

---

# Smarty Documentation

*Выпуск Last*

Konstantin Shpinev

июн. 23, 2021



|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>1. Описание системы IPTV/OTT Middleware Smarty</b>                    | <b>3</b>  |
| 1.1      | 1.1. Архитектура системы . . . . .                                       | 3         |
| 1.2      | 1.2. Системные требования . . . . .                                      | 3         |
| <b>2</b> | <b>2. Установка и настройка сервера Smarty</b>                           | <b>5</b>  |
| 2.1      | 2.1. Где скачать установочные пакеты . . . . .                           | 5         |
| 2.2      | 2.2. Установка на ОС Linux Debian . . . . .                              | 5         |
| 2.3      | 2.3. Установка на ОС Linux CentOS . . . . .                              | 8         |
| 2.4      | 2.4. Конфигурация Smarty . . . . .                                       | 8         |
| 2.5      | 2.5. Системные команды Smarty и настройка crontab . . . . .              | 12        |
| 2.6      | 2.6. Запуск, перезапуск и остановка Smarty . . . . .                     | 18        |
| 2.7      | 2.7. Установка обновлений Smarty . . . . .                               | 19        |
| 2.8      | 2.8. Масштабирование и отказоустойчивость . . . . .                      | 20        |
| <b>3</b> | <b>3. Настройка сервиса IPTV/OTT в Smarty</b>                            | <b>21</b> |
| 3.1      | 3.1. Первичная настройка . . . . .                                       | 21        |
| 3.2      | 3.2. Руководство по работе в панели администратора . . . . .             | 26        |
| 3.3      | 3.3. Общие особенности работы с услугами и аккаунтами в Smarty . . . . . | 41        |
| <b>4</b> | <b>4. Логирование работы Smarty</b>                                      | <b>43</b> |
| 4.1      | 4.1. Логи Smarty . . . . .   | 43        |
| <b>5</b> | <b>5. Интеграция Smarty с внешними системами и сервисами</b>             | <b>55</b> |
| 5.1      | 5.1. API для разработчиков . . . . .                                     | 55        |
| 5.2      | 5.2. Интеграция с биллинговой системой . . . . .                         | 55        |
| 5.3      | 5.3. Встраивание модулей в сайт . . . . .                                | 55        |
| 5.4      | 5.4. Интеграция с популярными видео-серверами . . . . .                  | 56        |
| 5.5      | 5.5. Интеграция с онлайн-кинотеатрами . . . . .                          | 57        |
| 5.6      | 5.6. Интеграция с CAS CMS . . . . .                                      | 61        |
| 5.7      | 5.7. Интеграция с платежными системами . . . . .                         | 61        |
| 5.8      | 5.8. Дополнительные инструменты . . . . .                                | 62        |
| <b>6</b> | <b>6. Установка и настройка портала для STB и Smart TV</b>               | <b>63</b> |
| 6.1      | 6.1. Установка портала . . . . .   | 63        |
| 6.2      | 6.2. Параметры конфигурации client.js . . . . .                          | 63        |
| 6.3      | 6.3. Механизм событий . . . . .  | 64        |
| 6.4      | 6.4. Пример конфигурации . . . . .                                       | 65        |

|          |   |           |
|----------|---|-----------|
| 6.5      | 6.5 Кастомизация стилей оформления портала . . . . .  | 68        |
| <b>7</b> | <b>А. Решение проблем и рекомендации</b>  | <b>71</b> |
| 7.1      | А.1. Проблемы в работе сервера Middleware и сопутствующих системах и их решение . .                     | 71        |
| 7.2      | А.2. Рекомендации . . . . .   | 75        |
| <b>8</b> | <b>В. Дополнительные материалы</b>  | <b>77</b> |
| 8.1      | В.1. Установка драйвера cx_Oracle на Debian и пример настройки подключения Smarty<br>к Oracle . . . . . | 77        |

Содержание:



---

## 1. Описание системы IPTV/OTT Middleware Smarty

---

Актуальное описание системы в новой базе знаний по продукту на сайте: <https://microimpuls.com/docs/smarty/about>

### **1.1 1.1. Архитектура системы**

Актуальное описание архитектуры системы в новой базе знаний по продукту на сайте: <https://microimpuls.com/docs/smarty/about>

### **1.2 1.2. Системные требования**

Актуальные системные требования в новой базе знаний по продукту на сайте: <https://microimpuls.com/docs/smarty/about/system-requirements>



---

## 2. Установка и настройка сервера Smarty

---

### 2.1 2.1. Где скачать установочные пакеты

Инсталляционные пакеты ПО распространяются через FTP, доступ к которому предоставляется на время действия договора.

### 2.2 2.2. Установка на ОС Linux Debian

Все модули Smarty поставляются в виде установочных deb-пакетов и устанавливаются утилитой dpkg. Обязательным модулем является smarty-base.

Для работы необходим web-сервер nginx, uwsgi и python 2.7, а также несколько других пакетов.

#### 2.2.1 2.2.1. Установка зависимостей

Установка через apt-get:

```
sudo apt-get install nginx git python-dev python2.7 libmysqlclient-dev libtiff5-dev libjpeg62-  
↳turbo-dev zlib1g-dev  
libfreetype6-dev liblcms2-dev libwebp-dev tcl8.5-dev tk8.5-dev python-tk uwsgi uwsgi-plugin-python_␣  
↳redis-server
```

Установка утилиты pip:

```
wget https://bootstrap.pypa.io/get-pip.py && python get-pip.py && rm get-pip.py
```

#### 2.2.2 2.2.2. Установка Smarty и модулей

Установка через dpkg:

```
dpkg -i smarty*.deb
dpkg -i python2.7-jsonrpc*.deb
```

После установки пакетов Smarty необходимо установить python-библиотеки через pip:

```
sudo pip install -r /usr/share/nginx/html/microimpuls/smarty/requirements.txt
```

После установки пакета smarty-base создается конфигурационный файл для nginx в `/etc/nginx/sites-available/smarty`. По умолчанию настроен на домен `smarty.example.com` и слушает порт 80 и 8180. Необходимо перенастроить данный домен на необходимый.

Файлы Smarty размещаются в `/usr/share/nginx/html/microimpuls/smarty`.

### 2.2.3 2.2.3. Установка схемы базы данных

Первичная установка схемы базы данных осуществляется командой:

```
smarty_manage migrate --settings=settings.<settings filename>
```

- `<settings filename>` - имя файла настроек Smarty, в котором должны быть установлены параметры подключения к БД (см. *Описание основных параметров*).

---

**Примечание:** Дополнительные материалы:

- *Особенности установки и настройки подключения Smarty к СУБД Oracle*
- 

### 2.2.4 2.2.4. Создание пользователя администратора

Создание пользователя с правами служебного администратора осуществляется командой:

```
smarty_manage createsuperuser --settings=settings.<settings filename>
```

- `<settings filename>` - имя файла настроек Smarty, в котором должны быть установлены параметры подключения к БД (см. *Описание основных параметров*).

### 2.2.5 2.2.5. Создание системных объектов в базе данных и примера настроек оператора

Для создания системных объектов Smarty в базе данных, а также примера настроек выполните команду:

```
smarty_manage setup_initial_data --settings=settings.<settings filename>
```

- `<settings filename>` - имя файла настроек Smarty, в котором должны быть установлены параметры подключения к БД (см. *Описание основных параметров*).

---

**Примечание:** Можно пропустить создание образца настроек оператора, добавив флаг `--no-sample-data`

---

## 2.2.6 2.2.6. Клонирование Client со всеми настройками оператора

В случае необходимости создания нового объекта Client и копирования всех данных можно использовать команду клонирования:

```
smarty_manage clone_client --src_client_id=<source client id> --settings=settings.<settings_
↳filename>
```

- *<source client id>* - ID объекта Client, который нужно клонировать в новый Client.
- *<settings filename>* - имя файла настроек Smarty, в котором должны быть установлены параметры подключения к БД (см. *Описание основных параметров*).

