-Auth-Token: your-api-key" | jq '.epgFetches[] | select(.status != "success")'

# Получить среднее время импорта для источника
curl -X GET "https://your-catena-domain.com/tv-management/api/v1/epg-fetches?epgSourceId=eK19SW3AAAE." \
-H "X-Auth-Token: your-api-key" | jq '[_epgFetches[]].importDuration | add / length'
```

3.3.11 Привязка каналов к источнику EPG

Источник EPG сам по себе не предоставляет программу передач каналам. Необходимо явно указать в настройках каждого канала:

- **EPG Source Name** — имя источника EPG
- **EPG Channel Name** — имя канала в XML EPG

Подробнее см. раздел [Интеграция каналов с EPG](#).

Пример настройки канала:

```
curl -X PUT https://your-catena-domain.com/tv-management/api/v1/channels/cK19SW3AAAE. \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "name": "sport1",
  "title": "Спорт HD",
  "epgSourceName": "main-epg",
  "epgChannelName": "sport-channel-1"
}'
```

После этого программа передач из источника `main-epg` для канала `sport-channel-1` будет доступна для канала `sport1`.

3.3.12 Формат EPG XML

Структура XMLTV

Catena поддерживает стандартный формат XMLTV. Базовая структура:

```
<?xml version="1.0" encoding="UTF-8"?>
<tv>
  <!-- Описание каналов -->
  <channel id="channel1">
    <display-name>Первый канал</display-name>
  </channel>

  <!-- Программы -->
  <programme start="20241016120000" stop="20241016130000" channel="channel1">
    <title lang="ru">Новости</title>
    <desc lang="ru">Главные новости дня</desc>
    <category lang="ru">News</category>
    <rating system="age">
      <value>0+</value>
    </rating>
  </programme>
</tv>
```

Поддерживаемые поля**Обязательные поля программы:**

- `start` — время начала (формат: YYYYMMDDHHmmss)
- `stop` — время окончания
- `channel` — идентификатор канала (используется для сопоставления)
- `title` — название передачи

Опциональные поля:

- `desc` — описание передачи
- `category` — жанр (News, Sports, Movie и т.д.)
- `rating` — возрастной рейтинг (0+, 6+, 12+, 16+, 18+)
- `lang` — язык программы

3.3.13 Типичные сценарии использования

Подключение стандартного XMLTV провайдера

Задача: Подключить EPG от стороннего провайдера

Шаги:

1. Получите URL XMLTV от провайдера (например, `https://provider.com/epg.xml`)
2. Создайте источник EPG в Catena с этим URL
3. Установите период обновления 1 день
4. Запустите первичную загрузку
5. Проверьте результат загрузки в `last_fetch_result`
6. Привяжите каналы к источнику, указав корректные `epgChannelName`
7. Проверьте отображение программы в клиентских приложениях

Использование нескольких источников EPG

Задача: Разные группы каналов берут EPG из разных источников

Пример:

- `epg-russia` — российские каналы
- `epg-europe` — европейские каналы
- `epg-sports` — спортивные каналы от специализированного провайдера

Преимущества:

- Независимое обновление разных групп каналов
- Возможность использовать специализированных провайдеров для определённых тематик
- Изоляция проблем — ошибка в одном источнике не влияет на другие

Мониторинг обновлений EPG

Задача: Отслеживать успешность загрузки EPG

Решение через API:

```
# Получить историю обновлений всех источников
curl -X GET https://your-catena-domain.com/tv-management/api/v1/epg-fetches \
-H "X-Auth-Token: your-api-key"

# Или получить историю конкретного источника
curl -X GET "https://your-catena-domain.com/tv-management/api/v1/epg-fetches?epgSourceId=eK19SW3AAAE." \
-H "X-Auth-Token: your-api-key"

# Проверить status для каждой записи
# Если status != "success" — есть проблема
# Если последнее обновление давно — EPG устарел
```

Настройка алертов:

- Создайте скрипт проверки истории обновлений EPG
- Запускайте по расписанию (например, каждый час)
- Отправляйте уведомления при ошибках или отсутствии обновлений
- Используйте endpoint `/epg-fetches` для получения полной истории

3.3.14 Лучшие практики

Выбор провайдера EPG**Критерии выбора:**

- **Полнота данных** — наличие описаний, жанров, рейтингов
- **Актуальность** — как часто обновляется EPG
- **Охват** — поддержка нужных вам каналов
- **Надёжность** — uptime сервиса, скорость ответа
- **Формат** — соответствие стандарту XMLTV
- **Стоимость** — бесплатные vs платные источники

Настройка периода обновления**Рекомендации:**

- **1 день** — для источников с ежедневным обновлением
- **7 дней** — для источников с программой на неделю вперёд
- **Меньше дня** — не рекомендуется, создаёт лишнюю нагрузку

Учитывайте:

- Как часто провайдер обновляет EPG
- Размер XML файла (большие файлы — реже обновлять)
- Нагрузку на ваш сервер

Обработка ошибок**Мониторинг:**

- Регулярно проверяйте `last_fetch_result.status`
- Настройте алерты при `status = "error"` или `"timeout"`
- Отслеживайте `fetched_at` — предупреждайте, если EPG не обновлялся >2 дней

При ошибках:

1. Проверьте доступность URL (откройте в браузере)
2. Проверьте формат XML — валидность структуры
3. Проверьте размер файла — возможно, слишком большой
4. Проверьте сетевые настройки сервера (firewall, проxy)

Оптимизация производительности

Рекомендации:

- Не загружайте EPG чаще, чем необходимо
- Используйте CDN для распространения EPG XML, если это ваш файл
- Убедитесь, что XML файл сжат (gzip)
- Разделяйте большие EPG на несколько источников по тематикам

3.3.15 Устранение проблем

EPG не загружается

Возможные причины:

- URL недоступен или возвращает ошибку
- XML файл имеет неверный формат
- Проблемы с сетью на стороне Catena сервера
- Timeout — файл слишком большой

Решение:

1. Проверьте `last_fetch_result.message` для деталей ошибки
2. Откройте URL в браузере — проверьте доступность
3. Валидируйте XML через онлайн-валидатор
4. Проверьте размер файла — попробуйте уменьшить
5. Проверьте логи сервера Catena

Программы не отображаются в приложении

Возможные причины:

- Канал не привязан к источнику EPG
- Неверно указан `epgChannelName` в настройках канала
- EPG не обновлялся или загрузка была с ошибкой
- Программы в EPG устарели (прошедшие даты)

Решение:

1. Проверьте настройки канала — `epgSourceName` и `epgChannelName`
2. Сравните `epgChannelName` с идентификатором канала в XML
3. Проверьте `last_fetch_result` — была ли успешная загрузка
4. Проверьте наличие программ через API `/epg-sources/{id}/programs`
5. Убедитесь, что в EPG есть программы на текущую/будущую дату

Неполные данные программ

Причина: EPG XML не содержит все поля (описания, жанры, рейтинги)

Решение:

- Свяжитесь с провайдером EPG для улучшения данных
- Используйте другого провайдера с более полными данными
- Примите это как есть — базовая информация (название, время) всё равно будет доступна

Несовпадение временных зон

Проблема: Время программ отображается неверно

Решение:

1. Убедитесь, что в EPG XML время указано в UTC или с указанием timezone
2. Проверьте настройки timezone на сервере Catena
3. Клиентские приложения должны конвертировать UTC в локальное время пользователя

Дубликаты программ

Причина: EPG обновляется, но старые программы не удаляются

Решение:

- Это нормальное поведение при обновлении
- Поле `deletedPrograms` в `last_fetch_result` показывает, сколько удалено
- Если дубликаты остаются, это может быть проблема с идентификацией программ в EPG XML

3.3.16 См. также

- [Управление каналами](#) — привязка каналов к источникам EPG
- [Интеграция EPG с каналами](#) — настройка связи
- [API Reference](#) — полная документация по API источников EPG

4. Управление абонентами

4.1 Управление абонентами

Абоненты (subscribers) — это конечные пользователи вашего IPTV-сервиса, которые получают доступ к просмотру телевизионных каналов. Система Catena обеспечивает гибкое управление абонентами, их подписками на пакеты каналов и контроль доступа.

4.1.1 Что такое абонент

Абонент в Catena — это учётная запись пользователя, который имеет доступ к просмотру каналов через подключённые пакеты.

Основные возможности:

- **Аутентификация по SMS** — основной способ входа абонентов через код, отправленный на телефон
- **Управление подписками** — подключение и отключение пакетов каналов
- **Контроль доступа** — автоматическое управление доступом к каналам на основе подписок
- **Токены воспроизведения** — уникальные токены для авторизации при просмотре
- **Мониторинг активности** — отслеживание сеансов просмотра и активности абонентов

Типичный workflow:

1. Создать учётную запись абонента с указанием телефона
2. Подключить пакеты каналов к абоненту
3. Абонент получает SMS с кодом для входа в приложение
4. После входа абонент получает доступ ко всем каналам из своих пакетов
5. Система автоматически управляет правами доступа на основе активных подписок

4.1.2 Основные параметры абонента

Технические параметры

Идентификатор абонента (Subscriber ID)

- Автоматически генерируется при создании абонента
- Формат: base64-кодированный Snowflake ID с заменой символов `+/=` на `-_.`
- Пример: `akI9SW3AAAE.`
- Используется для программного доступа через API
- Не редактируется после создания

ID портала (Portal ID)

- Идентификатор портала, к которому принадлежит абонент
- Автоматически устанавливается при создании
- Абонент может получать доступ только к каналам и пакетам своего портала

Личные данные

Имя абонента (Name)

- Отображаемое имя или идентификатор пользователя
- Может быть ФИО, никнеймом или идентификатором из внешней системы
- Используется для отображения в интерфейсе управления
- Примеры: "Иван Петров", "user123", "Apartment 42"

Номер телефона (Phone)

- Номер телефона абонента **без кода страны**
- Используется для **аутентификации через SMS** — основного способа входа
- Только цифры, без пробелов и специальных символов
- Паттерн валидации: `^[0-9]*$`
- Примеры: 9161234567, 234567890

Код страны (Phone Country Code)

- Код страны телефона **без знака плюс**
- Используется вместе с полем `phone` для формирования полного номера
- Только цифры
- Паттерн валидации: `^[0-9]*$`
- Примеры: 7 (Россия), 1 (США), 44 (Великобритания)

Полный номер телефона формируется как: `+{phoneCountryCode}{phone}`

Пример: `phoneCountryCode: "7" + phone: "9161234567" = +79161234567`

Параметры доступа

Токен воспроизведения (Playback Token)

- Уникальный токен для авторизации при воспроизведении видео
- Генерируется автоматически системой
- Используется `streaming`-сервером для проверки прав доступа
- Передаётся в приложение после успешной аутентификации
- Может быть регенерирован при необходимости

Список пакетов (Packages)

- Массив идентификаторов пакетов каналов, к которым подключён абонент
- Поле только для чтения — отображает текущие активные подписки
- Обновляется автоматически при подключении/отключении пакетов
- Определяет, к каким каналам абонент имеет доступ
- Пример: `["pK19SW3AAAE.", "bK19SW3AAAE."]`

4.1.3 Аутентификация абонентов

Вход по SMS (основной способ)

Catena использует аутентификацию через SMS как основной способ входа абонентов. Это обеспечивает:

- **Простоту использования** — не нужно запоминать пароли
- **Безопасность** — одноразовые коды, привязанные к телефону
- **Удобство** — быстрая регистрация и вход
- **Защита от мошенничества** — телефон как фактор аутентификации

Процесс входа по SMS:

1. **Абонент вводит номер телефона** в приложении
2. **Система отправляет SMS** с одноразовым кодом на указанный номер
3. **Абонент вводит код** из SMS в приложении
4. **Система проверяет код** и выдаёт токен доступа
5. **Абонент получает доступ** к просмотру каналов из своих пакетов

Важно: Номер телефона является уникальным идентификатором абонента в системе. Убедитесь, что номера указаны корректно при создании учётных записей.

Автоматическое создание абонентов

Система может автоматически создавать учётные записи абонентов при первой попытке входа через SMS, если такая функция включена в настройках портала. Это позволяет реализовать самостоятельную регистрацию пользователей.

4.1.4 Создание абонента

Через веб-интерфейс

1. **Откройте раздел "Абоненты"** в панели управления Catena
2. **Нажмите кнопку "Создать абонента"**
3. **Заполните обязательные поля:**
4. **Name** — имя или идентификатор абонента
5. **Phone Country Code** — код страны (например, 7 для России)
6. **Phone** — номер телефона без кода страны
7. **Сохраните абонента**
8. **Подключите пакеты** через раздел управления подписками

После создания абонент получит уникальный ID и сможет войти в систему через SMS.

Через Management API

```
curl -X POST https://your-catena-domain.com/tv-management/api/v1/subscribers \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "name": "Иван Петров",
  "phoneCountryCode": "7",
  "phone": "9161234567"
}'
```

Ответ:

```
{
  "subscriberId": "sK19SW3AAAE.",
  "portalId": "pK19SW3AAAE.",
  "name": "Иван Петров",
}
```

```

"phoneCountryCode": "7",
"phone": "9161234567",
"playback_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
"packages": []
}

```

4.1.5 Просмотр списка абонентов

Через веб-интерфейс

В разделе "Абоненты" отображается таблица со всеми абонентами портала:

- **Имя** — имя или идентификатор абонента
- **Телефон** — полный номер телефона
- **Пакеты** — количество подключённых пакетов
- **Последняя активность** — время последнего входа или просмотра
- **Статус** — активен/заблокирован
- **Действия** — кнопки редактирования, управления пакетами и удаления

Через Management API

Получить список всех абонентов:

```

curl -X GET https://your-catena-domain.com/tv-management/api/v1/subscribers \
-H "X-Auth-Token: your-api-key"

```

Ответ:

```

{
  "subscribers": [
    {
      "subscriberId": "sK19SW3AAAE.",
      "portalId": "pK19SW3AAAE.",
      "name": "Иван Петров",
      "phoneCountryCode": "7",
      "phone": "9161234567",
      "playback_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
      "packages": ["pK19SW3AAAE.", "bK19SW3AAAE."]
    },
    {
      "subscriberId": "tK19SW3AAAE.",
      "portalId": "pK19SW3AAAE.",
      "name": "Мария Сидорова",
      "phoneCountryCode": "7",
      "phone": "9267654321",
      "playback_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
      "packages": ["pK19SW3AAAE."]
    }
  ],
  "next": "cursor-for-next-page"
}

```

Пагинация:

```

curl -X GET "https://your-catena-domain.com/tv-management/api/v1/subscribers?cursor=cursor-for-next-page" \
-H "X-Auth-Token: your-api-key"

```

4.1.6 Получение информации об абоненте

Через Management API

```

curl -X GET https://your-catena-domain.com/tv-management/api/v1/subscribers/sK19SW3AAAE. \
-H "X-Auth-Token: your-api-key"

```

Ответ: Аналогичен объекту subscriber из списка.

4.1.7 Редактирование абонента

Через веб-интерфейс

1. **Откройте список абонентов**
2. **Найдите нужного абонента** и нажмите кнопку "Редактировать"
3. **Измените параметры:**
4. Name — имя абонента
5. Phone Country Code — код страны
6. Phone — номер телефона
7. **Сохраните изменения**

Примечание: При изменении номера телефона абонент должен будет пройти повторную аутентификацию по SMS с новым номером.

Через Management API

```
curl -X PUT https://your-catena-domain.com/tv-management/api/v1/subscribers/sK19SW3AAAE. \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "name": "Иван Петрович Петров",
  "phoneCountryCode": "7",
  "phone": "9161234567"
}'
```

4.1.8 Управление подписками на пакеты

Подключение пакета к абоненту

Через веб-интерфейс:

1. **Откройте карточку абонента**
2. **Перейдите в раздел "Пакеты"**
3. **Нажмите "Добавить пакет"**
4. **Выберите пакет** из списка доступных
5. **Подтвердите добавление**

Абонент немедленно получит доступ ко всем каналам из добавленного пакета.

Через Management API:

```
curl -X POST https://your-catena-domain.com/tv-management/api/v1/packages-subscribers \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "subscriberId": "sK19SW3AAAE.",
  "packageId": "pK19SW3AAAE."
}'
```

Ответ:

```
{
  "subscriberId": "sK19SW3AAAE.",
  "packageId": "pK19SW3AAAE.",
  "portalId": "pK19SW3AAAE."
}
```

Отключение пакета от абонента

Через веб-интерфейс:

1. **Откройте карточку абонента**
2. **Перейдите в раздел "Пакеты"**
3. **Найдите пакет** в списке активных
4. **Нажмите "Удалить"**
5. **Подтвердите отключение**

Абонент немедленно потеряет доступ ко всем каналам из отключённого пакета.

Через Management API:

```
curl -X DELETE https://your-catenal-domain.com/tv-management/api/v1/packages-subscribers \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "subscriberId": "sK19SW3AAAE.",
  "packageId": "pK19SW3AAAE."
}'
```

Массовое управление подписками

Для массового подключения или отключения пакетов используйте циклы или скрипты. Пример добавления пакета нескольким абонентам:

```
#!/bin/bash
SUBSCRIBERS=("sK19SW3AAAE." "tK19SW3AAAE." "uK19SW3AAAE.")
PACKAGE_ID="pK19SW3AAAE."

for SUBSCRIBER_ID in "${SUBSCRIBERS[@]}; do
  curl -X POST https://your-catenal-domain.com/tv-management/api/v1/packages-subscribers \
  -H "X-Auth-Token: your-api-key" \
  -H "Content-Type: application/json" \
  -d '{
    "subscriberId": "${SUBSCRIBER_ID}",
    "packageId": "${PACKAGE_ID}"
  }'
done
```

4.1.9 Удаление абонента

Через веб-интерфейс

1. **Откройте список абонентов**
2. **Найдите абонента** для удаления
3. **Нажмите кнопку "Удалить"**
4. **Подтвердите удаление**

Предупреждение: При удалении абонента:

- Учётная запись будет полностью удалена
- Все подписки на пакеты будут отменены
- История просмотров сохранится для аналитики
- Абонент потеряет доступ к просмотру каналов
- Восстановление учётной записи будет невозможно

Через Management API

```
curl -X DELETE https://your-catenal-domain.com/tv-management/api/v1/subscribers/sK19SW3AAAE. \
-H "X-Auth-Token: your-api-key"
```

4.1.10 Мониторинг активности абонентов

Просмотр сеансов воспроизведения

Catena автоматически регистрирует все сеансы просмотра каналов абонентами. Это позволяет отслеживать активность, популярность каналов и выявлять проблемы.

Получить сеансы конкретного абонента:

```
curl -X GET "https://your-catena-domain.com/tv-management/api/v1/play-sessions?subscriberId=sK19SW3AAAE." \
-H "X-Auth-Token: your-api-key"
```

Получить только активные сеансы:

```
curl -X GET "https://your-catena-domain.com/tv-management/api/v1/play-sessions?subscriberId=sK19SW3AAAE.&active=true" \
-H "X-Auth-Token: your-api-key"
```

Фильтрация по времени:

```
curl -X GET "https://your-catena-domain.com/tv-management/api/v1/play-sessions?subscriberId=sK19SW3AAAE.&opened_at_gte=1714233600&opened_at_lt=1714320000" \
-H "X-Auth-Token: your-api-key"
```

Ответ:

```
{
  "sessions": [
    {
      "sessionId": "sessK19SW3AAAE.",
      "subscriberId": "sK19SW3AAAE.",
      "channelId": "chK19SW3AAAE.",
      "channelName": "sport1",
      "portalId": "pK19SW3AAAE.",
      "openedAt": 1714233600,
      "closedAt": 1714237200,
      "active": false,
      "bytes": 524288000,
      "ip": "192.168.1.100",
      "userAgent": "VLC/3.0.16"
    }
  ],
  "next": "cursor-for-next-page"
}
```

Журнал операций

Все изменения в учётных записях абонентов (создание, удаление, изменение подписок) записываются в журнал операций.

Получить операции по абоненту:

```
curl -X GET "https://your-catena-domain.com/tv-management/api/v1/operations?subscriberId=sK19SW3AAAE." \
-H "X-Auth-Token: your-api-key"
```

Фильтрация по типу операции:

```
curl -X GET "https://your-catena-domain.com/tv-management/api/v1/operations?subscriberId=sK19SW3AAAE.&type=createPackageSubscriber" \
-H "X-Auth-Token: your-api-key"
```

Ответ:

```
{
  "operations": [
    {
      "operationId": "opK19SW3AAAE.",
      "type": "createSubscriber",
      "subscriberId": "sK19SW3AAAE.",
      "portalId": "pK19SW3AAAE.",
      "createdAt": "2024-10-16T10:00:00Z",
      "payload": {
        "name": "Иван Петров",
        "phone": "+79161234567"
      }
    },
    {
      "operationId": "opK19SW3AAAB.",
      "type": "createPackageSubscriber",
      "subscriberId": "sK19SW3AAAE.",

```

```

    "packageId": "pK19SW3AAAE.",
    "portalId": "pK19SW3AAAE.",
    "createdAt": "2024-10-16T10:05:00Z",
    "payload": {
      "packageId": "pK19SW3AAAE."
    }
  },
  "next": "cursor-for-next-page"
}

```

4.1.11 Типичные сценарии использования

Создание нового абонента с базовым пакетом

Задача: Зарегистрировать нового абонента и подключить базовый пакет

Шаги:

1. Создайте учётную запись абонента через API
2. Получите `subscriberId` из ответа
3. Подключите базовый пакет через API `packages-subscribers`
4. Абонент получает SMS для входа в приложение
5. После входа абонент видит каналы из базового пакета

Пример скрипта:

```

#!/bin/bash

# 1. Создаём абонента
RESPONSE=$(curl -s -X POST https://your-catenal-domain.com/tv-management/api/v1/subscribers \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "name": "Новый абонент",
  "phoneCountryCode": "7",
  "phone": "9161234567"
}')

# 2. Извлекаем ID абонента
SUBSCRIBER_ID=$(echo $RESPONSE | jq -r '.subscriberId')

# 3. Подключаем базовый пакет
curl -X POST https://your-catenal-domain.com/tv-management/api/v1/packages-subscribers \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "subscriberId": \"$SUBSCRIBER_ID\",
  "packageId": \"basic-package-id\"
}'

echo "Абонент создан с ID: $SUBSCRIBER_ID"

```

Апгрейд абонента на премиум пакет

Задача: Перевести абонента с базового на премиум пакет

Вариант 1: Добавить премиум к базовому

```

# Абонент получит доступ к каналам обоих пакетов
curl -X POST https://your-catenal-domain.com/tv-management/api/v1/packages-subscribers \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "subscriberId": "sK19SW3AAAE.",
  "packageId": "premium-package-id"
}'

```

Вариант 2: Заменить базовый на премиум

```

# Сначала отключаем базовый
curl -X DELETE https://your-catenal-domain.com/tv-management/api/v1/packages-subscribers \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "subscriberId": "sK19SW3AAAE.",

```

```

    "packageId": "basic-package-id"
  }'

# Затем подключаем премиум
curl -X POST https://your-catena-domain.com/tv-management/api/v1/packages-subscribers \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "subscriberId": "sK19SW3AAAE.",
  "packageId": "premium-package-id"
}'

```

Интеграция с биллинговой системой

Задача: Автоматически управлять подписками на основе платежей

Концепция:

1. Биллинговая система отслеживает платежи абонентов
2. При успешной оплате биллинг вызывает API Catena для подключения пакета
3. При окончании подписки биллинг отключает пакет через API
4. Catena автоматически управляет доступом к каналам

Пример webhook от биллинга:

```

import requests

def on_payment_success(subscriber_phone, package_name):
    # 1. Найти абонента по телефону
    subscribers = requests.get(
        f"https://catena.example.com/tv-management/api/v1/subscribers",
        headers={"X-Auth-Token": "your-api-key"}
    ).json()

    subscriber = next(
        s for s in subscribers['subscribers']
        if f"{s['phoneCountryCode']}{s['phone']}" == subscriber_phone
    )

    # 2. Подключить оплаченный пакет
    requests.post(
        "https://catena.example.com/tv-management/api/v1/packages-subscribers",
        headers={"X-Auth-Token": "your-api-key"},
        json={
            "subscriberId": subscriber['subscriberId'],
            "packageId": get_package_id(package_name)
        }
    )

def on_subscription_expired(subscriber_phone, package_name):
    # Аналогично, но через DELETE
    pass

```

Массовая миграция абонентов

Задача: Перенести абонентов из старой системы в Catena

Шаги:

1. Экспортируйте данные абонентов из старой системы (CSV/JSON)
2. Создайте скрипт массового импорта через API
3. Создайте учётные записи в Catena
4. Подключите соответствующие пакеты
5. Оповестите абонентов о переходе на новую систему

Пример скрипта импорта:

```

import csv
import requests

API_URL = "https://catena.example.com/tv-management/api/v1"
API_KEY = "your-api-key"

def import_subscribers(csv_file):

```

```

with open(csv_file, 'r') as f:
    reader = csv.DictReader(f)
    for row in reader:
        # Создать абонента
        response = requests.post(
            f"{API_URL}/subscribers",
            headers={"X-Auth-Token": API_KEY},
            json={
                "name": row['name'],
                "phoneCountryCode": row['country_code'],
                "phone": row['phone']
            }
        )

        subscriber_id = response.json()['subscriberId']

        # Подключить пакеты
        for package_id in row['packages'].split(','):
            requests.post(
                f"{API_URL}/packages-subscribers",
                headers={"X-Auth-Token": API_KEY},
                json={
                    "subscriberId": subscriber_id,
                    "packageId": package_id.strip()
                }
            )

        print(f"Импортирован: {row['name']} ({subscriber_id})")

import_subscribers('subscribers.csv')

```

4.1.12 Лучшие практики

Управление номерами телефонов

Рекомендации:

- **Валидация при вводе** — проверяйте формат номера до отправки в API
- **Уникальность** — один номер телефона = один абонент
- **Международный формат** — храните код страны и номер отдельно
- **Изменение номера** — требуйте подтверждения через SMS на новый номер
- **Деактивация** — при отключении номера оператором своевременно обновляйте данные

Безопасность

Защита токенов воспроизведения:

- Не передавайте `playback_token` третьим лицам
- Используйте HTTPS для всех API запросов
- Регулярно обновляйте токены при подозрении на компрометацию
- Логируйте попытки доступа с неверными токенами

Контроль доступа:

- Ограничивайте количество одновременных сеансов на абонента
- Отслеживайте подозрительную активность (разные IP, разные устройства)
- Блокируйте абонентов при обнаружении мошенничества

Управление подписками

Рекомендации:

- **Плавный переход** — предупреждайте об изменениях в подписках заранее
- **Автоматизация** — интегрируйте с биллингом для автоматического управления
- **Бесплатные пакеты** — используйте free packages в портале для демо-контента
- **Пробные периоды** — временно подключайте премиум пакеты для trial
- **История изменений** — используйте журнал операций для аудита

Коммуникация с абонентами

Когда отправлять уведомления:

- При создании учётной записи
- При изменении подписок
- При окончании оплаченного периода
- При изменении номера телефона
- При блокировке доступа

Каналы коммуникации:

- SMS — для кодов входа и критичных уведомлений
- Email — для информационных рассылок (если есть в вашей системе)
- Push-уведомления — через мобильное приложение
- In-app сообщения — при входе в приложение

4.1.13 Устранение проблем

Абонент не может войти по SMS

Возможные причины:

- Неверно указан номер телефона при регистрации
- SMS не доставлена (проблемы оператора связи)
- Код из SMS истёк
- Номер телефона заблокирован в SMS-шлюзе

Решение:

1. Проверьте номер телефона в учётной записи абонента
2. Убедитесь, что формат номера корректен (+код_страны + номер)
3. Проверьте логи SMS-шлюза на доставку сообщений
4. Попробуйте отправить SMS повторно
5. Если SMS не приходят — проверьте баланс SMS-шлюза и его настройки

Абонент не видит каналы

Возможные причины:

- У абонента нет подключённых пакетов
- Пакеты не содержат каналов
- Токен воспроизведения устарел или недействителен
- Технические проблемы со streaming-сервером

Решение:

1. Проверьте список пакетов в поле `packages` абонента
2. Убедитесь, что пакеты содержат каналы
3. Проверьте, что каналы активны и работают
4. Регенерируйте `playback_token` если необходимо
5. Проверьте логи streaming-сервера

Ошибки при подключении пакета

Возможные причины:

- Пакет уже подключён к абоненту
- Указан неверный `packageId` или `subscriberId`
- Пакет и абонент принадлежат разным порталам
- Пакет не существует или удалён

Решение:

1. Проверьте текущие пакеты абонента через GET `/subscribers/{id}`
2. Убедитесь, что ID пакета и абонента корректны
3. Проверьте, что пакет существует через GET `/packages/{id}`
4. Убедитесь, что `portalId` совпадает у пакета и абонента

Дубликаты номеров телефонов

Проблема: Попытка создать абонента с уже существующим номером

Решение:

- API должен возвращать ошибку при создании дубликата
- Перед созданием проверяйте наличие абонента с таким номером
- Используйте UPDATE вместо CREATE для существующих абонентов
- Реализуйте поиск по номеру телефона в вашем интерфейсе

4.1.14 См. также

- **Управление пакетами каналов** — создание и настройка пакетов для абонентов
- **Управление каналами** — настройка телевизионных каналов
- **Журнал операций** — отслеживание изменений в системе
- **Сеансы воспроизведения** — мониторинг активности просмотра

4.2 Управление подписками

Подписки (subscriptions) — это связи между абонентами и пакетами каналов, которые определяют, к каким каналам имеет доступ каждый абонент. Система подписок является ключевым механизмом монетизации IPTV-сервиса в Catena.

4.2.1 Что такое подписка

Подписка в Catena — это активная связь между абонентом и пакетом каналов. Когда у абонента есть подписка на пакет, он автоматически получает доступ ко всем каналам, входящим в этот пакет.

Ключевая концепция:

Абонент → Подписка → Пакет → Каналы → Просмотр

1. **Абонент** подписывается на один или несколько **пакетов**
2. Каждый **пакет** содержит набор **каналов**
3. Абонент получает доступ ко всем **каналам** из всех своих пакетов
4. При попытке просмотра система проверяет наличие подписки

Основные возможности:

- **Гибкое управление доступом** — подключение и отключение пакетов в реальном времени
- **Множественные подписки** — абонент может быть подписан на несколько пакетов одновременно
- **Бесплатные пакеты** — автоматический доступ к базовому контенту для всех абонентов
- **Журналирование** — полная история всех изменений подписок
- **API-first подход** — простая интеграция с биллинговыми системами

Типичный workflow:

1. Биллинговая система получает оплату от пользователя
2. Биллинг вызывает API Catena для создания подписки
3. Catena немедленно предоставляет доступ к каналам пакета
4. Абонент начинает смотреть каналы
5. По окончании периода биллинг отключает подписку
6. Доступ к платным каналам автоматически блокируется

4.2.2 Жизненный цикл подписки

Создание подписки

Когда создаётся подписка:

- При оплате пакета через биллинговую систему
- При ручном подключении администратором
- При активации промо-кода или бонуса
- При предоставлении пробного периода

Что происходит при создании:

1. Создаётся запись в базе данных о связи абонент-пакет
2. Абонент немедленно получает доступ ко всем каналам пакета
3. Запись добавляется в журнал операций (тип `createPackageSubscriber`)
4. При следующем запросе плеера список доступных каналов обновляется

Активная подписка**Во время действия подписки:**

- Абонент может смотреть все каналы из пакета без ограничений
- Система логирует все сеансы просмотра
- Поле `packages` у абонента содержит ID активных пакетов
- Streaming-сервер проверяет права доступа при каждом запросе потока

Отключение подписки**Когда отключается подписка:**

- По истечении оплаченного периода
- При отмене подписки пользователем
- При блокировке абонента администратором
- При удалении пакета из системы

Что происходит при отключении:

1. Удаляется запись о связи абонент-пакет
2. Абонент немедленно теряет доступ к каналам этого пакета
3. Запись добавляется в журнал операций (тип `deletePackageSubscriber`)
4. Активные сеансы просмотра каналов пакета прерываются

4.2.3 Создание подписки

Через веб-интерфейс

1. **Откройте карточку абонента** в разделе "Абоненты"
2. **Перейдите на вкладку "Подписки"** или "Пакеты"
3. **Нажмите "Добавить подписку"**
4. **Выберите пакет** из выпадающего списка доступных пакетов
5. **Подтвердите добавление**

Абонент немедленно получит доступ ко всем каналам выбранного пакета.

Через Management API**Создать подписку абонента на пакет:**