**Примечание:** Можно перенести также все телеканалы, фильмы, радиостанции, камеры и сервисы, используя соответствующие опции `--clone-channels` `--clone-video` `--clone-radio` `--clone-cameras` `--clone-apps`

## 2.2.7 2.2.7. Создание пользователя или восстановление пароля

В Smarty возможно создание или восстановление пользователя через команду `create_user`:

```
smarty_manage create_user --settings=settings.<settings filename> --username=new_user --
↳password=new_password --is_admin=True --client_id=1 --is_superuser=True
```

Параметры:

- `-username` - имя пользователя, обязательный.
- `-password` - пароль, обязательный.
- `-is_admin` - *True* или *False*, если *True*, то создаваемому пользователю будет доступна служебная часть сайта, по умолчанию *False*.
- `-client_id` - ID клиента, к которому будет привязан создаваемый пользователь.
- `-is_superuser` - *True* или *False*, если *True*, то будет создан суперпользователь, по умолчанию *False*.
- `-monitoring_user` - *True* или *False*, если *True*, то пользователь будет являться оператором мониторинга устройств, по умолчанию *False*.
- `-reset_password` - если *True*, то в случае, если указанный `username` уже используется, то вместо создания нового будет изменён пароль у старого пользователя; по умолчанию *False*.

## 2.2.8 2.2.8. Создание Client

В Smarty возможно создание оператора (Client) через команду `create_client`:

```
smarty_manage create_client --settings=settings.<settings filename> --name=Client_name --api_
↳key=api_key --domain_prefix=domain_prefix --email=email@example.com
```

Параметры:

- `-name` - название оператора, обязательный.
- `-api_key` - TVMW API Key, обязательный.

- `-domain_prefix` - префикс домена оператора, обязательный.
- `-email` - email оператора, обязательный.

## 2.3 2.3. Установка на ОС Linux CentOS

Поддержка CentOS является экспериментальной - корректная работа всех функций Smarty не гарантируется. Скрипт установки для fabric и примеры конфигурации на CentOS можно найти здесь: <https://github.com/microimpuls/smarty-centos>

## 2.4 2.4. Конфигурация Smarty

### 2.4.1 2.4.1. Файл настроек Smarty

После первичной установки базовый файл конфигурации Smarty находится по адресу `/etc/microimpuls/smarty/base.py` (ссылка на `/usr/share/nginx/html/microimpuls/smarty/settings/base.py`).

Основной файл конфигурации, используемый для production-режима работы - `/etc/microimpuls/smarty/prod.py`. На этот файл (или на другой используемый конфиг) должен указывать симлинк в `/usr/share/nginx/html/microimpuls/smarty/settings/<settings name>.py`. Именно в нем следует производить настройку Smarty, т.к. базовый файл конфигурации может быть перезаписан после установки обновлений.

Конфигурация производится путем присваивания значений переменным на Python.

#### 2.4.1.1. Обслуживание нескольких экземпляров Smarty на одном сервере

Для удобства конфигурации и размещения на одном сервере нескольких экземпляров Smarty рекомендуется вместо использования файла настроек `prod.py` создать собственный файл с кратким символическим именем, совпадающим с названием сервиса, например `myiptv.py`.

Данное имя затем также рекомендуется использовать как суффикс или префикс в именах файлов конфигурации `nginx`, `uwsgi`, именах папок для логов, pid-файлов и др.

### 2.4.2 2.4.2. Описание параметров конфигурации Smarty

Актуальная документация: <https://microimpuls.com/docs/smarty/configuring-and-management/smarty-config> <https://microimpuls.com/docs/smarty/configuring-and-management/logging>

### 2.4.3 2.4.3. Добавление лицензионного ключа сервера Smarty

Каждый экземпляр Smarty привязывается к аппаратной и программной конфигурации сервера лицензионным ключом, который может быть ограничен по времени действия и максимальному числу настроенных Client ID (см. *Мультипровайдер*).

Лицензионный ключ настраивается в файле конфигурации в следующих переменных:

```
SMARTY_KEY = 'XXXXXXXX-XXXXXXXX-XXXXXXXX-XXXXXXXX'  
SMARTY_MAX_CLIENTS = 2  
SMARTY_AVAILABLE_UNTIL = 'dd.mm.yyyy'
```

Для получения ключа необходимо обратиться к своему менеджеру по договору.

#### 2.4.4 2.4.4. Настройка кеширования

Для кеширования используется сервер **Redis** - является обязательным компонентом системы. Требуется версия Redis  $\geq 2.6$ .

По умолчанию конфигурация подразумевает локальную установку сервера Redis на тот же сервер Smarty, однако при необходимости их можно разделить. Для изменения параметров подключения к Redis необходимо в конфигурации Smarty прописать массив **CACHES** следующим образом:

```
CACHES = {
    "default": {
        "BACKEND": "core.cache.backends.RedisCache",
        "LOCATION": "redis://127.0.0.1:6379/1",
        "OPTIONS": {
        }
    }
}
```

В файле конфигурации Redis `/etc/redis/redis.conf` необходимо прописать:

```
stop-writes-on-bgsave-error no
```

Для вступления изменений в силу требуется перезагрузить Redis и uwsgi.

Также поддерживается работа в кластерном режиме с группой серверов Redis, пример настройки:

```
CACHES = {
    "default": {
        "BACKEND": "core.cache.backends.RedisCache",
        "LOCATION": "redis://192.168.33.11:7000/0", # не используется, но необходимо
        "OPTIONS": {
            "REDIS_CLIENT_CLASS": "rediscluster.client.StrictRedisCluster",
            "CONNECTION_POOL_CLASS": "core.cache.cluster_connection.ClusterConnectionPool",
            "CONNECTION_POOL_KWARGS": {
                "startup_nodes": [
                    # masters
                    {"host": "192.168.33.11", "port": "7000"},
                    {"host": "192.168.33.12", "port": "7000"},
                    {"host": "192.168.33.13", "port": "7000"},
                ]
            }
        }
    }
}
```

#### 2.4.5 2.4.5. Настройка модуля геолокации

Поддерживается несколько локаторов на основе IP-адреса абонента, работающие с разными источниками гео-данных. В *служебной панели администрирования* для настраиваемого Client ID необходимо установить используемый локатор, наиболее подходящий для оператора и его региона оказания услуг.

*Если до изменения локатора база данных стран и городов уже была заполнена, то рекомендуется очистить её.*

Все локации требуют создания/обновления своей базы данных. База данных может быть в виде SQL-таблиц или бинарных данных (либо и то, и то).

### 2.4.5.1. Локатор `django-geoip` (`ipgeobase`)

Представляет собой обёртку над <https://django-geoip.readthedocs.org/en/latest/>

Команда для обновления базы:

```
$ smarty_manage geoip_update --settings=settings.<settings name>
```

Создание стран и городов на основе данных `django-geoip` (работает только если в системе нет ни одной страны и города):

```
$ smarty_manage sync_geo_geoip --settings=settings.<settings name>
```

### 2.4.5.2. Локатор `ip2location`

Обновление базы:

```
$ smarty_manage update_ip2location --settings=settings.<settings name>
```

Эта команда скачивает бинарную базу данных для определения местоположения и CSV-базу для создания справочника городов и стран.

Создание стран и городов на основе данных `ip2location` (работает только если в системе нет ни одной страны и города):

```
$ smarty_manage sync_geo_ip2location --settings=settings.<settings name>
```

После выбора локатора и синхронизации данных механизм геолокации готов к использованию. Доступность тех или иных сервисов Middleware (телеканалы, фильмы, стриминг-сервисы, опции и т.д.) определяется тарифными пакетами (см. Возможности тарификации), в настройках которых можно указать те страны и города, в которых они действуют.

## 2.4.6 2.4.6. Настройка модуля мониторинга видеопотоков

Актуальная документация: <https://microimpuls.com/docs/smarty/configuring-and-management/monitoring-and-alarms>

## 2.4.7 2.4.7. Настройка модуля статистики и отчетов

Для сохранения данных телесмотра абонентов используется сервер **MongoDB**.

---

**Примечание:** Минимальная версия MongoDB необходимая для работы - 3.4

---

Настройки задаются переменными в файле конфигурации Smarty.

`MONGODB_HOST` str Адрес сервера MongoDB.

`MONGODB_PORT` int Порт сервера MongoDB.

`MONGODB_NAME` str Название базы данных.

**MONGODB\_USERNAME** str Имя пользователя для авторизации.

**MONGODB\_PASSWORD** str Пароль для авторизации.

**MONGODB\_AUTH\_METHOD** str Метод авторизации. Метод авторизации зависит от версии MongoDB.

В секции **INSTALLED\_APPS** в файле конфигурации Smarty необходимо добавить модуль `viewstats`.

### 2.4.8 2.4.8. Настройка модуля сбора статистики по абонентам

Для отображения информации внутри вкладки «Динамика абонентов» в секции **INSTALLED\_APPS** в файле конфигурации Smarty необходимо добавить модуль `smartystats` и `viewstats`.

---

**Примечание:** После добавления модулей необходимо установить все недостающие миграции.

---

Помимо этого необходимо настроить корректную работу management-команд `cache_max_online` и `save_stats`.

### 2.4.9 2.4.9. Настройка модуля мониторинга устройств

Актуальная документация: <https://microimpuls.com/docs/smarty/configuring-and-management/monitoring-and-alarms>

### 2.4.10 2.4.10. Настройка модуля отправки SMS

SMS отправляются системой при использовании виджетов, интегрированных с сайтом, например, во время регистрации абонента. Настройки задаются переменными в файле конфигурации Smarty.

**SMS\_BACKED** str Используемый СМС-шлюз для отправки сообщений. Модуль, реализующий взаимодействие со шлюзом, должен располагаться в директории Smarty в папке `sms/backends/`.

**SMS\_ATTEMPTS** int Количество максимальных попыток отправки сообщения, после которого оно считается отправленным неуспешно.

#### 2.4.10.1. Шлюз `smsc.ru`

Значение для **SMS\_BACKEND** = `'sms.backends.smscru.SMSCBackend'`

**SMSC\_LOGIN** str Имя пользователя в сервисе `smsc.ru`

**SMSC\_PASSWORD** str Пароль в сервисе `smsc.ru`

**SMSC\_SENDER** str Имя отправителя, которое будет отображаться в SMS, отправленных через сервис `smsc.ru`

#### 2.4.10.2. Шлюз `mobipace.com`

Значение для **SMS\_BACKEND** = `'sms.backends.mobipace.MobipaceBackend'`

**MOBIPACE\_LOGIN** str Имя пользователя в сервисе `mobipace`.

**MOBIPACE\_PASSWORD** str Пароль в сервисе mobipase.

**MOBIPACE\_SENDER** str Имя отправителя для СМС-сообщений.

### 2.4.11 2.4.11. Подключение системы мониторинга ошибок Sentry

Для подключения Sentry необходимо в файле конфигурации Smarty добавить в **INSTALLED\_APPS** модуль `raven.contrib.django.raven_compat` и прописать параметры подключения:

```
RAVEN_CONFIG = {
    'dsn': 'http://<LOGIN>:<PASS>@<SENTRY_HOST>/<PROJECT>',
}
```

Строку подключения можно получить из настроек проекта в Sentry.

### 2.4.12 2.4.12. Настройка nginx и uwsgi

Образец файла конфигурации для **nginx** находится в файле `/etc/nginx/sites-available/smarty`.

Конфигурация для **uwsgi** находится в файлах `/etc/uwsgi/apps-available/smarty` и `/etc/microimpuls/smarty/uwsgi/smarty.uwsgi`, на него (или на другой используемый конфиг) должен указывать симлинк в `/usr/share/nginx/html/microimpuls/smarty/<uwsgi settings name>.uwsgi`.

### 2.4.13 2.4.13. Настройка мультиязычности контента в Smarty

Smarty позволяет сохранять в базе данных контент с названиями локализуемых полей на разных языках - например, названия телеканалов, фильмов, категорий, жанров, EPG и др. Чтобы активировать этот механизм, необходимо добавить в файл конфигурации параметр **SMARTY\_ADDITIONAL\_LANGUAGES** с перечнем необходимых языков (не более 5 дополнительных к основному языков), а также указать основной язык. Названия языков должны совпадать с названием локализации в абонентском приложении, по умолчанию используются двухбуквенные названия.

**SMARTY\_DEFAULT\_LANGUAGE** str Название основного языка. По умолчанию `ru`.

**SMARTY\_ADDITIONAL\_LANGUAGES** list Список дополнительных языков, задается в квадратных скобках с указанием значений через запятую, например: [ 'en', 'fr', 'de', 'es', 'pt' ] По умолчанию пустой.

После настройки параметров мультиязычности и перезагрузки **uwsgi** в панели администратора Smarty в полях формы локализуемых полей появится возможность указать название на дополнительных языках.

Для того, чтобы сервер Smarty в ответе на запрос API вернул значение на нужном языке, необходимо дополнительно передавать параметр `lang`. Подробнее в документации [TVMiddleware API](#).

## 2.5 2.5. Системные команды Smarty и настройка crontab

---

**Примечание:** Внимание! Некоторые команды планировщика являются обязательными для функционирования сервиса.

---

## 2.5.1 2.5.1. Кеширование списка телеканалов для абонентских устройств

Команда:

```
smarty_manage cache_channel_list --settings=settings.<settings name>
```

Рекомендуется запускать эту команду каждую минуту. При пустом кеше списка телеканалов абоненту будет выдаваться сообщение, что список телеканалов пуст.

## 2.5.2 2.5.2. Импорт EpgChannel

Команда:

```
smarty_manage epg_channel_import --settings=settings.<settings name>
```

Данная команда поможет загрузить все каналы или обновить иконки из определенного источника. Для запуска обязательно необходимо указать `epg_source_id` или `--epg_source_name`.

Если команда вызывается для загрузки иконок или загрузки всех каналов из EpgChannelSource, то к загруженным иконкам по возможности будут созданы более маленькие копии следующих размеров: 500x500, 120x91 и 40x40.

Для источника обязательно должен существовать EpgChannelSource с указанием маски URL источника каналов. Для источников, у которых нет общего списка необходимо указывать `epg_channel_id` или `channel_id`.

Аргументы для запуска:

- `--epg_source_id` Идентификатор EpgSource, для которого необходимо произвести импортирование каналов.
- `--epg_source_name` Имя EpgSource, для которого необходимо произвести импортирование каналов.
- `--epg_channel_id` Идентификатор EpgChannel, для которого необходимо произвести импортирование.
- `--channel_id` Идентификатор Channel, для которого необходимо произвести импортирование.
- `--reimport_icons` Если указан этот аргумент, то для всех импортированных каналов будет произведено обновление иконок.
- `--force_import` Загрузка всех каналов из источника. Если указан данный аргумент то все аргументы кроме `epg_source_id` и `epg_source_name` будут проигнорированы.
- `--force_parser_handling` Принудительно разрешает использовать передачу управления парсеру (равносильно `TVMW_EPG_IMPORT_ALLOW_PARSER_HANDLING=True`)
- `--force_disable_parser_handling` Принудительно запрещает использовать передачу управления парсеру (равносильно `TVMW_EPG_IMPORT_ALLOW_PARSER_HANDLING=False`)

Использование опций `--force_parser_handling` и `--force_disable_parser_handling` приоритетнее параметра `TVMW_EPG_IMPORT_ALLOW_PARSER_HANDLING`.

Пример команды для повторного импортирования иконок для одного канала:

```
smarty_manage epg_channel_import -epg_source_id=1 -epg_channel_id=100  
-reimport_icons -settings=settings.<settings name>
```

### 2.5.3 2.5.3. Импорт EPG

Команда:

```
smarty_manage epg_import --settings=settings.<settings name>
```

Рекомендуется запускать импорт несколько раз в день для поддержания актуальности телепрограммы (см. *Настройка EPG и телеканалов*). Если не произвести импорт EPG, то программа телепередач на устройстве абонента будет пустой.

Для того чтобы произвести импорт EPG для одного определенного источника необходимо передать параметр `--epg_source_id` или `--epg_source_name`.

В данную команду можно также передать параметр `--epg_channel_id` для импорта EPG только для определенного EpgChannel.

Также с помощью параметра `--actual_days` можно указать количество дней, когда загруженные EPG считаются актуальными, то есть все EPG, старше чем указанное количество дней, будут удалены.

Также эта команда позволяет генерировать на основе загруженного EPG контент и подборки, для этого необходимо передать параметр `--create_content`. Если необходимо только сгенерировать контент и подборки, то помимо предыдущего параметра нужно передать параметр `--ignore_epg_import`. Для данной опции работает фильтрация по EpgChannel и по источнику, однако генерироваться контент будет `__` для всех встречаемых **EPG \_\_**, а не только загруженных.

### 2.5.4 2.5.4. Импорт EPG-премьер

Команда:

```
smarty_manage epg_premiere_import --settings=settings.<settings name>
```

Рекомендуется запускать 1-2 раза в день для поддержки актуальности списка премьер (см. *Настройка EPG и телеканалов*). Если не произвести импорт, то список премьер на устройстве абонента будет пустой.

В данную команду можно также передать параметр `--epg_channel_id` для импорта премьер только для определенного EpgChannel.