```
curl -X POST https://your-catenas-domain.com/tv-management/api/v1/packages-subscribers \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "subscriberId": "sK19SW3AAAE.",
  "packageId": "pK19SW3AAAE."
}'
```

Параметры запроса:

- **subscriberId** (обязательно) — ID абонента, которому подключается пакет
- **packageId** (обязательно) — ID пакета для подключения

Ответ:

```
{
  "subscriberId": "sK19SW3AAAE.",
  "packageId": "pK19SW3AAAE.",
  "portalId": "pK19SW3AAAE."
}
```

Важные моменты:

- Абонент и пакет должны принадлежать одному portalу
- Если подписка уже существует, API вернёт ошибку
- Изменения вступают в силу немедленно
- Операция записывается в журнал

4.2.4 Удаление подписки

Через веб-интерфейс

1. **Откройте карточку абонента**
2. **Перейдите на вкладку "Подписки"**
3. **Найдите пакет** в списке активных подписок
4. **Нажмите "Удалить" или "Отключить"**
5. **Подтвердите отключение**

Абонент немедленно потеряет доступ к каналам этого пакета.

Через Management API**Удалить подписку абонента на пакет:**

```
curl -X DELETE https://your-catenal-domain.com/tv-management/api/v1/packages-subscribers \
-H "X-Auth-Token: your-api-key" \
-H "Content-Type: application/json" \
-d '{
  "subscriberId": "sK19SW3AAAE.",
  "packageId": "pK19SW3AAAE."
}'
```

Параметры запроса:

- **subscriberId** (обязательно) — ID абонента
- **packageId** (обязательно) — ID пакета для отключения

Ответ:

HTTP 201 - подписка удалена

Важные моменты:

- Если подписки не существует, API вернёт ошибку
- Активные сеансы просмотра будут прерваны
- Изменения вступают в силу немедленно
- Операция записывается в журнал

4.2.5 Просмотр подписок

Подписки конкретного абонента

Получить список пакетов абонента:

```
curl -X GET https://your-caten-a-domain.com/tv-management/api/v1/subscribers/sK19SW3AAAE. \
-H "X-Auth-Token: your-api-key"
```

Ответ:

```
{
  "subscriberId": "sK19SW3AAAE.",
  "portalId": "pK19SW3AAAE.",
  "name": "Иван Петров",
  "phoneCountryCode": "7",
  "phone": "9161234567",
  "playback_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..",
  "packages": ["pK19SW3AAAE.", "sportK19SW3AAAE."]
}
```

Поле `packages` содержит массив ID всех пакетов, на которые подписан абонент.

Подписчики конкретного пакета

К сожалению, прямого API для получения списка абонентов пакета нет. Используйте журнал операций или получите всех абонентов и отфильтруйте по `packages`:

```
# Получить всех абонентов
curl -X GET https://your-caten-a-domain.com/tv-management/api/v1/subscribers \
-H "X-Auth-Token: your-api-key" \
| jq '.subscribers[] | select(.packages[] | contains("pK19SW3AAAE."))'
```

История подписок через журнал операций

Получить все операции с подписками конкретного абонента:

```
curl -X GET "https://your-caten-a-domain.com/tv-management/api/v1/operations?subscriberId=sK19SW3AAAE.&type=createPackageSubscriber&type=deletePackageSubscriber" \
-H "X-Auth-Token: your-api-key"
```

Получить операции с конкретным пакетом:

```
curl -X GET "https://your-caten-a-domain.com/tv-management/api/v1/operations?packageId=pK19SW3AAAE." \
-H "X-Auth-Token: your-api-key"
```

Ответ:

```
{
  "operations": [
    {
      "operationId": "opK19SW3AAAE.",
      "type": "createPackageSubscriber",
      "subscriberId": "sK19SW3AAAE.",
      "packageId": "pK19SW3AAAE.",
      "portalId": "pK19SW3AAAE.",
      "createdAt": "2024-10-16T10:05:00Z",
      "payload": {
        "packageId": "pK19SW3AAAE.",
        "subscriberId": "sK19SW3AAAE."
      }
    },
    {
      "operationId": "opK19SW3AAAB.",
      "type": "deletePackageSubscriber",
      "subscriberId": "sK19SW3AAAE.",
      "packageId": "pK19SW3AAAE.",
      "portalId": "pK19SW3AAAE.",
      "createdAt": "2024-10-20T15:30:00Z",
      "payload": {
        "packageId": "pK19SW3AAAE.",
        "subscriberId": "sK19SW3AAAE."
      }
    }
  ]
}
```

```
"next": "cursor-for-next-page"
}
```

Типы операций:

- `createPackageSubscriber` — создание подписки
- `deletePackageSubscriber` — удаление подписки
- `autoCreateSubscriber` — автоматическое создание абонента (может включать подписку на базовый пакет)

4.2.6 Бесплатные пакеты портала

Catena поддерживает концепцию "бесплатных пакетов" — пакетов, которые автоматически доступны всем абонентам портала без явного создания подписки.

Концепция бесплатных пакетов

Как это работает:

- В настройках портала определяется список бесплатных пакетов
- Все абоненты портала автоматически получают доступ к каналам из этих пакетов
- Не требуется создавать индивидуальные подписки для каждого абонента
- Идеально для базового контента, демо-каналов, рекламных каналов

Применение:

- **Базовый контент** — федеральные каналы, доступные всем
- **Пробный период** — демо-контент для новых пользователей
- **Промо-каналы** — рекламные и информационные каналы
- **Социально значимые каналы** — обязательные к распространению каналы

Управление бесплатными пакетами

Просмотр бесплатных пакетов портала:

```
curl -X GET https://your-catena-domain.com/tv-management/api/v1/portal \
-H "X-Auth-Token: your-api-key"
```

Ответ:

```
{
  "portalId": "pK19SW3AAAE.",
  "name": "my-iptv-portal",
  "domain": "iptv.example.com",
  "freePackages": ["basicK19SW3AAAE.", "demoK19SW3AAAE."],
  "branding": {
    "title": "My IPTV Service",
    "description": "Premium IPTV streaming"
  }
}
```

Добавить пакет в список бесплатных:

```
curl -X POST https://your-catena-domain.com/tv-management/api/v1/portal/free-packages/basicK19SW3AAAE. \
-H "X-Auth-Token: your-api-key"
```

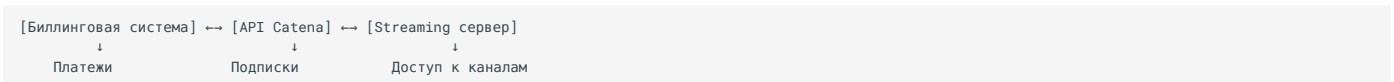
Удалить пакет из списка бесплатных:

```
curl -X DELETE https://your-catena-domain.com/tv-management/api/v1/portal/free-packages/basicK19SW3AAAE. \
-H "X-Auth-Token: your-api-key"
```

Важно:

- Изменения в бесплатных пакетах применяются ко всем абонентам мгновенно
- При добавлении — все абоненты получают доступ к каналам пакета
- При удалении — доступ теряют только те, у кого нет явной подписки

4.2.7 Интеграция с биллинговыми системами

Архитектура интеграции**Типичная схема:****Ответственность биллинга:**

- Приём платежей от пользователей
- Управление тарифами и периодами подписки
- Отслеживание окончания подписок
- Вызов API Catena для подключения/отключения пакетов

Ответственность Catena:

- Управление доступом к каналам
- Проверка прав при просмотре
- Логирование активности абонентов
- Предоставление статистики просмотров

Примеры интеграции**ПРИМЕР 1: WEBHOOK ПРИ ОПЛАТЕ****Биллинг отправляет webhook в ваш сервис при успешной оплате:**

```

from flask import Flask, request
import requests

app = Flask(__name__)

CATENA_API_URL = "https://catena.example.com/tv-management/api/v1"
CATENA_API_KEY = "your-api-key"

@app.route('/billing-webhook', methods=['POST'])
def billing_webhook():
    data = request.json

    if data['event'] == 'payment.success':
        # Получили оплату - активируем подписку
        subscriber_id = get_subscriber_id(data['user_phone'])
        package_id = get_package_id(data['tariff_name'])

        # Создаём подписку в Catena
        response = requests.post(
            f"{CATENA_API_URL}/packages-subscribers",
            headers={"X-Auth-Token": CATENA_API_KEY},
            json={
                "subscriberId": subscriber_id,
                "packageId": package_id
            }
        )

        if response.status_code == 200:
            return {"status": "ok", "message": "Subscription activated"}
        else:
            return {"status": "error", "message": response.text}, 500

    elif data['event'] == 'subscription.expired':
        # Подписка истекла - деактивируем
        subscriber_id = get_subscriber_id(data['user_phone'])
        package_id = get_package_id(data['tariff_name'])
  
```

```

# Удаляем подписку в Catena
response = requests.delete(
    f"{CATENA_API_URL}/packages-subscribers",
    headers={"X-Auth-Token": CATENA_API_KEY},
    json={
        "subscriberId": subscriber_id,
        "packageId": package_id
    }
)

return {"status": "ok", "message": "Subscription deactivated"}

return {"status": "ok"}

def get_subscriber_id(phone):
    """Получить ID абонента Catena по номеру телефона"""
    response = requests.get(
        f"{CATENA_API_URL}/subscribers",
        headers={"X-Auth-Token": CATENA_API_KEY}
    )
    subscribers = response.json()['subscribers']

    for sub in subscribers:
        full_phone = f"+{sub['phoneCountryCode']}{sub['phone']}"
        if full_phone == phone:
            return sub['subscriberId']

    # Если абонент не найден - создаём
    return create_subscriber(phone)

def get_package_id(tariff_name):
    """Сопоставить название тарифа с ID пакета"""
    tariff_mapping = {
        "basic": "basicK19SW3AAAE.",
        "premium": "premiumK19SW3AAAE.",
        "sport": "sportK19SW3AAAE."
    }
    return tariff_mapping.get(tariff_name)

if __name__ == '__main__':
    app.run(port=5000)

```

ПРИМЕР 2: ПЕРИОДИЧЕСКАЯ СИНХРОНИЗАЦИЯ

Регулярная проверка и синхронизация подписок:

```

import requests
from datetime import datetime, timedelta

CATENA_API_URL = "https://catena.example.com/tv-management/api/v1"
CATENA_API_KEY = "your-api-key"
BILLING_DB = "postgresql://billing-db"

def sync_subscriptions():
    """Синхронизация подписок между биллингом и Catena"""

    # 1. Получить активные подписки из биллинга
    active_billing_subscriptions = get_active_subscriptions_from_billing()

    # 2. Получить всех абонентов Catena
    response = requests.get(
        f"{CATENA_API_URL}/subscribers",
        headers={"X-Auth-Token": CATENA_API_KEY}
    )
    catena_subscribers = response.json()['subscribers']

    # 3. Сравнить и синхронизировать
    for billing_sub in active_billing_subscriptions:
        phone = billing_sub['phone']
        package_id = get_package_id(billing_sub['tariff'])

        # Найти абонента в Catena
        catena_sub = find_subscriber_by_phone(catena_subscribers, phone)

        if catena_sub:
            # Проверить, есть ли нужная подписка
            if package_id not in catena_sub['packages']:
                # Подписки нет - создаём
                create_subscription(catena_sub['subscriberId'], package_id)
                print(f"Activated: {phone} -> {package_id}")
            else:
                # Абонента нет - создаём с подпиской
                create_subscriber_with_package(phone, package_id)
                print(f"Created subscriber: {phone}")

    # 4. Отключить истекшие подписки
    for catena_sub in catena_subscribers:
        phone = f"+{catena_sub['phoneCountryCode']}{catena_sub['phone']}"

        for package_id in catena_sub['packages']:
            if not has_active_billing_subscription(phone, package_id):

```

```

        # В биллинге подписки нет - удаляем из Catena
        delete_subscription(catena_sub['subscriberId'], package_id)
        print(f"Deactivated: {phone} -> {package_id}")

def create_subscription(subscriber_id, package_id):
    requests.post(
        f"{CATENA_API_URL}/packages-subscribers",
        headers={"X-Auth-Token": CATENA_API_KEY},
        json={
            "subscriberId": subscriber_id,
            "packageId": package_id
        }
    )

def delete_subscription(subscriber_id, package_id):
    requests.delete(
        f"{CATENA_API_URL}/packages-subscribers",
        headers={"X-Auth-Token": CATENA_API_KEY},
        json={
            "subscriberId": subscriber_id,
            "packageId": package_id
        }
    )

# Запускать эту функцию по расписанию (например, каждый час)
if __name__ == '__main__':
    sync_subscriptions()

```

ПРИМЕР 3: ПРОБНЫЙ ПЕРИОД

Автоматическое предоставление пробного периода новым абонентам:

```

import requests
from datetime import datetime, timedelta

def activate_trial_subscription(phone, trial_days=7):
    """Активировать пробную подписку на N дней"""

    # 1. Создать или получить абонента
    subscriber_id = get_or_create_subscriber(phone)

    # 2. Подключить пробный пакет
    trial_package_id = "trialK19SW3AAAE."

    response = requests.post(
        f"{CATENA_API_URL}/packages-subscribers",
        headers={"X-Auth-Token": CATENA_API_KEY},
        json={
            "subscriberId": subscriber_id,
            "packageId": trial_package_id
        }
    )

    if response.status_code == 200:
        # 3. Запланировать автоматическое отключение
        schedule_subscription_cancellation(
            subscriber_id,
            trial_package_id,
            datetime.now() + timedelta(days=trial_days)
        )

        return {
            "success": True,
            "message": f"Trial activated for {trial_days} days",
            "expires_at": (datetime.now() + timedelta(days=trial_days)).isoformat()
        }

    return {"success": False, "error": response.text}

def schedule_subscription_cancellation(subscriber_id, package_id, cancel_date):
    """Запланировать отключение подписки"""
    # Сохранить в базу задач или использовать планировщик
    # Например, Celery, APScheduler, или cron job
    pass

```

4.2.8 Типичные сценарии использования

Подписка с автопродлением

Задача: Реализовать месячную подписку с автоматическим продлением

Решение:

1. Биллинг списывает оплату каждый месяц
2. При успешном списании биллинг проверяет наличие подписки в Catena
3. Если подписка есть — ничего не делать (она уже активна)
4. Если подписки нет — создать её через API
5. При неудачном списании — удалить подписку через API

```
def process_monthly_renewal(user_id, package_name):
    """Обработка ежемесячного продления"""

    # Попытка списания
    payment_success = billing_charge(user_id, get_package_price(package_name))

    subscriber_id = get_subscriber_id_by_user(user_id)
    package_id = get_package_id(package_name)

    if payment_success:
        # Платёж успешен - убедиться что подписка активна
        ensure_subscription_active(subscriber_id, package_id)
    else:
        # Платёж не прошёл - отключить подписку
        deactivate_subscription(subscriber_id, package_id)
        send_notification(user_id, "payment_failed")
```

Семейная подписка

Задача: Один платёж — доступ для нескольких абонентов (семейный аккаунт)

Решение:

1. В биллинге создать семейный тариф
2. При оплате подключить пакет всем абонентам семьи
3. Хранить связь между абонентами в биллинге

```
def activate_family_subscription(family_id, package_name):
    """Активировать семейную подписку"""

    package_id = get_package_id(package_name)

    # Получить всех членов семьи из биллинга
    family_members = get_family_members(family_id)

    for member in family_members:
        subscriber_id = get_subscriber_id(member['phone'])

        # Подключить пакет каждому
        requests.post(
            f"{CATENA_API_URL}/packages-subscribers",
            headers={"X-Auth-Token": CATENA_API_KEY},
            json={
                "subscriberId": subscriber_id,
                "packageId": package_id
            }
        )

    return {"activated": len(family_members)}
```

Временная акция

Задача: Дать доступ к премиум каналам на выходные

Решение:

1. В пятницу вечером подключить промо-пакет всем активным абонентам
2. В понедельник утром отключить промо-пакет

```
#!/bin/bash
# friday-промо.sh - запускается по cron в пятницу в 18:00

PROMO_PACKAGE_ID="weekendK19SW3AAAE."