### 2.5.5 2.5.5. Команда списания/продления аккаунтов с помощью встроенного биллинга согласно расчетным периодам

Команда:

```
smarty_manage check_accounts --settings=settings.<settings name>
```

Команда осуществляет деактивацию аккаунтов, для которых подошел к концу расчетный период, а также производит списание средств и продление действующих аккаунтов. Рекомендуется запускать каждую ночь (см. Описание встроенного биллинга).

## 2.5.6 2.5.6. Опрос анализаторов TS-потоков MicroTS (модуль мониторинга видеопотоков)

Актуальная документация: [https://microimpuls.com/docs/smarty/configuring-and-management/management-commands#check\\_streams](https://microimpuls.com/docs/smarty/configuring-and-management/management-commands#check_streams)

## 2.5.7 2.5.7. Рассылка информационных сообщений на экраны устройств и email о приближении срока деактивации/необходимости оплаты

Команда:

```
smarty_manage send_activation_expires_messages --days_count <количество оставшихся дней> --  
↪ settings=settings.<settings name>
```

Рекомендуется запускать каждый вечер.

## 2.5.8 2.5.8. Очистка старых недоставленных информационных сообщений

Команда:

```
smarty_manage clean_old_messages --days_count 3 --settings=settings.<settings name>
```

## 2.5.9 2.5.9. Повторная отправка SMS-сообщений, недоставленных с первого раза

Команда:

```
smarty_manage resend_sms --settings=settings.<settings name>
```

Рекомендуется запускать каждые 1-3 минуты.

## 2.5.10 2.5.10. Команда удаления всех сессий авторизации всех аккаунтов для заданного Client ID

Команда:

```
smarty_manage delete_authkeys --client_id=<client_id> --settings=settings.<settings name>
```

Внимание, выполнение команды приведет к логoutu всех устройств.

## 2.5.11 2.5.11. Команда проверки доступности стриминг-сервисов для механизма отказоустойчивости

Команда:

```
python manage.py check_stream_services --settings=settings.<settings name>
```

При настройке отказоустойчивой схемы сервиса с балансировкой нагрузки рекомендуется выполнять эту команду каждую минуту.

Команда проверяет сервисы по тем методам проверки, которые настроены в свойствах стриминг-сервиса. .. `_make_autopayments`:

### 2.5.12 2.5.12. Команда совершения автоплатежей

Команда:

```
smarty_manage make_autopayments --settings=settings.<settings name>
```

Производит оплату для тех клиентов, у которых активен автоплатёж, будет списание средств при проверке аккаунтов сегодня и количество средств недостаточно для проделения всех аккаунтов клиента. Рекомендуется выполнять непосредственно перед вызовом `check_accounts`.

При указании параметра `--use_pool` обработка клиентов будет производиться в многопоточном режиме.

### 2.5.13 2.5.13. Команда миграции клиента на подписочную модель

Команда:

```
smarty_manage migrate_to_nb --client_id=<client_id> --settings=settings.<settings name>
```

Мигрирует клиента на подписочную модель тарифов. Создаёт для каждого пользователя и аккаунта подписки на каждый подключенный тариф.

**Изменения, сделанные данной командой являются необратимыми, рекомендуется сделать резервную копию БД перед применением. Ни в коем случае не использовать без необходимости.**

### 2.5.14 2.5.14. Команда кэширования существующих иконок

Команда:

```
smarty_manage recache_icons --settings=settings.<settings name>
```

Вызывается в случае отсутствия информации о существующих иконках.

Команда проверяет и сохраняет в кэше существование иконок для всех EpgChannel по размерам, указанным в `SMARTY_DEFAULT_ICON_SIZE` и `SMARTY_DEFAULT_ICON_SIZES`.

### 2.5.15 2.5.15. Очистка старых отчетов

Команда:

```
python /usr/share/nginx/html/microimpuls/smarty/manage.py delete_old_reports --save-days=30 --  
↪ settings=settings.<settings name>
```

В данную команду необходимо передать параметр `--save-days` для указания количества дней, за которое отчеты нужно сохранить.

## 2.5.16 2.5.16. Очистка лога действий абонента

Команда:

```
python /usr/share/nginx/html/microimpuls/smarty/manage.py clear_customer_log --days=60 --
↳ settings=settings.<settings name>
```

Параметр `--days` обязателен и определяет, что при вызове команды будут удалены записи старше `days` дней.

## 2.5.17 2.5.17. Сохранение максимального числа абонентов онлайн для статистики

Команда:

```
smarty_manage cache_max_online --settings=settings.<settings name>
```

Можно запускать с любой регулярностью, но не реже раз в сутки.

## 2.5.18 2.5.18. Сбор и хранение статистики по каждому клиенту

Команда:

```
smarty_manage save_stats --settings=settings.<settings name>
```

Необходимо запускать раз в сутки.

## 2.5.19 2.5.19. Пример настройки crontab

Пример:

```
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
*/1 * * * * python /usr/share/nginx/html/microimpuls/smarty/manage.py cache_channel_list --
↳ settings=settings.prod
0 5,9,13 * * * python /usr/share/nginx/html/microimpuls/smarty/manage.py epg_import --
↳ settings=settings.prod
0 3 * * * python /usr/share/nginx/html/microimpuls/smarty/manage.py clean_old_messages --
↳ days_count 3 --settings=settings.prod
```

## 2.5.20 2.5.20. Команда генерации видео для архивных записей

Команда:

```
smarty_manage make_vodpvr --client_id=<client_id> --settings=settings.<settings name>
```

Выполняет создание видео для каждой программы, если для нее существует канал ведущий архивную запись. Отличается от контента

### 2.5.21 2.5.21. Команда сбора дополнительной статистики для отчета по количеству пользователей

Команда:

```
smarty_manage collect_daily_stats --settings=settings.<settings name>
```

Необходимо выполнять каждый день в начале дня (время записи сохраняется в UTC+0).

### 2.5.22 2.5.22. Команда валидации платежей

Команда:

```
smarty_manage check_payment_status --settings=settings.<settings name>
```

Параметры:

`--timeout_minutes` – количество минут по прошествии которых платёж считается «устаревшим» и более не проверяется.

Необходимо для платёжных систем, у которых отсутствует автоматическая нотификация. На данный момент поддерживаются: Fortebank и Payture.

### 2.5.23 2.5.23. Команда кеширования данных для графиков «Динамики абонентов»

Команда:

```
smarty_manage cache_customers_dynamic --settings=settings.<settings name>
```

Параметры:

`--client_id` – список идентификаторов клиентов, через запятую, для которых нужно провести кеширование. Если в аргумент ничего не передано, то кешируются данные для всех клиентов.

- Команду рекомендуется запускать раз в час.
- Для небольших баз графики будут работать быстро и корректно и без данной команды.
- При первом запуске команда может выполняться долго (зависит от количества аккаунтов, транзакций и тарифов) из-за того, что будут рассчитываться данные за весь прошлый месяц. При повторном запуске будут рассчитываться данные только за последний день или час (уже кешированные данные повторно не кешируются).
- Поскольку команда может выполняться долго рекомендуется сначала попробовать для одного клиента (аргумент `-client_id`).

## 2.6 2.6. Запуск, перезапуск и остановка Smarty

Для управления процессами сервера приложений uwsgi используется init-скрипт `/etc/init.d/uwsgi`:

```
$ /etc/init.d/uwsgi
Usage: /etc/init.d/uwsgi {start|stop|status|restart|reload|force-reload}
```

Все команды действуют на все запущенные экземпляры uwsgi.

Логи по умолчанию сохраняются в `/var/log/uwsgi/`, `/var/log/nginx/` и `/var/log/microimpuls/`.

## 2.7 2.7. Установка обновлений Smarty

**Примечание:** Перед установкой пакетов обновления, пожалуйста, сделайте резервную копию конфигурации, файлов Smarty, а также дампа базы данных.

Обновления устанавливаются командой `dpkg`:

```
dpkg -i smarty*.deb
```

После установки обновления необходимо установить новые требуемые python-библиотеки через `pip`:

```
sudo pip install -r /usr/share/nginx/html/microimpuls/smarty/requirements.txt --ignore-installed
```

Миграция схемы БД осуществляется командой:

```
python /usr/share/nginx/html/microimpuls/smarty/manage.py migrate --settings=settings.<settings_↵
↳filename>
```

После установки всех обновлений и миграции схем БД необходимо перезапустить сервер приложений uwsgi, завершить все команды `epg_import` и `cache_channel_list` (через вызов `kill`), а затем выполнить команду обновления кеша:

```
python /usr/share/nginx/html/microimpuls/smarty/manage.py flushall --settings=settings.<settings_↵
↳filename>
```

**Примечание:** Если не выполнить команду обновления кеша `flushall`, то в кеше могут оказаться данные со старой структурой, что может привести к непредсказуемым ошибкам в работе приложений.

### 2.7.1 2.7.1. Устранение ошибки конфликта миграций

В процессе миграции схемы БД может возникнуть ошибка конфликта миграций: `To fix them run 'python manage.py makemigrations --merge'`. Не нужно делать команду `makemigrations`. Ошибка может возникнуть в случае нарушения правильного порядка действий при установке или обновлении системы. Чтобы устранить эту ошибку, необходимо выполнить следующие действия:

- Удалить содержимое папки `/tvmiddleware/migrations`
- Переустановить пакет обновления `smarty`, который был установлен в тот момент, когда возникла данная ошибка. Временно удалить из папки `/tvmiddleware/migrations` все новые миграции (дата создания у которых новее чем дата последнего успешного обновления `smarty`).
- Очистить таблицу `django_migrations` в базе данных `smarty`.
- Выполнить команду:

```
$ python manage.py migrate --fake --settings=settings.<settings filename>
```

- Скопировать обратно миграции, временно удаленные на шаге 2. Повторно выполнить команду миграции:

```
$ python manage.py migrate --settings=settings.<settings filename>
```

## 2.8 2.8. Масштабирование и отказоустойчивость

### 2.8.1 2.8.1. Доступные средства масштабирования и отказоустойчивости

Подробная информация о кластерных конфигурациях Smarty в новой базе знаний по продукту на сайте <https://microimpuls.com/docs/smarty/scaling-and-redundancy>.

### 2.8.2 2.8.2. Настройка подключения к СУБД с использованием репликации

Актуальная документация по настройке подключения к SQL-кластеру: <https://microimpuls.com/docs/smarty/scaling-and-redundancy/smarty-sql-cluster-connection>

### 2.8.3 2.8.3. Настройка логирования статистики запросов в statsd

statsd - сервер агрегации статистических данных: <https://github.com/etsy/statsd>.

Smarty позволяет выгружать в statsd статистику по запросам к API (количество запросов, время ответа, количество выполненных SQL-запросов, время ответа СУБД). Для этого необходимо добавить в файл конфигурации Smarty параметры, указанные ниже:

```
MIDDLEWARE_CLASSES += (
    'core.middleware.StatsMiddleware',
)

STATSD_HOST = 'X.X.X.X'
STATSD_PORT = '8125'
```

Где:

**STATSD\_HOST** str IP-адрес сервера statsd для выгрузки данных статистики и мониторинга работы сервера Smarty.

**STATSD\_PORT** int Порт сервера statsd для выгрузки данных статистики и мониторинга работы сервера Smarty.

**STATSD\_PREFIX** str Префикс, который будет добавляться (если задан) к ключам параметров, передаваемых в statsd.

---

**Примечание:** Внимание! Необходимо обеспечить доступность сервера statsd и правильность настроек подключения, в противном случае подключенная `core.middleware.StatsMiddleware`` и отсутствие соединения со statsd может приводить к чрезмерному потреблению оперативной памяти.

---

---

## 3. Настройка сервиса IPTV/OTT в Smarty

---

Для настройки сервиса доступны две панели администрирования - служебная и основная. Служебная панель необходима для настройки общих и базовых сущностей, а также редактирования любых объектов.

В основной панели настраивается непосредственно сам сервис IPTV/OTT для конкретного оператора (Client ID).

|           |   |
|-----------|---|
| Служебная | <a href="http://smarty.example.com/admin">http://smarty.example.com/admin</a> |
| Основная  | <a href="http://smarty.example.com">http://smarty.example.com</a>             |

Служебная панель администрирования доступна только для супер-администратора (общий администратор системы).

---

**Примечание:** Домен example.com приведен для примера, используйте свой домен, настроенный в файле конфигурации nginx.

---

### 3.1 3.1. Первичная настройка

#### 3.1.1 3.1.1. Добавление Client ID и возможность работы в режиме «Мультипровайдер»

---

**Примечание:** Можно пропустить этот шаг, если была выполнена команда `setup_initial_data`

---

Мультипровайдер - это возможность подключения нескольких проектов или операторов в рамках одной инсталляции системы. Для каждого проекта при этом будет использоваться независимый набор настроек, абонентская база, параметры устройств, услуг и т.д.

Для функционирования системы необходимо, чтобы был создан хотя бы один Client ID. Под Client ID подразумевается оператор услуг или проект оператора.

Создать Client ID необходимо в служебной панели администрирования по ссылке: <http://smarty.example.com/admin/clients/client/>

Описание полей:

**Название** Название оператора или проекта.

**E-Mail** E-mail для отправки системных сообщений.

**Поддомен** Имя поддомена для настройки порталов, например provider.example.com.

**Адрес сайта** Адрес сайта сервиса.

**API key** Ключ API для подключения абонентских устройств.

**Billing API key** Ключ API для скрипта интеграции с внешним биллингом.

**Валюта по умолчанию** Валюта, которая будет выбрана при оплате услуг абонентом.

**Способ оплаты по умолчанию** Способ оплаты, который будет выбран при оплате услуг абонентом.

**Максимально дней пробного просмотра** Ограничение для запросов к API на создание аккаунтов.

**Маска номера договора** Маска формирования номера договора (см. Документы).

**Механизм GeoIP** Используемый механизм геолокации (см. *Настройка гео-локации*).

#### 3.1.1.1. Привязка абонентского устройства к Client ID

Для привязки приложения абонента к конкретному Client ID используется ключ API (`api_key`) и `client_id`. Эти параметры необходимо прописать в файле настроек `client.js` при настройке портала, а также при сборке нативных клиентов под устройства.

API-ключ рекомендуется генерировать через утилиту `pwgen` с ключом `-s` длиной не менее 64 символов, например:

```
pwgen -s 64
```

#### 3.1.2 3.1.2. Добавление шаблонов порталов

---

**Примечание:** Можно пропустить этот шаг, если была выполнена команда `setup_initial_data`

---

В служебной панели администрирования добавить установленные шаблоны порталов: <http://smarty.example.com/admin/tvmiddleware/playdevicetemplate/>

Названия (пути) стандартных шаблонов: `classic impuls iridium focus futuristic`

#### 3.1.3 3.1.3. Добавление поддерживаемых устройств

---

**Примечание:** Можно пропустить этот шаг, если была выполнена команда `setup_initial_data`

---

В служебной панели администрирования добавить поддерживаемые типы устройств: <http://smarty.example.com/admin/tvmiddleware/playdevice/>

Описание полей:

**Название** Название типа устройства.

**Системное название** Системное название типа устройства, возможные значения см. ниже.

Поддерживаемые типы устройств (системные названия):

`android` - под данным типом устройства распознается мобильный клиент под Android.

`android_stb` - Smart TV или STB под OS Android.

`dune` - STB Dune HD, начиная с модели Dune HD TV-101 и новее (модели PRO 4K, NEO 4K PLUS, NEO 4K T2 PLUS, SKY 4K PLUS работают под OS Android и относятся к типу устройства `android_stb`).

`eltex` - STB Eltex под OS Linux (NV-102).

`tvip` - STB Tvip.

`lg_netcast` - LG Smart TV под OS Netcast.

`lg_webos` - LG Smart TV под OS WebosTV.

`mag` - STB MAG.

`pc` - под данным типом устройства распознается ПК-клиент (не поддерживается на текущий момент).

`sagemcom` - STB Sagemcom.

`samsung_smart_tv` - Samsung Smart TV под Orsay (серии E, F, H).

`tizen_tv` - Samsung Smart TV под Tizen (серии J, K, M, Q, N, R).

`ios` - под данным типом устройства распознается мобильный клиент под iOS.

`wrt` - STB WRTech.

`amino` - STB Amino.

`imaqliq` - STB Imaqliq под OS Linux (G-box, Q-box). Модели под OS Android (Q-box Ultra) относятся к типу устройства `android_stb`

`kodi` - медиаплеер Kodi.

`vewd` - Smart TV, запускающие видео через HTML5-плеер (бывш. OperaTV). К таким телевизорам могут относиться устройства на платформах: Zeasn (Philips Smart TV), AmazonFireOS, Vewd и другие.

`tbrowser` - телевизоры TCL и те, которые используют внутренний движок `tbrowser` (Panasonic FSR400).