# Получить всех абонентов
```

```

SUBSCRIBERS=$(curl -s -X GET "$CATENA_API_URL/subscribers" \
-H "X-Auth-Token: $CATENA_API_KEY" \
| jq -r '.subscribers[].subscriberId')

# Подключить промо-пакет каждому
for SUBSCRIBER_ID in $SUBSCRIBERS; do
  curl -X POST "$CATENA_API_URL/packages-subscribers" \
  -H "X-Auth-Token: $CATENA_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{
    \"subscriberId\": \"$SUBSCRIBER_ID\",
    \"packageId\": \"$PROMO_PACKAGE_ID\"
  }"
done

echo "Promo activated for $(echo "$SUBSCRIBERS" | wc -l) subscribers"

```

```

#!/bin/bash
# monday-cleanup.sh - запускается по cron в понедельник в 06:00

PROMO_PACKAGE_ID="weekendK19SW3AAAE."

SUBSCRIBERS=$(curl -s -X GET "$CATENA_API_URL/subscribers" \
-H "X-Auth-Token: $CATENA_API_KEY" \
| jq -r '.subscribers[] | select(.packages[] | contains("'"$PROMO_PACKAGE_ID")) | .subscriberId')

for SUBSCRIBER_ID in $SUBSCRIBERS; do
  curl -X DELETE "$CATENA_API_URL/packages-subscribers" \
  -H "X-Auth-Token: $CATENA_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{
    \"subscriberId\": \"$SUBSCRIBER_ID\",
    \"packageId\": \"$PROMO_PACKAGE_ID\"
  }"
done

echo "Promo deactivated for $(echo "$SUBSCRIBERS" | wc -l) subscribers"

```

Понижение тарифа

Задача: Абонент переходит с премиум на базовый тариф

Решение:

```

def downgrade_subscription(subscriber_id, from_package, to_package):
    """Понизить тариф абонента"""

    from_package_id = get_package_id(from_package)
    to_package_id = get_package_id(to_package)

    # 1. Отключить премиум пакет
    requests.delete(
        f"{CATENA_API_URL}/packages-subscribers",
        headers={"X-Auth-Token": CATENA_API_KEY},
        json={
            "subscriberId": subscriber_id,
            "packageId": from_package_id
        }
    )

    # 2. Подключить базовый пакет
    requests.post(
        f"{CATENA_API_URL}/packages-subscribers",
        headers={"X-Auth-Token": CATENA_API_KEY},
        json={
            "subscriberId": subscriber_id,
            "packageId": to_package_id
        }
    )

    # 3. Записать в биллинг
    billing_record_downgrade(subscriber_id, from_package, to_package)

    return {"success": True, "new_package": to_package}

```

4.2.9 Лучшие практики

Проектирование пакетов

Рекомендации по структуре пакетов:

- **Базовый пакет** — минимальный набор каналов для всех
- **Тематические пакеты** — спорт, кино, детские, новости
- **Премиум пакеты** — эксклюзивный контент, HD/4K каналы
- **Комбо-пакеты** — несколько тематик в одном (экономия для абонента)

Избегайте:

- Слишком много мелких пакетов — усложняет выбор
- Дублирование каналов между пакетами — путаница в биллинге
- Пересечения пакетов без логики — один канал в 5 разных пакетах

Обработка ошибок

При интеграции с биллингом:

```
def safe_create_subscription(subscriber_id, package_id, retry_count=3):
    """Создание подписки с повторными попытками"""

    for attempt in range(retry_count):
        try:
            response = requests.post(
                f"{CATENA_API_URL}/packages-subscribers",
                headers={"X-Auth-Token": CATENA_API_KEY},
                json={
                    "subscriberId": subscriber_id,
                    "packageId": package_id
                },
                timeout=10
            )

            if response.status_code == 200:
                return {"success": True}
            elif response.status_code == 409:
                # Подписка уже существует - это OK
                return {"success": True, "already_exists": True}
            else:
                # Другая ошибка
                error_msg = response.json().get('message', 'Unknown error')
                log_error(f"Failed to create subscription: {error_msg}")

        except requests.exceptions.Timeout:
            log_warning(f"Timeout on attempt {attempt + 1}")
            if attempt < retry_count - 1:
                time.sleep(2 ** attempt) # Exponential backoff
            continue
        except Exception as e:
            log_error(f"Unexpected error: {str(e)}")
            break

    # Все попытки неудачны - сохранить для ручной обработки
    save_failed_operation("create_subscription", subscriber_id, package_id)
    return {"success": False, "error": "Failed after retries"}
```

Синхронизация состояния

Регулярная сверка данных:

```
def audit_subscriptions():
    """Проверка согласованности подписок между системами"""

    discrepancies = []

    # Получить данные из обеих систем
    billing_subscriptions = get_billing_subscriptions()
    catena_subscriptions = get_catena_subscriptions()

    # Найти расхождения
    for billing_sub in billing_subscriptions:
        if not exists_in_catena(billing_sub, catena_subscriptions):
            discrepancies.append({
```

```

        "type": "missing_in_catena",
        "subscriber": billing_sub['phone'],
        "package": billing_sub['package']
    })

for catena_sub in catena_subscriptions:
    if not exists_in_billing(catena_sub, billing_subscriptions):
        discrepancies.append({
            "type": "missing_in_billing",
            "subscriber": catena_sub['phone'],
            "package": catena_sub['package']
        })

if discrepancies:
    # Отправить уведомление администратору
    send_audit_report(discrepancies)

    # Опционально: автоматически исправить
    auto_fix_discrepancies(discrepancies)

return discrepancies

```

Логирование и мониторинг

Что логировать:

- Все создания и удаления подписок
- Ошибки при вызове API
- Время отклика API Catena
- Несоответствия между биллингом и Catena

Метрики для отслеживания:

```

import prometheus_client as prom

# Метрики Prometheus
subscription_creations = prom.Counter(
    'catena_subscription_creations_total',
    'Total number of subscription creations',
    ['package_name', 'status']
)

subscription_deletions = prom.Counter(
    'catena_subscription_deletions_total',
    'Total number of subscription deletions',
    ['package_name', 'status']
)

api_latency = prom.Histogram(
    'catena_api_latency_seconds',
    'Latency of Catena API calls',
    ['endpoint', 'method']
)

def monitored_create_subscription(subscriber_id, package_id):
    """Создание подписки с мониторингом"""

    package_name = get_package_name(package_id)

    with api_latency.labels('/packages-subscribers', 'POST').time():
        try:
            response = requests.post(...)

            if response.status_code == 200:
                subscription_creations.labels(package_name, 'success').inc()
                return {"success": True}
            else:
                subscription_creations.labels(package_name, 'error').inc()
                return {"success": False}

        except Exception as e:
            subscription_creations.labels(package_name, 'exception').inc()
            raise

```

Уведомления абонентам

Когда отправлять уведомления:

1. **При активации подписки** — "Добро пожаловать! Теперь доступны каналы: ..."
2. **За 3 дня до окончания** — "Ваша подписка истекает через 3 дня"
3. **При продлении** — "Подписка продлена до ..."
4. **При отключении** — "Подписка отключена. Для продления..."
5. **При ошибке оплаты** — "Не удалось списать оплату. Проверьте..."

```
def notify_subscription_activated(subscriber_id, package_name):
    """Уведомление об активации подписки"""

    subscriber = get_subscriber(subscriber_id)
    phone = f"+{subscriber['phoneCountryCode']}{subscriber['phone']}"

    # Получить список каналов пакета
    package = get_package(get_package_id(package_name))
    channels = ", ".join(package['channels'][:5]) # Первые 5 каналов

    message = f"""
    Подписка активирована!

    Пакет: {package_name}
    Доступные каналы: {channels} и другие

    Приятного просмотра!
    """

    send_sms(phone, message)
```

4.2.10 Устранение проблем

Подписка не создаётся

Возможные причины:

- Неверный `subscriberId` или `packageId`
- Абонент и пакет из разных порталов
- Подписка уже существует
- Проблемы с авторизацией API

Решение:

1. Проверьте существование абонента: `GET /subscribers/{id}`
2. Проверьте существование пакета: `GET /packages/{id}`
3. Убедитесь, что `portalId` совпадает
4. Проверьте текущие подписки абонента
5. Проверьте валидность API ключа

Абонент не видит каналы после создания подписки

Возможные причины:

- Пакет не содержит каналов
- Приложение не обновило список каналов
- Проблемы со streaming-сервером

Решение:

1. Проверьте содержимое пакета: `GET /packages/{id}`
2. Убедитесь, что в пакете есть каналы
3. Попросите абонента перезапустить приложение
4. Проверьте `playback_token` абонента
5. Проверьте логи streaming-сервера

Подписка не удаляется**Возможные причины:**

- Подписки не существует (уже удалена)
- Неверные параметры запроса
- Это бесплатный пакет портала (удалить нельзя)

Решение:

1. Проверьте текущие подписки абонента
2. Убедитесь, что это не бесплатный пакет портала
3. Проверьте правильность `subscriberId` и `packageId`
4. Посмотрите журнал операций для этого абонента

Расхождения между биллингом и Catena

Проблема: В биллинге подписка активна, в Catena — нет (или наоборот)

Решение:

1. Реализуйте регулярную синхронизацию (каждые 15-60 минут)
2. Используйте журнал операций для выявления проблем
3. При расхождении приоритет имеет биллинг (источник истины)
4. Логируйте все изменения для анализа

```
def fix_sync_issue(subscriber_id):
    """Исправить рассинхронизацию для абонента"""

    # 1. Получить "правду" из биллинга
    billing_packages = get_billing_packages(subscriber_id)

    # 2. Получить текущее состояние в Catena
    subscriber = get_catena_subscriber(subscriber_id)
    catena_packages = subscriber['packages']

    # 3. Синхронизировать
    for package_id in billing_packages:
        if package_id not in catena_packages:
            # Должна быть, но нет - добавляем
            create_subscription(subscriber_id, package_id)
            log_info(f"Fixed: added {package_id} to {subscriber_id}")

    for package_id in catena_packages:
        if package_id not in billing_packages:
            # Есть, но не должна быть - удаляем
            delete_subscription(subscriber_id, package_id)
            log_info(f"Fixed: removed {package_id} from {subscriber_id}")
```

4.2.11 См. также

- **Управление абонентами** — создание и настройка учётных записей абонентов
- **Управление пакетами каналов** — создание и настройка пакетов
- **Управление каналами** — добавление каналов в пакеты

- **Журнал операций** — отслеживание всех изменений подписок
- **Настройка портала** — конфигурация бесплатных пакетов

5. Мониторинг

5.1 Мониторинг сеансов воспроизведения

Сеансы воспроизведения (play sessions) — это записи о просмотре каналов абонентами. Catena автоматически регистрирует каждое открытие потока и сохраняет подробную информацию о сеансе для мониторинга, аналитики и отладки.

5.1.1 Что такое сеанс воспроизведения

Сеанс воспроизведения — это период времени, когда абонент смотрит конкретный канал. Каждый сеанс содержит информацию о том, кто, что, когда и откуда смотрел.

Жизненный цикл сеанса:

Открытие потока → Активный просмотр → Закрытие потока
(openedAt) (active: true) (closedAt)

Что записывается:

- **Кто смотрит:** subscriberId, токен
- **Что смотрит:** channelId, channelName, programId
- **Когда:** openedAt, closedAt, updatedAt (timestamps)
- **Откуда:** IP адрес, userAgent (плеер/устройство)
- **Сколько:** bytes (объём переданных данных), длительность сеанса
- **Статус:** active (сеанс открыт или закрыт)

Применение:

- **Мониторинг в реальном времени** — кто сейчас смотрит каналы
- **Отладка проблем** — почему абонент не может смотреть канал
- **Аналитика просмотров** — популярные каналы, время просмотра
- **Биллинг** — подсчёт потреблённого трафика
- **Безопасность** — выявление аномалий (один токен из разных IP)
- **Статистика** — отчёты для владельцев контента

5.1.2 Структура сеанса воспроизведения

Основные поля

Идентификаторы:

- **sessionId** — уникальный ID сеанса
- Формат: base64-кодированный Snowflake ID
- Пример: `sessK19SW3AAAE.`
- Генерируется при открытии потока
- **subscriberId** — ID абонента, смотрящего канал
- Связь с учётной записью пользователя
- Пример: `sK19SW3AAAE.`
- **channelId** — ID канала, который смотрят
- Пример: `chK19SW3AAAE.`
- **channelName** — техническое имя канала
- Удобнее для отладки, чем ID
- Пример: `sport1`, `news-hd`
- **programId** — ID программы (если смотрят из архива)
- Null для live просмотра
- Пример: `prK19SW3AAAE.`
- **portalId** — ID портала
- Изоляция данных между порталами
- Пример: `pK19SW3AAAE.`

Временные метки:

- **openedAt** — Unix timestamp открытия сеанса
- Когда абонент начал смотреть
- Пример: `1714233600` (28 апреля 2024, 10:00:00 UTC)
- **closedAt** — Unix timestamp закрытия сеанса
- Когда поток был остановлен
- Null для активных сеансов
- Пример: `1714237200`
- **updatedAt** — Unix timestamp последнего обновления
- Обновляется периодически во время просмотра
- Используется для определения "мёртвых" соединений

Сетевая информация:

- **ip** — IP адрес абонента
- Пример: 192.168.1.100, 2001:db8::1
- Используется для геолокации и выявления аномалий
- **userAgent** — строка User-Agent плеера
- Определяет устройство и приложение
- Примеры:
 - VLC/3.0.16
 - Mozilla/5.0 (Linux; Android 11) AppleWebKit/537.36
 - Catena/1.0 (Android 11; Samsung SM-G991B)

Статистика:

- **active** — флаг активности сеанса
- **true** — сеанс открыт, идёт просмотр
- **false** — сеанс закрыт, просмотр завершён
- **bytes** — объём переданных данных в байтах
- Обновляется во время просмотра
- Пример: 5242880000 (5 ГБ)
- Используется для биллинга по трафику
- **token** — токен воспроизведения абонента
- Используется streaming-сервером для авторизации
- Пример: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...

Устройство:

- **deviceId** — идентификатор устройства
- Если приложение передаёт device ID
- Пример: device_android_samsung_s21
- Помогает отслеживать количество устройств у абонента

5.1.3 Получение списка сеансов

Базовый запрос**Получить список всех сеансов:**

```
curl -X GET https://your-catenas-domain.com/tv-management/api/v1/play-sessions \
-H "X-Auth-Token: your-api-key"
```

Ответ:

```
{
  "sessions": [
    {
      "sessionId": "sessK19SW3AAAE.",
      "subscriberId": "sk19SW3AAAE.",
      "channelId": "chK19SW3AAAE.",
      "channelName": "sport1",
      "programId": null,
      "portalId": "pk19SW3AAAE.",
      "openedAt": 1714233600,
      "closedAt": null,
      "updatedAt": 1714237200,
      "active": true,
      "bytes": 1073741824,
      "ip": "192.168.1.100",
      "userAgent": "Catena/1.0 (Android 11)",
    }
  ]
}
```

```

    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9... ",
    "deviceId": "device_123"
  },
  {
    "sessionId": "sessK19SW3AAAB.",
    "subscriberId": "sK19SW3AAAB.",
    "channelId": "chK19SW3AAAB.",
    "channelName": "news-hd",
    "programId": null,
    "portalId": "pK19SW3AAAE.",
    "openedAt": 1714230000,
    "closedAt": 1714233600,
    "updatedAt": 1714233600,
    "active": false,
    "bytes": 524288000,
    "ip": "10.0.0.50",
    "userAgent": "VLC/3.0.16",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9... ",
    "deviceId": null
  }
],
"next": "cursor-for-next-page"
}

```

Пагинация

Для больших объёмов данных используется cursor-based пагинация:

```

curl -X GET "https://your-caten-domain.com/tv-management/api/v1/play-sessions?cursor=cursor-for-next-page" \
-H "X-Auth-Token: your-api-key"

```

Рекомендации:

- Обработывайте данные постранично
- Используйте фильтры для уменьшения объёма
- Для периодического мониторинга запрашивайте только активные сеансы

5.1.4 Фильтрация сеансов

По абоненту

Получить все сеансы конкретного абонента:

```

curl -X GET "https://your-caten-domain.com/tv-management/api/v1/play-sessions?subscriberId=sK19SW3AAAE." \
-H "X-Auth-Token: your-api-key"

```

Применение:

- Просмотр истории конкретного пользователя
- Отладка проблем абонента
- Анализ паттернов просмотра

Несколько абонентов:

```

curl -X GET "https://your-caten-domain.com/tv-management/api/v1/play-sessions?subscriberId=sK19SW3AAAE.&subscriberId=sK19SW3AAAB." \
-H "X-Auth-Token: your-api-key"

```

По каналу

Получить все сеансы конкретного канала:

```

curl -X GET "https://your-caten-domain.com/tv-management/api/v1/play-sessions?channelId=chK19SW3AAAE." \
-H "X-Auth-Token: your-api-key"

```

Применение:

- Определение популярности канала
- Анализ пиковых нагрузок на канал
- Отладка проблем с конкретным каналом

Несколько каналов:

```
curl -X GET "https://your-catena-domain.com/tv-management/api/v1/play-sessions?channelId=chK19SW3AAAE.&channelId=chK19SW3AAAB." \
-H "X-Auth-Token: your-api-key"
```

По статусу активности**Только активные сеансы (кто смотрит сейчас):**

```
curl -X GET "https://your-catena-domain.com/tv-management/api/v1/play-sessions?active=true" \
-H "X-Auth-Token: your-api-key"
```

Только завершённые сеансы (история):

```
curl -X GET "https://your-catena-domain.com/tv-management/api/v1/play-sessions?active=false" \
-H "X-Auth-Token: your-api-key"
```

Применение:

- **active=true** — мониторинг в реальном времени
- **active=false** — анализ истории, построение отчётов

По времени**Сеансы, открытые после определённого времени:**

```
curl -X GET "https://your-catena-domain.com/tv-management/api/v1/play-sessions?opened_at_gte=1714233600" \
-H "X-Auth-Token: your-api-key"
```

Сеансы, открытые до определённого времени:

```
curl -X GET "https://your-catena-domain.com/tv-management/api/v1/play-sessions?opened_at_lt=1714320000" \
-H "X-Auth-Token: your-api-key"
```

Сеансы в интервале времени:

```
curl -X GET "https://your-catena-domain.com/tv-management/api/v1/play-sessions?opened_at_gte=1714233600&opened_at_lt=1714320000" \
-H "X-Auth-Token: your-api-key"
```

Применение:

- Анализ просмотров за конкретный период
- Построение временных графиков
- Выявление пиковых часов

Преобразование дат в Unix timestamp:

```
# Текущая дата/время
date +%s
# Результат: 1714233600

# Конкретная дата (GNU date)
date -d "2024-04-28 10:00:00" +%s

# macOS
date -j -f "%Y-%m-%d %H:%M:%S" "2024-04-28 10:00:00" +%s
```

Комбинированные фильтры

Активные сеансы конкретного абонента:

```
curl -X GET "https://your-catena-domain.com/tv-management/api/v1/play-sessions?subscriberId=sKl9SW3AAAE.&active=true" \
-H "X-Auth-Token: your-api-key"
```

Сеансы канала за последние 24 часа:

```
# Текущее время минус 24 часа
TIMESTAMP_24H_AGO=$(date -d '24 hours ago' +%s)

curl -X GET "https://your-catena-domain.com/tv-management/api/v1/play-sessions?channelId=chKl9SW3AAAE.&opened_at_gte=$TIMESTAMP_24H_AGO" \
-H "X-Auth-Token: your-api-key"
```

5.1.5 Типичные сценарии использования

Сценарий 1: Мониторинг в реальном времени

Задача: Отобразить dashboard с текущими зрителями

Решение:

```
#!/bin/bash
# realtime-dashboard.sh

API_URL="https://catena.example.com/tv-management/api/v1"
API_KEY="your-api-key"

while true; do
    # Получить активные сеансы
    RESPONSE=$(curl -s -X GET "$API_URL/play-sessions?active=true" \
        -H "X-Auth-Token: $API_KEY")

    # Общее количество зрителей
    TOTAL_VIEWERS=$(echo $RESPONSE | jq '.sessions | length')

    # Топ-5 популярных каналов
    TOP_CHANNELS=$(echo $RESPONSE | jq -r '.sessions | group_by(.channelName) |
        map({channel: .[0].channelName, viewers: length}) |
        sort_by(.viewers) | reverse | .[0:5]')

    clear
    echo "=== Текущие зрители ==="
    echo "Всего: $TOTAL_VIEWERS"
    echo ""
    echo "Топ каналов:"
    echo "$TOP_CHANNELS" | jq -r '.[] | "\(.channel): \(.viewers) зрителей"'

    sleep 10
done
```

Python версия с Prometheus метриками:

```
import requests
import time
from prometheus_client import Gauge, start_http_server

# Метрики
active_sessions = Gauge('catena_active_sessions', 'Number of active sessions')
channel_viewers = Gauge('catena_channel_viewers', 'Viewers per channel', ['channel'])

API_URL = "https://catena.example.com/tv-management/api/v1"
API_KEY = "your-api-key"

def update_metrics():
    response = requests.get(
        f"{API_URL}/play-sessions?active=true",
        headers={"X-Auth-Token": API_KEY}
    )

    sessions = response.json()['sessions']

    # Обновить общее количество
    active_sessions.set(len(sessions))

    # Подсчитать по каналам
    channels = {}
    for session in sessions:
        channel = session['channelName']
        channels[channel] = channels.get(channel, 0) + 1

    # Обновить метрики по каналам
```

```

    for channel, count in channels.items():
        channel_viewers.labels(channel=channel).set(count)

if __name__ == '__main__':
    # Запустить HTTP сервер для Prometheus
    start_http_server(8000)

    while True:
        update_metrics()
        time.sleep(30)

```

Сценарий 2: Отладка проблем абонента

Задача: Абонент жалуется, что не может смотреть канал

Шаги отладки:

1. Проверить активные сеансы абонента:

```

curl -X GET "https://your-catenas-domain.com/tv-management/api/v1/play-sessions?subscriberId=sK19SW3AAAE.&active=true" \
-H "X-Auth-Token: your-api-key"

```

Анализ: - Если сеансов нет — проблема с авторизацией или сетью - Если есть сеанс — проверить `updatedAt` (недавно ли обновлялся) - Проверить IP и `userAgent` — соответствует ли устройству абонента

1. Проверить историю последних сеансов:

```

# Последние 1 час
TIMESTAMP_1H_AGO=$(date -d '1 hour ago' +%s)

curl -X GET "https://your-catenas-domain.com/tv-management/api/v1/play-sessions?subscriberId=sK19SW3AAAE.&opened_at_gte=$TIMESTAMP_1H_AGO" \
-H "X-Auth-Token: your-api-key"

```

Что искать: - Частые переподключения (много коротких сеансов) - Низкий объём переданных данных (проблемы со stream) - Разные IP адреса (абонент переключается между сетями)

1. Проверить, может ли абонент смотреть конкретный канал:

```

# Проверить подписки
curl -X GET "https://your-catenas-domain.com/tv-management/api/v1/subscribers/sK19SW3AAAE." \
-H "X-Auth-Token: your-api-key" \
| jq '.packages'

# Проверить, в каких пакетах этот канал
curl -X GET "https://your-catenas-domain.com/tv-management/api/v1/channels/chK19SW3AAAE." \
-H "X-Auth-Token: your-api-key" \
| jq '.packages'

```

Сценарий 3: Выявление аномалий

Задача: Обнаружить подозрительную активность (account sharing)

Индикаторы аномалий:

1. Одновременный просмотр с разных IP:

```

import requests
from collections import defaultdict

def detect_concurrent_ips():
    response = requests.get(
        f"{API_URL}/play-sessions?active=true",
        headers={"X-Auth-Token": API_KEY}
    )

    sessions = response.json()['sessions']

    # Группировка по абонентам
    subscriber_ips = defaultdict(set)
    for session in sessions:
        subscriber_id = session['subscriberId']
        ip = session['ip']
        subscriber_ips[subscriber_id].add(ip)

    # Найти абонентов с multiple IP
    anomalies = []
    for subscriber, ips in subscriber_ips.items():

```

```

        if len(ips) > 1:
            anomalies.append({
                'subscriber': subscriber,
                'ips': list(ips),
                'count': len(ips)
            })

    return anomalies

# Вывести аномалии
for anomaly in detect_concurrent_ips():
    print(f"△ Абонент {anomaly['subscriber']} смотрит с {anomaly['count']} IP:")
    for ip in anomaly['ips']:
        print(f"    - {ip}")

```

1. Слишком много одновременных сеансов:

```

def detect_excessive_sessions(max_sessions=3):
    response = requests.get(
        f"{API_URL}/play-sessions?active=true",
        headers={"X-Auth-Token": API_KEY}
    )

    sessions = response.json()['sessions']

    subscriber_sessions = defaultdict(list)
    for session in sessions:
        subscriber_sessions[session['subscriberId']].append(session)

    violations = []
    for subscriber, sessions in subscriber_sessions.items():
        if len(sessions) > max_sessions:
            violations.append({
                'subscriber': subscriber,
                'session_count': len(sessions),
                'channels': [s['channelName'] for s in sessions]
            })

    return violations

```

1. Необычные User-Agents:

```

def detect_unusual_user_agents():
    # Получить историю за последние 7 дней
    timestamp_7d = int(time.time()) - 7 * 24 * 3600

    response = requests.get(
        f"{API_URL}/play-sessions?opened_at_gte={timestamp_7d}",
        headers={"X-Auth-Token": API_KEY}
    )

    sessions = response.json()['sessions']

    subscriber_agents = defaultdict(set)
    for session in sessions:
        subscriber_agents[session['subscriberId']].add(session['userAgent'])

    # Найти тех, кто использует много разных устройств
    suspicious = []
    for subscriber, agents in subscriber_agents.items():
        if len(agents) > 5: # Более 5 разных устройств
            suspicious.append({
                'subscriber': subscriber,
                'devices': list(agents)
            })

    return suspicious

```

Сценарий 4: Аналитика и отчёты

Задача: Построить отчёт о просмотрах за месяц

Скрипт для сбора статистики:

```

import requests
import json
from datetime import datetime, timedelta
from collections import defaultdict

API_URL = "https://catena.example.com/tv-management/api/v1"
API_KEY = "your-api-key"

def get_monthly_report(year, month):
    # Начало и конец месяца
    start_date = datetime(year, month, 1)
    if month == 12:

```

```

        end_date = datetime(year + 1, 1, 1)
    else:
        end_date = datetime(year, month + 1, 1)

    start_ts = int(start_date.timestamp())
    end_ts = int(end_date.timestamp())

    # Получить все сеансы за месяц
    all_sessions = []
    cursor = None

    while True:
        url = f"{API_URL}/play-sessions?opened_at_gte={start_ts}&opened_at_lt={end_ts}"
        if cursor:
            url += f"&cursor={cursor}"

        response = requests.get(url, headers={"X-Auth-Token": API_KEY})
        data = response.json()

        all_sessions.extend(data['sessions'])

        cursor = data.get('next')
        if not cursor:
            break

    # Анализ
    stats = {
        'total_sessions': len(all_sessions),
        'unique_subscribers': len(set(s['subscriberId'] for s in all_sessions)),
        'total_bytes': sum(s['bytes'] for s in all_sessions),
        'channels': defaultdict(int),
        'daily': defaultdict(int)
    }

    for session in all_sessions:
        # По каналам
        stats['channels'][session['channelName']] += 1

        # По дням
        day = datetime.fromtimestamp(session['openedAt']).strftime('%Y-%m-%d')
        stats['daily'][day] += 1

    # Сортировка каналов по популярности
    stats['top_channels'] = sorted(
        stats['channels'].items(),
        key=lambda x: x[1],
        reverse=True
    )[:10]

    return stats

# Получить отчёт за апрель 2024
report = get_monthly_report(2024, 4)

print(f"Отчёт за апрель 2024")
print(f"Всего сеансов: {report['total_sessions']}")
print(f"Уникальных зрителей: {report['unique_subscribers']}")
print(f"Передано данных: {report['total_bytes'] / 1024**3:.2f} ГБ")
print(f"\nТоп-10 каналов:")
for channel, views in report['top_channels']:
    print(f"    {channel}: {views} просмотров")

```

Сценарий 5: Биллинг по трафику

Задача: Подсчитать потреблённый трафик каждым абонентом

```

def calculate_traffic_billing(start_date, end_date, price_per_gb=0.5):
    """
    Подсчитать стоимость трафика для каждого абонента
    price_per_gb: цена за 1 ГБ в долларах
    """
    start_ts = int(start_date.timestamp())
    end_ts = int(end_date.timestamp())

    # Получить все сеансы за период
    sessions = []
    cursor = None

    while True:
        url = f"{API_URL}/play-sessions?opened_at_gte={start_ts}&opened_at_lt={end_ts}"
        if cursor:
            url += f"&cursor={cursor}"

        response = requests.get(url, headers={"X-Auth-Token": API_KEY})
        data = response.json()
        sessions.extend(data['sessions'])

        cursor = data.get('next')
        if not cursor:
            break

```

```

# Группировка по абонентам
subscriber_traffic = defaultdict(int)
for session in sessions:
    subscriber_traffic[session['subscriberId']] += session['bytes']

# Расчёт стоимости
billing = []
for subscriber_id, bytes_used in subscriber_traffic.items():
    gb_used = bytes_used / 1024**3
    cost = gb_used * price_per_gb

    billing.append({
        'subscriber_id': subscriber_id,
        'bytes': bytes_used,
        'gb': round(gb_used, 2),
        'cost': round(cost, 2)
    })

# Сортировка по стоимости
billing.sort(key=lambda x: x['cost'], reverse=True)

return billing

# Рассчитать за последний месяц
end = datetime.now()
start = end - timedelta(days=30)

billing = calculate_traffic_billing(start, end)

print("Топ-10 потребителей трафика:")
for item in billing[:10]:
    print(f"Абонент {item['subscriber_id']}: {item['gb']} ГБ = ${item['cost']}")

```

5.1.6 Лучшие практики

Периодический сбор данных

Рекомендации:

- **Активные сеансы:** опрашивать каждые 30-60 секунд для мониторинга
- **История:** собирать раз в день для анализа
- **Архивирование:** перемещать старые данные (>30 дней) в холодное хранилище

Пример cron job:

```

# Каждую минуту – мониторинг активных сеансов
*/1 * * * * /usr/local/bin/monitor-active-sessions.sh

# Каждый час – сбор статистики
0 * * * * /usr/local/bin/collect-hourly-stats.sh

# Каждый день в 01:00 – генерация отчётов
0 1 * * * /usr/local/bin/generate-daily-report.sh

```

Оптимизация запросов

Используйте фильтры для уменьшения объёма данных:

```

# ПЛОХО – получить все сеансы
curl -X GET "$API_URL/play-sessions"

# ХОРОШО – только активные
curl -X GET "$API_URL/play-sessions?active=true"

# ЛУЧШЕ – активные за последний час
TIMESTAMP_1H=$(date -d '1 hour ago' +%s)
curl -X GET "$API_URL/play-sessions?active=true&opened_at_gte=$TIMESTAMP_1H"

```

Хранение исторических данных

Стратегия хранения:

```

import sqlite3
from datetime import datetime

def archive_sessions_to_db(db_path='sessions.db'):
    """Сохранить завершённые сеансы в локальную БД"""

```

```

# Подключение к SQLite
conn = sqlite3.connect(db_path)
cursor = conn.cursor()

# Создать таблицу (если не существует)
cursor.execute('''
    CREATE TABLE IF NOT EXISTS sessions (
        session_id TEXT PRIMARY KEY,
        subscriber_id TEXT,
        channel_id TEXT,
        channel_name TEXT,
        opened_at INTEGER,
        closed_at INTEGER,
        bytes INTEGER,
        ip TEXT,
        user_agent TEXT
    )
''')

# Получить закрытые сеансы за последние 24 часа
timestamp_24h = int(time.time()) - 24 * 3600

response = requests.get(
    f"{API_URL}/play-sessions?active=false&opened_at_gte={timestamp_24h}",
    headers={"X-Auth-Token": API_KEY}
)

sessions = response.json()['sessions']

# Сохранить в БД
for session in sessions:
    cursor.execute('''
        INSERT OR IGNORE INTO sessions VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
    ''', (
        session['sessionId'],
        session['subscriberId'],
        session['channelId'],
        session['channelName'],
        session['openedAt'],
        session['closedAt'],
        session['bytes'],
        session['ip'],
        session['userAgent']
    ))

conn.commit()
conn.close()

return len(sessions)

# Запускать ежедневно
archived = archive_sessions_to_db()
print(f"Archived {archived} sessions")

```

Мониторинг аномалий

Настройка алертов:

```

def check_anomalies_and_alert():
    """Проверка аномалий и отправка уведомлений"""

    issues = []

    # 1. Проверить слишком длинные сеансы (>24 часа)
    threshold = int(time.time()) - 24 * 3600

    response = requests.get(
        f"{API_URL}/play-sessions?active=true",
        headers={"X-Auth-Token": API_KEY}
    )

    for session in response.json()['sessions']:
        if session['openedAt'] < threshold:
            issues.append({
                'type': 'long_session',
                'session_id': session['sessionId'],
                'subscriber': session['subscriberId'],
                'duration_hours': (time.time() - session['openedAt']) / 3600
            })

    # 2. Проверить account sharing
    concurrent_ips = detect_concurrent_ips()
    for anomaly in concurrent_ips:
        if anomaly['count'] > 2:
            issues.append({
                'type': 'account_sharing',
                'subscriber': anomaly['subscriber'],
                'ip_count': anomaly['count']
            })

```

```
# 3. Отправить уведомления
if issues:
    send_alert(issues)

return issues

def send_alert(issues):
    """Отправка уведомлений (email, Slack, Telegram)"""
    message = "⚠ Обнаружены аномалии:\n\n"

    for issue in issues:
        if issue['type'] == 'long_session':
            message += f"- Долгий сеанс: {issue['session_id']} ({issue['duration_hours']:.1f}ч)\n"
        elif issue['type'] == 'account_sharing':
            message += f"- Account sharing: {issue['subscriber']} ({issue['ip_count']} IP)\n"

    # Отправить в Slack
    # requests.post(SLACK_WEBHOOK_URL, json={'text': message})

    print(message)
```

5.1.7 Устранение проблем

Сеансы не создаются

Проблема: Абоненты смотрят, но сеансы не появляются в API

Возможные причины:

1. Streaming-сервер не интегрирован с Management API
2. Неверная конфигурация webhook на streaming-сервере
3. Проблемы с сетью между серверами

Решение:

1. Проверьте конфигурацию streaming-сервера (Flussonic)
2. Убедитесь, что webhook настроен на Management API
3. Проверьте логи streaming-сервера на ошибки

Сеансы не закрываются

Проблема: Сеансы остаются активными даже после остановки просмотра

Причины:

- Абонент закрыл приложение без корректной остановки потока
- Потеря сетевого соединения
- Streaming-сервер не отправил webhook о закрытии

Решение:

- Сеансы имеют timeout (обычно 5-10 минут неактивности)
- Проверяйте поле `updatedAt` — если давно не обновлялось, сеанс "мёртвый"
- Настройте автоматическую очистку "зависших" сеансов

Неточные данные по трафику

Проблема: Поле `bytes` не соответствует реальному трафику

Причины:

- Streaming-сервер обновляет счётчик периодически, а не в реальном времени
- Сеанс был прерван до финального обновления
- Разные методы подсчёта (payload vs full packets)

Решение:

- Используйте данные как приблизительные оценки
- Для точного биллинга используйте логи streaming-сервера
- Учитывайте задержку обновления (обычно 30-60 секунд)

5.1.8 См. также

- **Управление абонентами** – создание и настройка учётных записей
- **Управление каналами** – настройка телевизионных каналов
- **Журнал операций** – аудит действий в системе
- **Управление порталами** – изоляция данных между порталами

5.2 Журнал операций

Журнал операций (operations log) — это полный аудит-лог всех действий с абонентами, пакетами и подписками в Catena. Каждое изменение записывается с временной меткой, что позволяет отслеживать историю и **производить расчёты для биллинга**.

5.2.1 Что такое операция

Операция — это запись о конкретном действии, выполненном в системе. Каждая операция содержит информацию о том, что было сделано, когда и с какими объектами.

Зачем нужен журнал операций:

- **Расчёт доходов** — подсчёт созданных и отменённых подписок для биллинга
- **Аудит действий** — кто, что и когда делал в системе
- **Отладка проблем** — история изменений для выявления причин
- **Аналитика бизнеса** — метрики роста, churn rate, популярные пакеты
- **Безопасность** — отслеживание подозрительных действий
- **Compliance** — доказательства для регуляторов и аудиторов

Что записывается:

```
Создание подписки → Операция createPackageSubscriber
↓
Начало биллингового периода
↓
Отмена подписки → Операция deletePackageSubscriber
↓
Расчёт стоимости = (дата отмены - дата создания) × цена
```

5.2.2 Типы операций

Операции с абонентами

autoCreateSubscriber — автоматическое создание абонента

- Происходит при первом входе через SMS (если включена автоматическая регистрация)
- Система создаёт учётную запись "на лету"
- Может автоматически подключить бесплатные пакеты

createSubscriber — ручное создание абонента

- Администратор или биллинг создали учётную запись
- Через веб-интерфейс или Management API
- Обычно с последующим подключением пакетов

deleteSubscriber — удаление абонента

- Полное удаление учётной записи
- Автоматически отменяет все подписки
- Необратимое действие

disableSubscriber — блокировка абонента

- Временная блокировка доступа
- Подписки сохраняются, но доступ к просмотру блокируется
- Используется при неоплате, нарушениях правил

enableSubscriber — разблокировка абонента

- Восстановление доступа после блокировки
- Подписки остаются активными

Операции с подписками

createPackageSubscriber — создание подписки

- Подключение пакета к абоненту
- **Ключевая операция для биллинга** — начало оплачиваемого периода
- Записывается дата начала подписки

deletePackageSubscriber — удаление подписки

- Отключение пакета от абонента
- **Ключевая операция для биллинга** — конец оплачиваемого периода
- Используется для расчёта стоимости

Операции с пакетами

createPackage — создание пакета

- Создан новый тарифный план
- Аудит для отслеживания изменений в продуктовой линейке

deletePackage — удаление пакета

- Пакет удалён из системы
- Все подписки на него должны быть предварительно отменены

5.2.3 Структура операции

Основные поля

operationId — уникальный идентификатор операции

- Формат: base64-кодированный Snowflake ID
- Пример: `oK19SW3AAAE`
- Генерируется автоматически при создании записи

type — тип операции

- Один из predefined типов (см. выше)
- Используется для фильтрации и группировки
- Пример: `createPackageSubscriber`

portalId — идентификатор портала

- К какому portalу относится операция
- Используется для изоляции данных между порталами
- Пример: `pK19SW3AAAE`

subscriberId — идентификатор абонента (опционально)

- Присутствует в операциях, связанных с абонентами
- Null для операций с пакетами
- Пример: `sK19SW3AAAE.`

packageld — идентификатор пакета (опционально)

- Присутствует в операциях, связанных с пакетами
- Null для операций создания/удаления абонентов
- Пример: `pkK19SW3AAAE.`

createdAt — время создания операции

- Формат: ISO 8601 timestamp
- Пример: `2024-10-16T10:00:00Z`
- **Ключевое поле для биллинга** — точная дата события

updatedAt — время последнего обновления

- Обычно совпадает с `createdAt`
- Может отличаться, если операция была изменена
- Пример: `2024-10-16T10:00:00Z`

payload — дополнительные данные операции

- JSON объект с деталями операции
- Содержимое зависит от типа операции
- Примеры:
 - `{"subscriberId": "sK19SW3AAAE.", "name": "John Doe"}`
 - `{"packageId": "pkK19SW3AAAE.", "packageName": "premium"}`

5.2.4 Получение списка операций

Базовый запрос

Получить все операции:

```
curl -X GET https://your-catena-domain.com/tv-management/api/v1/operations \
-H "X-Auth-Token: your-api-key"
```

Ответ:

```
{
  "operations": [
    {
      "operationId": "opK19SW3AAAE.",
      "type": "createSubscriber",
      "portalId": "pK19SW3AAAE.",
      "subscriberId": "sK19SW3AAAE.",
      "packageId": null,
      "createdAt": "2024-10-16T10:00:00Z",
      "updatedAt": "2024-10-16T10:00:00Z",
      "payload": {
        "name": "Иван Петров",
        "phone": "+79161234567"
      }
    },
    {
      "operationId": "opK19SW3AAAB.",
      "type": "createPackageSubscriber",
      "portalId": "pK19SW3AAAE.",
      "subscriberId": "sK19SW3AAAE.",
      "packageId": "pkK19SW3AAAE.",
      "createdAt": "2024-10-16T10:05:00Z",
      "updatedAt": "2024-10-16T10:05:00Z",
      "payload": {
```

```

      "packageName": "premium"
    }
  ],
  "next": "cursor-for-next-page"
}

```

Пагинация

Для больших объёмов используется cursor-based пагинация:

```

curl -X GET "https://your-catenal-domain.com/tv-management/api/v1/operations?cursor=cursor-for-next-page" \
-H "X-Auth-Token: your-api-key"

```

5.2.5 Фильтрация операций

По абоненту

Все операции конкретного абонента:

```

curl -X GET "https://your-catenal-domain.com/tv-management/api/v1/operations?subscriberId=sK19SW3AAAE." \
-H "X-Auth-Token: your-api-key"

```

Применение:

- История изменений учётной записи
- Аудит действий с абонентом
- Расчёт стоимости подписок абонента

Несколько абонентов:

```

curl -X GET "https://your-catenal-domain.com/tv-management/api/v1/operations?subscriberId=sK19SW3AAAE.&subscriberId=sK19SW3AAAB." \
-H "X-Auth-Token: your-api-key"

```

По пакету

Все операции с конкретным пакетом:

```

curl -X GET "https://your-catenal-domain.com/tv-management/api/v1/operations?packageId=pkK19SW3AAAE." \
-H "X-Auth-Token: your-api-key"

```

Применение:

- Подсчёт активаций пакета
- Анализ популярности тарифного плана
- Расчёт доходов от конкретного пакета

По типу операции

Только создания подписок:

```

curl -X GET "https://your-catenal-domain.com/tv-management/api/v1/operations?type=createPackageSubscriber" \
-H "X-Auth-Token: your-api-key"

```

Только удаления подписок:

```

curl -X GET "https://your-catenal-domain.com/tv-management/api/v1/operations?type=deletePackageSubscriber" \
-H "X-Auth-Token: your-api-key"

```

Несколько типов:

```

curl -X GET "https://your-catenal-domain.com/tv-management/api/v1/operations?type=createPackageSubscriber&type=deletePackageSubscriber" \
-H "X-Auth-Token: your-api-key"

```

Применение:

- Фокусировка на операциях, влияющих на биллинг
- Подсчёт новых подписок (создания)
- Анализ оттока (удаления)

По времени**Операции после определённой даты:**

```
curl -X GET "https://your-catena-domain.com/tv-management/api/v1/operations?created_at_gte=2024-10-01" \
-H "X-Auth-Token: your-api-key"
```

Операции до определённой даты:

```
curl -X GET "https://your-catena-domain.com/tv-management/api/v1/operations?created_at_lt=2024-11-01" \
-H "X-Auth-Token: your-api-key"
```

Операции за период (месяц):

```
curl -X GET "https://your-catena-domain.com/tv-management/api/v1/operations?created_at_gte=2024-10-01&created_at_lt=2024-11-01" \
-H "X-Auth-Token: your-api-key"
```

Применение:

- Ежемесячные отчёты
- Расчёты за биллинговый период
- Анализ трендов по времени

Формат даты:

- ISO 8601 format: `YYYY-MM-DD` или `YYYY-MM-DDTHH:MM:SSZ`
- Примеры: `2024-10-01`, `2024-10-16T10:00:00Z`
- Timezone: UTC

5.2.6 Расчёты для биллинга

Подсчёт доходов за период

Задача: Рассчитать доход за октябрь 2024

Логика расчёта:

1. Получить все операции `createPackageSubscriber` за октябрь
2. Для каждой подписки определить длительность
3. Умножить на стоимость пакета
4. Суммировать

Скрипт расчёта:

```
import requests
from datetime import datetime, timedelta
from collections import defaultdict

API_URL = "https://catena.example.com/tv-management/api/v1"
API_KEY = "your-api-key"

# Цены пакетов (храните в вашей биллинговой БД)
PACKAGE_PRICES = {
    "basicK19SW3AAAE.": 10.0,      # $10/месяц
    "premiumK19SW3AAAE.": 20.0,   # $20/месяц
    "sportK19SW3AAAE.": 15.0     # $15/месяц
}

def calculate_monthly_revenue(year, month):
```

```

"""
Рассчитать доход за месяц на основе операций
"""

# Границы месяца
start_date = f"{year}-{month:02d}-01"
if month == 12:
    end_date = f"{year + 1}-01-01"
else:
    end_date = f"{year}-{month + 1:02d}-01"

# 1. Получить все создания подписок за месяц
subscriptions = get_operations(
    type="createPackageSubscriber",
    created_at_gte=start_date,
    created_at_lt=end_date
)

# 2. Получить все удаления подписок
cancellations = get_operations(
    type="deletePackageSubscriber",
    created_at_gte=start_date,
    created_at_lt=end_date
)

# 3. Сгруппировать по пакетам
revenue_by_package = defaultdict(lambda: {
    'created': 0,
    'cancelled': 0,
    'revenue': 0.0
})

# Подсчитать созданные подписки
for op in subscriptions:
    package_id = op['packageId']
    price = PACKAGE_PRICES.get(package_id, 0)

    revenue_by_package[package_id]['created'] += 1
    revenue_by_package[package_id]['revenue'] += price

# Подсчитать отменённые (возвраты)
for op in cancellations:
    package_id = op['packageId']
    revenue_by_package[package_id]['cancelled'] += 1

# 4. Итоговый отчёт
total_revenue = 0
report = []

for package_id, stats in revenue_by_package.items():
    package_name = get_package_name(package_id)
    total_revenue += stats['revenue']

    report.append({
        'package': package_name,
        'created': stats['created'],
        'cancelled': stats['cancelled'],
        'net_growth': stats['created'] - stats['cancelled'],
        'revenue': stats['revenue']
    })

return {
    'total_revenue': total_revenue,
    'packages': sorted(report, key=lambda x: x['revenue'], reverse=True)
}

def get_operations(**filters):
    """Получить операции с фильтрами и пагинацией"""
    operations = []
    cursor = None

    while True:
        # Построить URL с параметрами
        params = []
        for key, value in filters.items():
            if isinstance(value, list):
                for v in value:
                    params.append(f"{key}={v}")
            else:
                params.append(f"{key}={value}")

        if cursor:
            params.append(f"cursor={cursor}")

        url = f"{API_URL}/operations"
        if params:
            url += "?" + "&".join(params)

        response = requests.get(url, headers={"X-Auth-Token": API_KEY})
        data = response.json()

        operations.extend(data['operations'])

        cursor = data.get('next')
        if not cursor:

```

```

        break

    return operations

# Пример использования
revenue = calculate_monthly_revenue(2024, 10)

print(f"Доход за октябрь 2024: ${revenue['total_revenue']:.2f}")
print("\nПо пакетам:")
for pkg in revenue['packages']:
    print(f"    {pkg['package']}: {pkg['created']} активаций, ${pkg['revenue']:.2f}")
    print(f"        Отмен: {pkg['cancelled']}, Чистый рост: {pkg['net_growth']}")

```

Расчёт пропорциональной оплаты (pro-rata)

Задача: Абонент подключил пакет 15-го числа, а месяц закончился 30-го. Сколько взять?

Решение:

```

from datetime import datetime, timedelta

def calculate_prorata_billing(subscriber_id, start_date, end_date):
    """
    Рассчитать пропорциональную стоимость подписок
    """
    # Получить все операции абонента за период
    operations = get_operations(
        subscriber_id=subscriber_id,
        created_at_gte=start_date.isoformat(),
        created_at_lte=end_date.isoformat()
    )

    # Отфильтровать операции с подписками
    subscriptions = {} # package_id: (create_date, delete_date or None)

    for op in operations:
        package_id = op['packageId']
        op_date = datetime.fromisoformat(op['createdAt'].replace('Z', '+00:00'))

        if op['type'] == 'createPackageSubscriber':
            if package_id not in subscriptions:
                subscriptions[package_id] = {'created': op_date, 'deleted': None}

        elif op['type'] == 'deletePackageSubscriber':
            if package_id in subscriptions:
                subscriptions[package_id]['deleted'] = op_date

    # Рассчитать стоимость для каждой подписки
    total_cost = 0.0
    billing_details = []

    for package_id, dates in subscriptions.items():
        created = dates['created']
        deleted = dates['deleted'] or end_date