### 3.1.4 3.1.4. Подключение разрешенных типов устройств для Client ID

---

**Примечание:** Можно пропустить этот шаг, если была выполнена команда `setup_initial_data`

---

В служебной панели администрирования добавить разрешенные типы устройств для каждого Client ID: <http://smarty.example.com/admin/tvmiddleware/clientplaydevice/>

### 3.1.5 3.1.5. Настройка EPG и иконок телеканалов

В системе существует базовое понятие EPG Channel - это телеканал с прикрепленными иконками и программой передач. При создании сетки каналов оператора каждому каналу ставится в соответствие один из базовых каналов. Таким образом, за телеканалами оператора закрепляется иконка и телепрограмма (EPG).

Телепрограмма может быть получена из разных источников, которые настраиваются в служебной панели администрирования: <http://smarty.example.com/admin/tvmiddleware/epgservice/>

Описание полей:

**Название источника** Название для отображения.

**Имя модуля парсера** Имя должно соответствовать имени файла с классом парсера в папке /tvmiddleware/epg\_parsers/.

**Маска URL** Предоставляется поставщиком EPG.

Существующие парсеры:

| Имя модуля       | Поставщик EPG  |
|------------------|--|
| dummy_source     | Специальный источник EPG, который генерирует 60-минутные отбивки. Может использоваться, например, для каналов без EPG или для видеокамер с PVR. Маска URL: оставить пустым   |
| yandex           | <a href="http://tv.yandex.ru">http://tv.yandex.ru</a> , бесплатный доступ (парсер с сайта).<br>Маска URL: оставить пустым  |
| teleguide        | <a href="http://teleguide.info">http://teleguide.info</a> , бесплатный доступ (парсер с сайта).<br>Маска URL: оставить пустым  |
| epgservice       | <a href="http://epgservice.ru">http://epgservice.ru</a> , платный доступ, формат XMLTV.<br>Маска URL:<br><a href="http://xmldata.epgservice.ru:8181/EPGService/hs/xmldata/&lt;id&gt;/file/%s&lt;id&gt;">http://xmldata.epgservice.ru:8181/EPGService/hs/xmldata/&lt;id&gt;/file/%s&lt;id&gt;</a> - идентификатор сервиса, предоставляется epgservice |
| xmltv_common     | Универсальный парсер XMLTV.<br>Маска URL: указать на источник XMLTV  |
| xmltv_from_files | Парсер XMLTV-файлов, основан на xmltv_common.<br>Маска URL: указать путь до файла на сервере Smarty  |
| walla            | <a href="http://walla.co.il">http://walla.co.il</a> , бесплатный доступ (парсер с сайта).<br>Маска URL: оставить пустым  |
| ucom             | <a href="https://www.ucom.am/en/personal/home-services/u-tv/epg">https://www.ucom.am/en/personal/home-services/u-tv/epg</a> (парсер с сайта)<br>Маска URL: оставить пустым   |

Настройка EPG-каналов осуществляется в служебной панели администрирования: <http://smarty.example.com/admin/tvmiddleware/epgchannel/>

Описание полей:

**Название** Название канала.

**URL иконки** Путь к иконке, абсолютный или относительный, начиная с /tvmiddleware/media/.

**Источник EPG** Имя источника.

**ID канала в источнике EPG** ID канала в сервисе источника.

**Номер для сортировки** Позиция в общем списке, используется для автоматической сортировки оператора.

**Сдвиг в часах** Сдвиг программы в часах относительно UTC+0.

Иконки каналов по умолчанию находятся по адресу /tvmiddleware/media/img/logo/default/.

### Использование иконок нескольких размеров

Если приложение требует иконки с определенными размерами, то сервер будет выдавать иконки с адресом <имя файла><ширина>\_<высота>.<расширение>. Например, если иконка стандартного размера

располагается по адресу `/tvmiddleware/media/img/logo/default/somelogo.png`, то иконка размера 400x400px - `/tvmiddleware/media/img/logo/default/somelogo400_400.png`.

Требуемые размеры иконок передаются приложением как аргументы `icon_width`, `icon_height` в запросах TVMiddleware API.

---

**Примечание:** Сервер не проверяет существование файла с иконкой, указание неправильных размеров приведет к выдаче URL на несуществующую иконку.

---

### 3.1.5.1. Добавление нового типа парсера

Для добавления собственного парсера EPG необходимо создать модуль на Python в папке `/tvmiddleware/epg_parsers/`, который должен содержать класс `EpgParser`, наследуемый от `EpgParserBase` и реализующий все его методы, а затем создать запись в EPG Source.

### 3.1.5.2. Редактирование EPG в ручном режиме

Редактирование EPG доступно в служебной панели администрирования по адресу: <http://smarty.example.com/admin/tvmiddleware/epg/>

### 3.1.5.3. Добавление EPG-категорий и EPG-жанров

---

**Примечание:** Можно пропустить этот шаг, если была выполнена команда `setup_initial_data`

---

Для возможности более детальной и удобной фильтрации контента введены понятия EPG-категорий и EPG-жанров - данные метрики предоставляются поставщиком EPG в составе описания каждой конкретной программы. Таким образом, помимо категории телеканала, пользователю также доступны категория и жанр любой передачи в отдельности, которые могут не совпадать с тематикой самого канала.

---

**Примечание:** Именно на основе EPG-категорий и EPG-жанров работает фильтрация передач в экране «ТВ по интересам».

---

Добавление EPG-категорий и EPG-жанров осуществляется аналогично, поэтому ниже будет представлено описание этого процесса для категорий.

1. В первую очередь создаются категории, которые в дальнейшем будут отображаться для абонентов в приложении: <http://smarty.example.com/admin/tvmiddleware/epgcategory/>

Описание полей:

**Category name** Название категории.

2. Далее создается «карта отображения» созданных категорий, на те, что предоставляет источник EPG (список данных категорий запрашивается у поставщика EPG): <http://smarty.example.com/admin/tvmiddleware/epgsourcecategorymap/>

Описание полей:

**Источник EPG** Имя источника.

**Название категории у источника** Имя категории в том виде, в котором его отдаёт источник EPG.

**Категория EPG** Имя одного из объектов `epgcategory`, заранее созданных в панели администратора на шаге 1, либо созданных в процессе.

Шаг 2 необходимо проделать для всех названий категорий, отдаваемых источником.

#### 3.1.5.4. Добавление EPG-премьер

Помимо программы передач можно получить также программу премьер для указанного источника EPG. Программа премьер может быть получена из разных источников, которые настраиваются в служебной панели администрирования: <http://smarty.example.com/admin/tvmiddleware/epgpremieresource/>

Описание полей:

**Epg source** Источник, для каналов которого будут загружаться премьеры.

**Адрес URL источника** Предоставляется поставщиком премьер.

## 3.2 3.2. Руководство по работе в панели администратора

### 3.2.1 3.2.1. Общие сведения об административном интерфейсе

Условно интерфейс можно разделить на две области: панель управления и область данных.

Панель управления имеет следующие элементы:

- Ссылки на разделы настроек — обеспечивает удобную навигацию по интерфейсу.
- Выбор текущего оператора в рамках функции *Мультипровайдер*.
- Выбор языка — кнопки переключения языка интерфейса (русский и английский).
- Имя пользователя — показывает имя текущего пользователя, а так же позволяет выйти из административного интерфейса, если при нажатии на имя пользователя в открывшемся списке выбрать «Выход».

Область данных может выглядеть по-разному в зависимости от текущего раздела.

### 3.2.2 3.2.2. Описание интерфейса

Все настройки административного интерфейса тематически сгруппированы в меню на панели управления. При выборе любого пункта выводится список настраиваемых сущностей. Если в списке нет ни одного пункта, то вместо списка выводится сообщение о том, что они не найдены.

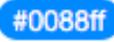
Для списков доступна сортировка, но только по одному столбцу. При этом доступные для сортировки столбцы имеют нижнее точечное подчеркивание своего наименования.

| <u>ID</u> | <u>Название</u> | <u>Категория</u> |
|-----------|-----------------|------------------|
|-----------|-----------------|------------------|

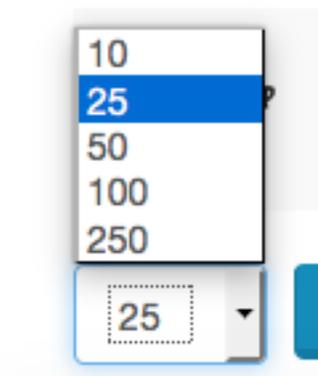
Чтобы отсортировать список нужно просто нажать на название столбца. Первый клик отсортирует список по возрастанию, второй — по убыванию, дальнейшие клики будут чередовать эти два способа сортировки. При этом сортировка по возрастанию обозначается стрелкой вверх рядом с наименованием столбца, а сортировка по убыванию — стрелкой вниз.

## Название

Для некоторых данных используется специальная колонка *Порядок сортировки*. Она сортирует элементы не только в административном интерфейсе, но и определяет порядок отображения элементов в интерфейсе на устройстве абонентов. В этой колонке каждому элементу списка соответствует свой значок стрелки. В зависимости от того, вверх или вниз направлена стрелка, при нажатии на нее элемент уйдет вверх или вниз по списку соответственно.

|                          | ID | Цвет  | Название                         | Порядок сортировки  |
|--------------------------|----|---|----------------------------------|--|
| <input type="checkbox"/> | 24 |  | <a href="#">Основные</a>         |                     |
| <input type="checkbox"/> | 59 |  | <a href="#">Фильмы и сериалы</a> |                     |
| <input type="checkbox"/> | 69 |  | <a href="#">Познавательные</a>   |                     |
| <input type="checkbox"/> | 60 |  | <a href="#">Спорт</a>            |                     |

Если список элементов большой, то он разбивается на страницы. На одной странице обычно размещается 25 записей, но можно выбрать другое значение — 10, 50, 100 или 250, за эту функцию отвечает раскрывающийся список внизу страницы.



При выборе нового значения текущая страница обновляется, и в зависимости от получившегося количества страниц, отображается либо та же по счету страница, на которой была произведена смена значений, либо первая ближайшая к ней. Навигация между страницами осуществляется с помощью навигационной панели с номерами страниц. На панели располагается 10 кнопок с номерами страниц, остальные кнопки позволяют перемещаться между страницами. Так кнопки < и > ведут на предыдущую и следующую страницы соответственно. А кнопки << и >> загружают первую и последнюю страницы соответственно.



Почти во всех разделах доступен поиск или фильтрация данных.



Для возврата от результатов поиска к полному списку служит кнопка **Сбросить**.

Практически для всех настроек доступно добавление/удаление пунктов. Эту функцию обеспечивают кнопки **Добавить**, **Изменить** и **Удалить выбранные** над списком.



При этом кнопки **Изменить** и **Удалить выбранные** становятся активными, только после выбора хотя бы одного пункта списка.

Для удаления сущности достаточно нажать на кнопку **Удалить выбранные**. После нажатия кнопки **Изменить** открывается страница редактирования, где можно менять значения параметров.

## Редактирование дата-центра: mskix

Название:

mskix

Расположение:

Moscow MMTS-9

Включено

**СОХРАНИТЬ ИЗМЕНЕНИЯ**

Кнопка **Сохранить изменения** сохраняет внесенные правки. Кнопка **Вернуться к списку** не сохраняя внесенных правок, просто перемещает пользователя к списку настраиваемых сущностей.

В некоторых разделах доступна сводная статистика активности, например в аккаунтах абонентов.



Синим цветом в таких таблицах обозначается общее количество записей. Зеленым обозначается количество записей, у которых в настройках выбрано: *Активен* или *Включен*, либо их статус *Online*, соответственно красный цвет — количество записей, у которых не включены значения *Активен* или *Включен*. Серый цвет — количество записей со статусом *Offline*.

### 3.2.3 3.2.3. Обзор основных разделов

Панель администратора позволяет управлять настройками таких компонентов как:

- Абонентская база

- Тарифные пакеты и набор услуг
- Телеканалы
- Телепрограмма (EPG)
- Радиостанции
- Каталог видеотеки
- Каталог приложений
- Стриминг-сервисы (Live, VOD, PVR и др.)
- Устройства

Для удобства настройки сгруппированы в меню на главной панели и разделены на категории:

- Общие настройки
- Видео-серверы
- Мониторинг
- Настройки стриминга
- Биллинг
- Настройки контента
- Абоненты
- Отчеты

Чтобы начать работать с настройками следует выбрать необходимый пункт в выпадающем списке интересующей категории. Каждая настройка представляет собой список, элементы которого можно добавлять/удалять, а так же менять значения их параметров, что позволяет настраивать различные компоненты.

### 3.2.4 3.2.4. Раздел: Общие настройки

#### 3.2.4.1. Настройки STB и приложений

Этот раздел содержит список устройств для просмотра сервиса IPTV (приставки Set-Top Box, Smart TV, мобильные устройства, компьютер и др.), которые поддерживаются оператором (см. *Подключение разрешенных оператору типов устройств*).

Здесь указываются базовые настройки для взаимодействия устройств с сервисом.

Для редактирования настроек устройства можно использовать кнопку **Настройки**, либо нажать на название устройства.

##### 3.2.4.1.1 Рекомендации по заполнению блока «Логотип оператора»

**Логотип для главного меню** - используется во всех шаблонах, кроме *infinitely* и мобильного приложения, рекомендуемые размеры:

- *futuristic* (STB&TV) - 178x48 пикселей
- *impuls* (STB&TV) - 311x123 пикселя
- *classic* (STB&TV) - 206x74 пикселя

**Логотип для страницы авторизации** - используется только в шаблоне `classic`, рекомендуемый размер: 140x174 пикселя

**Логотип для стартовой страницы** - используется только в шаблоне `classic`, рекомендуемый размер: 178x140 пикселей

### 3.2.4.2 Локализация уведомлений

Данный раздел позволяет локализовать имена транзакций, уведомления биллинга, отображаемые для абонента, а также имена ценовых категорий для контента.

Локализация уведомлений доступна для всех настроенных языков Smarty.

### 3.2.4.3 Правовые документы

Данный раздел позволяет создавать/настраивать/удалять правовые документы для абонентов, такие как: оферта, политика конфиденциальности, правила пользования сервисом и другие.

Отображение и акцепт правовых документов доступен не во всех шаблонах клиентских приложений.

### 3.2.4.4. Настройка прав пользователей

В этом разделе администратору доступно редактирование прав других администраторов или модераторов сервиса для ограничения их доступа к тем или иным разделам или функциональности.

Добавление новых пользователей производится в служебной панели администрирования по ссылке: <http://smarty.example.com/admin/users/user/>.

Права доступа разделены по группам согласно категориям разделов в панели администратора. Детальные права на выполнение тех или иных действий с данными состоят из:

- *Can view ...* - имеет доступ к просмотру информации
- *Can create ...* - имеет доступ к созданию элементов
- *Can edit ...* - имеет доступ к редактированию
- *Can delete ...* - имеет доступ к удалению

### 3.2.4.5 Провайдеры HbbTV

Раздел позволяет настраивать интеграцию с HBB TV провайдерами. На текущий момент интегрированы 2 провайдера: Teletarget и GetShopTV. Для получения ключей и URL-адресов необходимо обращаться к самим провайдерам напрямую.

После настройки провайдеров в данном разделе появится возможность задать показ интерактивов от конкретного провайдера для конкретных каналов. HbbTV ID совпадает с UID клиента на странице «Общие настройки Client».

Отображение HbbTV-интерактивов доступно не на всех устройствах и не во всех шаблонах клиентских приложений.

### 3.2.4.6. Виджеты для интеграции с сайтом

В этом разделе настраиваются виджеты для интеграции сайта с сервисом IPTV. Подробнее о механизме *встраивания модулей в сайт*.

Доступны следующие типы виджетов:

- *Channel list* - список телеканалов с группировкой по тарифным пакетам и возможностью поиска.
- *Registration* - страница регистрации с помощью e-mail и СМС.
- *Account page* - личный кабинет абонента, из которого доступно подключение/отключение тарифных пакетов, оплата, редактирования профиля и др.
- *EPG program* - телепрограмма на все подключенные телеканалы.

## 3.2.5 3.2.5. Раздел: Настройки стриминга

### 3.2.5.1. Дата-центры

Под дата-центром подразумевается либо физический узел размещения группы серверов, либо виртуальная группа стриминг-сервисов. Используется для объединения сервисов и дальнейшей маршрутизации на основании предпочтительного географического либо иного отношения аккаунтов к тем или иным сервисам.

### 3.2.5.2. Стриминг-сервисы

Стриминг-сервисы представляют собой серверы, осуществляющие вещание и обработку видеопотоков. Набор настроек различается в зависимости от типа выбранного стриминг-сервиса, однако параметры в блоках *Основные параметры* являются общими для всех.