        # Количество дней использования
        days_used = (deleted - created).days

        # Стоимость пакета в день
        monthly_price = PACKAGE_PRICES.get(package_id, 0)
        daily_price = monthly_price / 30

        # Пропорциональная стоимость
        cost = daily_price * days_used
        total_cost += cost

        billing_details.append({
            'package_id': package_id,
            'package_name': get_package_name(package_id),
            'created': created.isoformat(),
            'deleted': deleted.isoformat(),
            'days': days_used,
            'monthly_price': monthly_price,
            'cost': round(cost, 2)
        })

    return {
        'subscriber_id': subscriber_id,
        'period': f"{start_date.date()} - {end_date.date()}",
        'total_cost': round(total_cost, 2),
        'items': billing_details
    }

# Пример: расчёт за октябрь
bill = calculate_prorata_billing(
    subscriber_id="sK19SW3AAAE.",
    start_date=datetime(2024, 10, 1),
    end_date=datetime(2024, 11, 1)
)

```

```
print(f"Абонент: {bill['subscriber_id']}")
print(f"Период: {bill['period']}")
print(f"К оплате: ${bill['total_cost']}")
print("\nДетализация:")
for item in bill['items']:
    print(f"    {item['package_name']}: {item['days']} дней × ${item['monthly_price']}/30 = ${item['cost']}")
```

Подсчёт метрик роста

Задача: Рассчитать MRR (Monthly Recurring Revenue) и churn rate

```
def calculate_growth_metrics(year, month):
    """Метрики роста бизнеса"""

    start_date = f"{year}-{month:02d}-01"
    if month == 12:
        end_date = f"{year + 1}-01-01"
    else:
        end_date = f"{year}-{month + 1:02d}-01"

    # Получить операции за месяц
    creates = get_operations(
        type="createPackageSubscriber",
        created_at_gte=start_date,
        created_at_lt=end_date
    )

    deletes = get_operations(
        type="deletePackageSubscriber",
        created_at_gte=start_date,
        created_at_lt=end_date
    )

    # Подсчитать метрики
    new_subscriptions = len(creates)
    cancelled_subscriptions = len(deletes)

    # MRR (упрощённо - без учёта разных цен пакетов)
    avg_package_price = 15.0 # средняя цена пакета
    new_mrr = new_subscriptions * avg_package_price
    lost_mrr = cancelled_subscriptions * avg_package_price
    net_mrr_change = new_mrr - lost_mrr

    # Churn rate
    # Для точного расчёта нужно знать количество активных подписок на начало месяца
    active_subscriptions_start = get_active_subscriptions_count(start_date)
    churn_rate = (cancelled_subscriptions / active_subscriptions_start * 100) if active_subscriptions_start > 0 else 0

    return {
        'month': f"{year}-{month:02d}",
        'new_subscriptions': new_subscriptions,
        'cancelled_subscriptions': cancelled_subscriptions,
        'net_growth': new_subscriptions - cancelled_subscriptions,
        'new_mrr': new_mrr,
        'lost_mrr': lost_mrr,
        'net_mrr_change': net_mrr_change,
        'churn_rate': round(churn_rate, 2)
    }

# Отчёт за октябрь 2024
metrics = calculate_growth_metrics(2024, 10)

print(f"Метрики за {metrics['month']}:")
print(f"    Новых подписок: {metrics['new_subscriptions']}")
print(f"    Отменено: {metrics['cancelled_subscriptions']}")
print(f"    Чистый рост: {metrics['net_growth']}")
print(f"    Новый MRR: ${metrics['new_mrr']}")
print(f"    Потерянный MRR: ${metrics['lost_mrr']}")
print(f"    Изменение MRR: ${metrics['net_mrr_change']:.2f}")
print(f"    Churn rate: {metrics['churn_rate']}%")
```

5.2.7 Типичные сценарии использования

Сценарий 1: Ежемесячный биллинговый отчёт

Задача: Подготовить счета для всех абонентов за месяц

Решение:

```
import csv
from datetime import datetime

def generate_monthly_invoices(year, month, output_csv='invoices.csv'):
    """Генерация счетов на основе журнала операций"""
```

```

start_date = datetime(year, month, 1)
if month == 12:
    end_date = datetime(year + 1, 1, 1)
else:
    end_date = datetime(year, month + 1, 1)

# Получить всех абонентов
response = requests.get(
    f"{API_URL}/subscribers",
    headers={"X-Auth-Token": API_KEY}
)
subscribers = response.json()['subscribers']

# Для каждого абонента рассчитать стоимость
invoices = []

for subscriber in subscribers:
    subscriber_id = subscriber['subscriberId']

    # Рассчитать pro-rata биллинг
    bill = calculate_prorata_billing(
        subscriber_id,
        start_date,
        end_date
    )

    if bill['total_cost'] > 0:
        invoices.append({
            'subscriber_id': subscriber_id,
            'subscriber_name': subscriber['name'],
            'phone': f"+{subscriber['phoneCountryCode']}{subscriber['phone']}",
            'amount': bill['total_cost'],
            'details': bill['items']
        })

# Сохранить в CSV
with open(output_csv, 'w', newline='', encoding='utf-8') as f:
    writer = csv.DictWriter(f, fieldnames=['subscriber_id', 'subscriber_name', 'phone', 'amount'])
    writer.writeheader()

    for invoice in invoices:
        writer.writerow({
            'subscriber_id': invoice['subscriber_id'],
            'subscriber_name': invoice['subscriber_name'],
            'phone': invoice['phone'],
            'amount': f"${invoice['amount']:.2f}"
        })

total_revenue = sum(inv['amount'] for inv in invoices)

return {
    'invoices_count': len(invoices),
    'total_revenue': total_revenue,
    'file': output_csv
}

# Генерация счетов за октябрь
result = generate_monthly_invoices(2024, 10)

print(f"Создано счетов: {result['invoices_count']}")
print(f"Общий доход: ${result['total_revenue']:.2f}")
print(f"Файл: {result['file']}")

```

Сценарий 2: Выявление изменений подписок

Задача: Найти всех абонентов, которые изменили подписки в октябре

Применение:

- Email рассылка о изменениях
- Анализ причин отмены
- Retention кампании

```

def find_subscription_changes(year, month):
    """Найти изменения подписок"""

    start_date = f"{year}-{month:02d}-01"
    if month == 12:
        end_date = f"{year + 1}-01-01"
    else:
        end_date = f"{year}-{month + 1:02d}-01"

    # Получить операции с подписками
    operations = get_operations(
        type=["createPackageSubscriber", "deletePackageSubscriber"],
        created_at_gte=start_date,
        created_at_lt=end_date
    )

```

```

)

# Группировка по абонентам
changes = defaultdict(lambda: {'added': [], 'removed': []})

for op in operations:
    subscriber_id = op['subscriberId']
    package_id = op['packageId']
    package_name = get_package_name(package_id)

    if op['type'] == 'createPackageSubscriber':
        changes[subscriber_id]['added'].append({
            'package': package_name,
            'date': op['createdAt']
        })
    else:
        changes[subscriber_id]['removed'].append({
            'package': package_name,
            'date': op['createdAt']
        })

# Результат
results = {
    'upgrades': [],      # Добавили пакеты
    'downgrades': [],    # Удалили пакеты
    'churned': []        # Удалили все подписки
}

for subscriber_id, activity in changes.items():
    subscriber = get_subscriber(subscriber_id)

    if activity['added'] and not activity['removed']:
        # Только добавления - апгрейд
        results['upgrades'].append({
            'subscriber': subscriber,
            'packages': activity['added']
        })
    elif activity['removed'] and not activity['added']:
        # Только удаления - возможно churn
        if len(activity['removed']) >= len(subscriber['packages']):
            results['churned'].append({
                'subscriber': subscriber,
                'packages': activity['removed']
            })
        else:
            results['downgrades'].append({
                'subscriber': subscriber,
                'packages': activity['removed']
            })
    else:
        # И добавления и удаления - изменение плана
        pass

return results

# Анализ изменений за октябрь
changes = find_subscription_changes(2024, 10)

print(f"Апгрейды: {len(changes['upgrades'])}")
print(f"Даунгрейды: {len(changes['downgrades'])}")
print(f"Полностью отменили: {len(changes['churned'])}")

# Email рассылка для churn prevention
for churned in changes['churned']:
    send_retention_email(churned['subscriber'], churned['packages'])

```

Сценарий 3: Аудит действий

Задача: Проверить, кто и когда изменял подписку абонента

```

def audit_subscriber_history(subscriber_id):
    """Полная история изменений абонента"""

    # Получить все операции
    operations = get_operations(subscriberId=subscriber_id)

    # Сортировка по времени
    operations.sort(key=lambda x: x['createdAt'])

    # Форматированный вывод
    print(f"История абонента {subscriber_id}:\n")

    for op in operations:
        timestamp = op['createdAt']
        op_type = op['type']

        if op_type == 'createSubscriber':
            print(f"{timestamp} - Создан абонент")
            print(f"    Имя: {op['payload'].get('name')}")
            print(f"    Телефон: {op['payload'].get('phone')}")

```

```

elif op_type == 'createPackageSubscriber':
    package_name = get_package_name(op['packageId'])
    print(f"{timestamp} - Подключен пакет: {package_name}")

elif op_type == 'deletePackageSubscriber':
    package_name = get_package_name(op['packageId'])
    print(f"{timestamp} - Отключен пакет: {package_name}")

elif op_type == 'disableSubscriber':
    print(f"{timestamp} - Абонент заблокирован")
    print(f" Причина: {op['payload'].get('reason', 'не указана')}")

elif op_type == 'enableSubscriber':
    print(f"{timestamp} - Абонент разблокирован")

elif op_type == 'deleteSubscriber':
    print(f"{timestamp} - Абонент удалён")

print()

# Пример
audit_subscriber_history("sKl9SW3AAAE.")

```

Сценарий 4: Reconciliation с внешним биллингом

Задача: Сверить данные Catena с внешней биллинговой системой

```

def reconcile_with_billing(year, month):
    """Сверка с внешним биллингом"""

    start_date = f"{year}-{month:02d}-01"
    if month == 12:
        end_date = f"{year + 1}-01-01"
    else:
        end_date = f"{year}-{month + 1:02d}-01"

    # 1. Получить операции из Catena
    catena_operations = get_operations(
        type=["createPackageSubscriber", "deletePackageSubscriber"],
        created_at_gte=start_date,
        created_at_lt=end_date
    )

    # 2. Получить транзакции из биллинга
    billing_transactions = get_billing_transactions(start_date, end_date)

    # 3. Сравнить
    discrepancies = []

    # Проверить, есть ли в Catena все транзакции из биллинга
    for transaction in billing_transactions:
        if transaction['type'] == 'subscription_create':
            # Найти соответствующую операцию в Catena
            found = any(
                op['subscriberId'] == transaction['subscriber_id'] and
                op['packageId'] == transaction['package_id'] and
                abs((datetime.fromisoformat(op['createdAt']).replace('Z', '+00:00')) -
                    transaction['date']).total_seconds()) < 300 # 5 минут допуск
                for op in catena_operations
                if op['type'] == 'createPackageSubscriber'
            )

            if not found:
                discrepancies.append({
                    'type': 'missing_in_catena',
                    'billing_transaction': transaction,
                    'severity': 'high'
                })

    # Проверить обратное – есть ли в биллинге все операции из Catena
    for op in catena_operations:
        if op['type'] == 'createPackageSubscriber':
            found = any(
                t['subscriber_id'] == op['subscriberId'] and
                t['package_id'] == op['packageId']
                for t in billing_transactions
                if t['type'] == 'subscription_create'
            )

            if not found:
                discrepancies.append({
                    'type': 'missing_in_billing',
                    'catena_operation': op,
                    'severity': 'medium'
                })

    return discrepancies

# Reconciliation за октябрь

```

```
discrepancies = reconcile_with_billing(2024, 10)

if discrepancies:
    print(f"⚠ Найдено {len(discrepancies)} расхождений:")
    for d in discrepancies:
        print(f"  - {d['type']}: severity={d['severity']}")
else:
    print("  Данные синхронизированы, расхождений нет")
```

5.2.8 Лучшие практики

Регулярный экспорт данных

Рекомендация: Ежедневно экспортируйте операции в свою БД для долгосрочного хранения

```
import psycopg2
from datetime import datetime, timedelta

def export_operations_to_db():
    """Экспорт операций в PostgreSQL"""

    # Подключение к БД
    conn = psycopg2.connect("postgresql://billing_db")
    cursor = conn.cursor()

    # Создать таблицу (если не существует)
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS catena_operations (
            operation_id TEXT PRIMARY KEY,
            type TEXT NOT NULL,
            portal_id TEXT,
            subscriber_id TEXT,
            package_id TEXT,
            created_at TIMESTAMP NOT NULL,
            payload JSONB
        )
    ''')

    # Получить операции за последние 24 часа
    yesterday = (datetime.now() - timedelta(days=1)).strftime('%Y-%m-%d')

    operations = get_operations(created_at_gte=yesterday)

    # Вставить в БД
    inserted = 0
    for op in operations:
        try:
            cursor.execute('''
                INSERT INTO catena_operations
                (operation_id, type, portal_id, subscriber_id, package_id, created_at, payload)
                VALUES (%s, %s, %s, %s, %s, %s, %s)
                ON CONFLICT (operation_id) DO NOTHING
            ''', (
                op['operationId'],
                op['type'],
                op['portalId'],
                op.get('subscriberId'),
                op.get('packageId'),
                op['createdAt'],
                json.dumps(op.get('payload', {}))
            ))
            inserted += 1
        except Exception as e:
            print(f"Error inserting {op['operationId']}: {e}")

    conn.commit()
    conn.close()

    return inserted

# Запускать ежедневно по cron
exported = export_operations_to_db()
print(f"Exported {exported} operations to database")
```

Использование в биллинговой системе

Архитектура:

```
Catena Operations Log ↔ Биллинговая система
    ↓                      ↓
Факт события      Финансовый расчёт
(подписка)        (счёт)
```

Workflow:

1. **Событие в Catena** — создание/удаление подписки
2. **Webhook или polling** — биллинг узнаёт о событии
3. **Запись в биллинг** — транзакция в биллинговой БД
4. **Расчёт стоимости** — на основе тарифов и периода
5. **Выставление счёта** — абоненту или партнёру

Пример webhook от Catena:

```
from flask import Flask, request
import requests

app = Flask(__name__)

@app.route('/catena-webhook', methods=['POST'])
def catena_webhook():
    """Обработка событий из Catena"""

    event = request.json

    if event['type'] == 'createPackageSubscriber':
        # Новая подписка - начать биллинг
        billing_start_subscription(
            subscriber_id=event['subscriberId'],
            package_id=event['packageId'],
            start_date=event['createdAt']
        )

        return {"status": "ok", "action": "billing_started"}

    elif event['type'] == 'deletePackageSubscriber':
        # Отмена подписки - остановить биллинг
        billing_end_subscription(
            subscriber_id=event['subscriberId'],
            package_id=event['packageId'],
            end_date=event['createdAt']
        )

        return {"status": "ok", "action": "billing_stopped"}

    return {"status": "ok", "action": "ignored"}

if __name__ == '__main__':
    app.run(port=5001)
```

Мониторинг финансовых метрик**Дашборд для руководства:**

```
def get_financial_dashboard():
    """Финансовый дашборд в реальном времени"""

    today = datetime.now().date()
    month_start = today.replace(day=1)

    # Метрики текущего месяца
    current_month = calculate_monthly_revenue(today.year, today.month)

    # Метрики прошлого месяца
    last_month_date = month_start - timedelta(days=1)
    last_month = calculate_monthly_revenue(
        last_month_date.year,
        last_month_date.month
    )

    # Сравнение
    revenue_growth = (
        (current_month['total_revenue'] - last_month['total_revenue']) /
        last_month['total_revenue'] * 100
    ) if last_month['total_revenue'] > 0 else 0

    # Операции за сегодня
    today_str = today.isoformat()
    tomorrow_str = (today + timedelta(days=1)).isoformat()

    today_creates = len(get_operations(
        type="createPackageSubscriber",
        created_at_gte=today_str,
        created_at_lt=tomorrow_str
    ))

    today_cancels = len(get_operations(
```

```

        type="deletePackageSubscriber",
        created_at_gte=today_str,
        created_at_lt=tomorrow_str
    ))

    return {
        'current_month_revenue': current_month['total_revenue'],
        'last_month_revenue': last_month['total_revenue'],
        'revenue_growth_percent': round(revenue_growth, 1),
        'today_new_subscriptions': today_creates,
        'today_cancellations': today_cancels,
        'today_net_growth': today_creates - today_cancels
    }

# Вывести дашборд
dashboard = get_financial_dashboard()

print(" Финансовый дашборд")
print(f"Доход текущего месяца: ${dashboard['current_month_revenue']:.2f}")
print(f"Доход прошлого месяца: ${dashboard['last_month_revenue']:.2f}")
print(f"Рост: {dashboard['revenue_growth_percent']:+.1f}%")
print(f"\nСегодня:")
print(f"    Новых подписок: {dashboard['today_new_subscriptions']}")
print(f"    Отмен: {dashboard['today_cancellations']}")
print(f"    Чистый пост: {dashboard['today_net_growth']:+d}")

```

5.2.9 Отчёты для бизнеса

Еженедельный отчёт для руководства

Скрипт автоматической генерации:

```

def generate_weekly_report():
    """Еженедельный отчёт для руководства"""

    # Последние 7 дней
    end_date = datetime.now()
    start_date = end_date - timedelta(days=7)

    operations = get_operations(
        type=["createPackageSubscriber", "deletePackageSubscriber",
              "createSubscriber", "deleteSubscriber"],
        created_at_gte=start_date.isoformat(),
        created_at_lt=end_date.isoformat()
    )

    # Анализ
    stats = {
        'new_subscribers': 0,
        'deleted_subscribers': 0,
        'new_subscriptions': 0,
        'cancelled_subscriptions': 0,
        'packages': defaultdict(lambda: {'adds': 0, 'removes': 0})
    }

    for op in operations:
        if op['type'] == 'createSubscriber':
            stats['new_subscribers'] += 1
        elif op['type'] == 'deleteSubscriber':
            stats['deleted_subscribers'] += 1
        elif op['type'] == 'createPackageSubscriber':
            stats['new_subscriptions'] += 1
            package_name = get_package_name(op['packageId'])
            stats['packages'][package_name]['adds'] += 1
        elif op['type'] == 'deletePackageSubscriber':
            stats['cancelled_subscriptions'] += 1
            package_name = get_package_name(op['packageId'])
            stats['packages'][package_name]['removes'] += 1

    # Формирование отчёта
    report = f"""
Еженедельный отчёт ({start_date.strftime('%Y-%m-%d')} - {end_date.strftime('%Y-%m-%d')})

Абоненты:
Новых: {stats['new_subscribers']}
Удалено: {stats['deleted_subscribers']}
Чистый пост: {stats['new_subscribers'] - stats['deleted_subscribers']:+d}

Подписки:
Активировано: {stats['new_subscriptions']}
Отменено: {stats['cancelled_subscriptions']}
Чистый пост: {stats['new_subscriptions'] - stats['cancelled_subscriptions']:+d}

По пакетам:
"""

    for package, counts in sorted(stats['packages'].items()):
        net = counts['adds'] - counts['removes']
        report += f"    {package}: {counts['adds']} активаций, {counts['removes']} отмен (net: {net:+d})\n"

```

```

    return report

# Генерация и отправка отчёта
report = generate_weekly_report()
print(report)

# Отправить на email руководству
# send_email(to="management@company.com", subject="Weekly Report", body=report)

```

5.2.10 Лучшие практики

Хранение данных

Рекомендации:

- **В Catena:** Храните операции минимум 90 дней
- **В биллинговой БД:** Храните навсегда для налоговой отчётности
- **Архивирование:** Экспортируйте старые операции в cold storage (S3, glacier)

Политика хранения:

Catena Operations API:

- Последние 90 дней: полный доступ
- 90-365 дней: архив (медленный доступ)
- >365 дней: холодное хранилище

Биллинговая БД:

- Все операции навсегда
- Индексы для быстрого поиска
- Регулярные бэкапы

Защита от дубликатов

Проблема: Повторная отправка webhook может создать дубликат операции

Решение:

```

def process_operation_idempotent(operation_data):
    """Идемпотентная обработка операции"""

    operation_id = operation_data['operationId']

    # Проверить, обрабатывали ли уже
    cursor.execute(
        "SELECT 1 FROM processed_operations WHERE operation_id = %s",
        (operation_id,)
    )

    if cursor.fetchone():
        print(f"Operation {operation_id} already processed, skipping")
        return {"status": "already-processed"}

    # Обработать операцию
    process_billing_event(operation_data)

    # Отметить как обработанную
    cursor.execute(
        "INSERT INTO processed_operations (operation_id, processed_at) VALUES (%s, NOW())",
        (operation_id,)
    )
    conn.commit()

    return {"status": "processed"}

```

Мониторинг критичных операций

Настройка алертов:

```

def monitor_critical_operations():
    """Мониторинг критичных операций для финансов"""

    # Проверить последний час
    hour_ago = (datetime.now() - timedelta(hours=1)).isoformat()

    operations = get_operations(
        type=["createPackageSubscriber", "deletePackageSubscriber"],
        created_at_gte=hour_ago
    )

```

```

)

# Аномалии
alerts = []

# 1. Слишком много отмен подписок
cancellations = [op for op in operations if op['type'] == 'deletePackageSubscriber']
if len(cancellations) > 50: # Попор
    alerts.append({
        'level': 'warning',
        'message': f'High cancellation rate: {len(cancellations)} in last hour'
    })

# 2. Нет новых подписок (странно)
creations = [op for op in operations if op['type'] == 'createPackageSubscriber']
if len(creations) == 0 and datetime.now().hour in range(10, 22):
    alerts.append({
        'level': 'info',
        'message': 'No new subscriptions in last hour (working hours)'
    })

# 3. Массовые удаления абонентов
deletions = [op for op in operations if op['type'] == 'deleteSubscriber']
if len(deletions) > 10:
    alerts.append({
        'level': 'critical',
        'message': f'Mass subscriber deletion: {len(deletions)} in last hour'
    })

# Отправить алерты
for alert in alerts:
    send_alert(alert)

return alerts

```

5.2.11 Получение конкретной операции

Запрос по ID:

```

curl -X GET https://your-catena-domain.com/tv-management/api/v1/operations/opK19SW3AAAE. \
-H "X-Auth-Token: your-api-key"

```

Ответ:

```

{
  "operationId": "opK19SW3AAAE.",
  "type": "createPackageSubscriber",
  "portalId": "pK19SW3AAAE.",
  "subscriberId": "sK19SW3AAAE.",
  "packageId": "pkK19SW3AAAE.",
  "createdAt": "2024-10-16T10:05:00Z",
  "updatedAt": "2024-10-16T10:05:00Z",
  "payload": {
    "packageName": "premium",
    "managerEmail": "admin@company.com"
  }
}

```

Применение:

- Детальная информация об операции
- Восстановление контекста события
- Отладка конкретной транзакции

5.2.12 Устранение проблем

Операции не появляются в журнале

Проблема: Действия выполняются, но не записываются в журнал

Возможные причины:

1. Проблемы с базой данных
2. Ошибки в коде при записи операций
3. Асинхронная запись с задержкой

Решение:

1. Проверьте статус базы данных
2. Просмотрите логи сервера на ошибки
3. Подождите 1-2 минуты — операции могут записываться асинхронно

Расхождения в расчётах

Проблема: Расчёты на основе операций не совпадают с биллингом

Возможные причины:

- Разные методы расчёта (calendar month vs billing cycle)
- Не учитываются refunds или корректировки
- Разные цены в разное время (изменение тарифов)
- Пропущены операции из-за сбоев

Решение:

1. Документируйте методологию расчётов
2. Регулярно сверяйте данные (reconciliation)
3. Храните историю цен пакетов
4. Логируйте все корректировки отдельно

Отсутствуют старые операции

Проблема: Операции старше определённой даты не доступны

Причина: Политика хранения данных в Catena API

Решение:

- Регулярно экспортируйте данные в свою БД
- Не полагайтесь на API как единственный источник истории
- Используйте собственное долгосрочное хранилище

5.2.13 См. также

- **Управление абонентами** — создание абонентов записывает операции
- **Управление подписками** — создание/удаление подписок
- **Управление пакетами** — операции с пакетами
- **Сеансы воспроизведения** — дополнительные данные для биллинга по трафику

6. Клиентские приложения

6.1 Руководство пользователя Android-приложения Catena

Добро пожаловать в приложение Catena для просмотра телевидения на Android!

6.1.1 Содержание

1. [Установка приложения](#)
 2. [Вход в приложение](#)
 3. [Просмотр телеканалов](#)
 4. [Просмотр видео](#)
 5. [Просмотр архива передач](#)
 6. [Полезные советы](#)
 7. [Решение проблем](#)
-

6.1.2 Установка приложения

Системные требования

- **Операционная система:** Android 5.0 (Lollipop) или выше
- **Минимум оперативной памяти:** 2 ГБ
- **Свободное место:** 50 МБ
- **Интернет-соединение:** Wi-Fi или мобильная сеть (рекомендуется от 5 Мбит/с)

Установка из Google Play Store

1. **Откройте Google Play Store** на вашем Android устройстве
2. В поиске введите "**Catena**" или "**com.flussonic.catena**"
3. Найдите приложение **Catena** от разработчика Flussonic
4. Нажмите кнопку "**Установить**"
5. Дождитесь завершения установки
6. Нажмите "**Открыть**" для запуска приложения

Прямая ссылка: [Catena в Google Play Store](#)

Установка APK файла

Если приложение недоступно в Google Play Store в вашем регионе:

1. Скачайте APK файл с официального сайта или получите у администратора сервиса
2. В настройках Android разрешите "**Установку из неизвестных источников**"
3. Путь: Настройки → Безопасность → Неизвестные источники
4. Откройте скачанный APK файл
5. Нажмите "**Установить**"
6. После установки запустите приложение

Важно: Устанавливайте APK только из проверенных источников!

6.1.3 Вход в приложение

Первый запуск

При первом запуске приложения вам необходимо войти с помощью номера телефона.

ШАГ 1: ВВЕДИТЕ НОМЕР ТЕЛЕФОНА

1. Запустите приложение Catena
2. Выберите **код страны** из выпадающего списка (например, "+7" для России)
3. Введите **номер телефона** без кода страны
4. Пример: для номера +7 912 345-67-89 введите 9123456789
5. Нажмите кнопку "**Отправить код**" (или клавишу Enter на клавиатуре)

ШАГ 2: ВВЕДИТЕ КОД ИЗ СМС

1. Вы получите СМС с **4-значным кодом**
2. Введите этот код в поле "SMS код"
3. Код проверяется автоматически при вводе 4-й цифры
4. Или нажмите кнопку "**Подтвердить**"

УСПЕШНЫЙ ВХОД

После успешного входа: - Ваш токен доступа сохранится автоматически - При следующем запуске вход произойдет автоматически - Вы попадете на главный экран со списком телеканалов

Выход из аккаунта

Чтобы выйти из приложения: 1. На главном экране нажмите кнопку "**Logout**" в правом верхнем углу 2. Вы вернетесь на экран входа

6.1.4 Просмотр телеканалов

Выбор канала

На главном экране вы увидите **список всех доступных телеканалов**.

Для каждого канала отображается: - **Название канала** - **Текущая передача** (что идет сейчас) - **Следующая передача** (что будет после)

НАЧАТЬ ПРОСМОТР

1. Найдите интересующий вас канал в списке
2. **Нажмите на канал** - откроется экран воспроизведения
3. Видео начнет воспроизводиться автоматически

Обновление списка каналов

Чтобы обновить информацию о каналах и программах: - **Потяните экран вниз** (swipe down) на главном экране - Список каналов и программы обновятся

6.1.5 Просмотр видео

Управление воспроизведением

В портретной ориентации (вертикально): - **Нажмите на экран** - появятся элементы управления воспроизведением - Доступны стандартные кнопки: - Пауза/Воспроизведение - Перемотка назад - Перемотка вперед

В альбомной ориентации (горизонтально): - Автоматически включается **полноэкранный режим** - Системные панели скрываются для максимального комфорта - Нажмите на экран, чтобы показать элементы управления - Элементы управления автоматически скроются через 3 секунды

Переключение между каналами

СПОСОБ 1: СВАЙП ВВЕРХ/ВНИЗ (В ПОРТРЕТНОМ РЕЖИМЕ)

1. **Свайп вверх** (проведите пальцем вверх по экрану) → переключение на следующий канал
2. **Свайп вниз** (проведите пальцем вниз по экрану) → переключение на предыдущий канал

При начале свайпа вы увидите: - Вверху: название предыдущего канала - Внизу: название следующего канала

СПОСОБ 2: КНОПКА "NEXT CHANNEL"

- Внизу экрана воспроизведения есть кнопка **"Switch to [название канала]"**
- Нажмите на нее для переключения на следующий канал

Поворот экрана

- **Портретный режим:** Видеоплеер занимает верхнюю часть экрана, внизу показывается программа передач
- **Альбомный режим:** Полноэкранный просмотр видео, все элементы интерфейса скрываются

6.1.6 Просмотр архива передач

Что такое архив?

Архив позволяет просматривать программы, которые уже вышли в эфир. Вы можете посмотреть передачи, которые показывали ранее.

Как открыть программу передач (EPG)

На экране воспроизведения **в портретном режиме:** - Прокрутите вниз - там отображается **список программ** для текущего канала - Список показывает программы: - Которые уже вышли (прошедшие) - Которая идет сейчас (выделена) - Которые будут показаны позже (будущие)

Просмотр программы из архива

1. На экране воспроизведения прокрутите вниз до **списка программ**
2. Найдите передачу, которую хотите посмотреть
3. **Нажмите на программу** из списка
4. Программа начнет воспроизводиться с начала

Информация о программе

Для каждой программы отображается: - **Время начала** и **время окончания** - **Название передачи** - **Описание** (если доступно) - Текущая программа **выделяется** визуально

Автоматическое переключение

- Когда заканчивается одна программа, автоматически начнется **следующая программа**
- Вы можете непрерывно смотреть канал, программы будут переключаться автоматически
- При автоматическом переключении следующая программа начинается с начала

Текущее время воспроизведения

В верхней части экрана показывается: - **Точное время воспроизведения** - какой момент записи вы сейчас смотрите - Это помогает ориентироваться при просмотре архива

6.1.7 Полезные советы

Для смартфонов

1. **Поверните телефон горизонтально** для полноэкранного просмотра
2. **Потяните вниз** на главном экране для обновления списка каналов
3. **Свайп вверх/вниз** на плеере для быстрого переключения каналов

Автоматический вход

- После первого успешного входа токен сохраняется
- При следующем запуске приложения вход происходит **автоматически**
- Вам не нужно каждый раз вводить номер телефона и код

Качество воспроизведения

- Приложение автоматически подстраивает качество под скорость вашего интернета
- Для лучшего качества используйте Wi-Fi соединение

Управление жестами

- **Одно касание экрана** - показать/скрыть элементы управления
- **Свайп вверх** - следующий канал
- **Свайп вниз** - предыдущий канал

6.1.8 Решение проблем

Не приходит СМС с кодом

1. Проверьте правильность введенного номера телефона
2. Убедитесь, что выбран правильный код страны
3. Подождите 1-2 минуты - иногда СМС приходят с задержкой
4. Попробуйте запросить код повторно

Ошибка "sms code doesn't match"

- Проверьте правильность введенного кода из СМС
- Код действителен ограниченное время - запросите новый код, если прошло много времени

Не воспроизводится видео

1. Проверьте интернет-соединение
2. Попробуйте переключиться на другой канал
3. Закройте и откройте приложение заново
4. Убедитесь, что у вас есть доступ к выбранному каналу

Список каналов пустой

1. Проверьте подключение к интернету
2. Потяните экран вниз для обновления
3. Выйдите и войдите заново
4. Обратитесь к администратору - возможно, для вашего аккаунта не настроены каналы

Видео тормозит или буферизуется

1. Проверьте скорость интернет-соединения
2. Подключитесь к Wi-Fi вместо мобильного интернета
3. Закройте другие приложения, использующие интернет
4. Попробуйте просмотр в другое время суток

6.1.9 Поддержка

Если у вас возникли проблемы с приложением, которые не описаны в этом руководстве: - Обратитесь к администратору вашего ТВ-сервиса - Укажите точное описание проблемы - Сообщите модель вашего устройства и версию Android

Приятного просмотра!