#### 3.2.5.2.1. Динамическая и статическая маршрутизация

Если для телеканала, фильма или другой единицы контента заданы активные стриминг-сервисы и не задан прямой URI потока, то будет использована динамическая маршрутизация. В момент обращения абонентской приставки к соответствующему контенту осуществляется поиск одного из подключенных стриминг-сервисов на основании типа контента, подключенных тарифных пакетов, а также доступности и нагруженности сервиса. Затем, исходя из настроек стриминг-сервиса, формируется URL контента, по маске либо после вычисления скрипта.

При статической маршрутизации URL контента генерируется при формировании плейлиста. Такой тип маршрутизации может быть использован для потоков без авторизации, Multicast-потоков для IPTV, либо внешних Unicast-потоков партнеров.

#### 3.2.5.2.2. Динамическая маршрутизация, задаваемая скриптом

Скрипт позволяет создать нестандартную логику маршрутизации. Используемый язык - Python. В результате работы скрипта должна быть определена переменная `uri`, содержащая URL видеопотока.

Пример скрипта:

```

def get_random_proxy(datacenter):
    if datacenter == 4:
        proxies = [
            {
                'ip': '1.1.1.1', 'port': 8181,
                'key': 'DrRSwkrMudmsYbOK'
            },
            {
                'ip': '2.2.2.2', 'port': 8181,
                'key': 'DrRSwkrMudmsYbOK'
            },
            {
                'ip': '3.3.3.3', 'port': 8181,
                'key': 'DrRSwkrMudmsYbOK'
            }
        ]
    else:
        return 0
    return random.choice(proxies)

uri = 'http://1.2.3.4:8080/%s/?s=DeZcC2A00kjLw1Bb' % prefix

proxy = get_random_proxy(adid)
if proxy:
    uri = 'http://%s:%d/%s/%s' % (proxy['ip'], proxy['port'], proxy['key'], uri.replace('http://',
↪ ''))

```

Выше приведен пример скрипта, в котором URL видеопотока задается сначала по маске, а затем, если у аккаунта задан определенный дата-центр (id = 4 в примере), то для него случайным образом выбирается один из прокси-серверов, после чего URL заменяется на прокси.

### 3.2.5.3. Технические работы

Технические работы используются для частичного ограничения доступа к сервису когда это необходимо. Например, в заданный временной период, пока проводятся технические работы либо произошла авария, абонентам может быть недоступен просмотр записанных программ.

## 3.2.6 3.2.6. Раздел: Биллинг

### 3.2.6.1. Тарифные пакеты

Раздел позволяет управлять списком тарифных пакетов и их настройками. См. Возможности тарификации.

### 3.2.6.3. Финансовые операции

Раздел содержит информацию о движении денежных средств по аккаунтам абонентов.

Данные могут добавляться как вручную, так и автоматически в случае использования биллинга (см. *Сценарии взаимодействия с биллинговой системой*). Если используется внешняя система биллинга, то для получения списка транзакций в этом разделе необходима синхронизация через Billing API.

Поиск здесь представляет собой фильтр, как по одному параметру, так и по нескольким сразу:

**Параметры**

Клиент

Аккаунт

От До

#ID или поиск по примечанию

Все операции ▾

Только подтвержденные операции

**НАЙТИ**

Также доступен экспорт отчета по транзакциям в файл CSV:

**СОХРАНИТЬ ОТЧЕТ ▾**

анс В CSV-файл

после **Примечание**

## 3.2.7 3.2.7. Раздел: Настройки контента

### 3.2.7.1. Категории ТВ

В этом разделе добавляются категории телеканалов. Каждый телеканал должен обязательно относиться к той или иной категории. В абонентском приложении, в зависимости от шаблона, но как правило, присутствует возможность отображения телеканалов определенной категории для упрощения поиска нужного контента.

### 3.2.7.2. Каналы

Это один из основных разделов для настройки сервиса IPTV/OTT. Здесь производится настройка списка телеканалов, которые вещает оператор, а также конфигурация их вещания и отображения.

Кроме ручного выставления порядка каналов с помощью поля *Порядок сортировки* списку телеканалов можно автоматически задать сортировку, которая будет использоваться на устройствах пользователей, используя методы из списка **Авто-сортировка**, который расположен выше остальных кнопок управления:

**Автоматически** Сортировка осуществляется по номерам кнопок каналов, которые задаются в поле *Номер для сортировки* при настройке EPG-каналов, см. *Настройка EPG*. При использовании Microimpuls Middleware как платформы от ООО «Майкроимпульс» в рамках услуги «Виртуальный оператор» данный метод отсортирует каналы согласно заключенным лицензионным договорам между ООО «Майкроимпульс» и правообладателями и действующему законодательству.

**По ID** При добавлении канала в список ему присваивается ID, данная сортировка происходит по этому параметру.

**По названию** Сортировка осуществляется по наименованию канала.

**Пользовательская сортировка** Если была применена одна из предыдущих сортировок, выбор этого пункта вернет к первоначальной ручной сортировке оператора.

### 3.2.7.3. Телепрограмма

Раздел позволяет просматривать EPG для всех каналов, а также очищать и принудительно переимпортировать EPG для отдельных каналов.

Выбор канала осуществляется в левом меню. Для очистки телепрограммы необходимо нажать кнопку **Очистить EPG**, для импортирования - **Принудительно импортировать EPG**.

---

**Примечание:** Автоматический импорт настраивается через планировщик, см. *Настройка выполнения команд в crontab*.

Расширенное редактирование EPG доступно в служебной панели администратора, см. *Редактирование EPG в ручном режиме*.

---

### 3.2.7.4. Жанры и категории VOD

В этом разделе можно добавлять/удалять и редактировать жанры для фильмов, предоставляемых по услуге Video-On-Demand.

Жанры отображаются в пользовательском интерфейсе на абонентских устройствах, при выборе пункта меню, соответствующего данной услуге. Так же для определения порядка, в котором жанры выводятся на устройстве абонента, используется колонка *Порядок сортировки*.

### 3.2.7.5. Фильмы

В этом разделе осуществляется управление каталогом фильмов и видео-файлов.

В списке фильмов есть кнопка **Assets**, при нажатии на которую будет открыт раздел редактирования ассетов (файлов), относящихся к данному видео. У одного видео может быть несколько ассетов, выбор конкретного ассета для воспроизведения доступен абоненту при просмотре информации о фильме на своем устройстве.

Обратите внимание, что для того чтобы вернуться к изначальному списку ассетов видеотеки, следует нажать кнопку **Вернуться к списку**, которая расположена над списком.

### 3.2.7.6. Радиостанции

В этом разделе осуществляется редактирование списка радиостанций.

### 3.2.7.7. Рекламные ролики

В этом разделе добавляются рекламные ролики, которые затем могут быть сформированы в *Рекламные блоки* (см. далее).

---

**Примечание:** Поддержка данного функционала возможна не на всех устройствах и её реализация зависит от абонентского приложения.

---

### 3.2.7.8. Рекламные блоки

В этом разделе добавляются рекламные блоки, состоящие из последовательности роликов.

### 3.2.7.9. Каталог приложений

В данном разделе производится управление каталогом внешних приложений, доступных на абонентском устройстве в портале, кроме основного сервиса IPTV. Внешним приложение может быть, например, плеер Youtube, онлайн-чат, служба прогноза погоды или пробок, игры и другие сервисы. Приложение представляет собой Web-страницу на Javascript.

Типы приложений:

- Web-приложение во внешнем окне - любой URL, открывающийся в браузере устройства. Особенность данного вида приложения заключается в том, что не на всех устройствах после его открытия можно вернуться обратно в родительское приложение.
- Web-приложение во внутреннем окне - данный тип приложения используется для собственных виджетов оператора, написанных специально для абонентского портала Justify. Для разработки такого приложения можно воспользоваться документацией: <http://mi-justify-dev-docs.readthedocs.io/>

- Ссылка на раздел видеотеки - при создании данного типа виджета в приложении появится ещё одна ссылка на открытие видеотеки (в главном меню или в списке сервисов, в зависимости от настройки виджета).
- Воспроизведение потока по ссылке - воспроизведение любого потока по ссылке, указанной в поле «URL / Название / ID приложения».
- Виджет - в приложение будет добавлен внутренний виджет, разработанный специально для шаблона. Виджеты для шаблона *futuristic*:
  - **Прогноз погоды**
    - \* Системное название: WeatherWidget
    - \* url: /templates/futuristic/default/apps/weather-widget/weather.widget.js
    - \* **Атрибуты:**
      - city\_\_NUM\_\_name: название города
      - city\_\_NUM\_\_id: идентификатор города в источнике, заданном в конфиге Smarty
  - **Курс валют**
    - \* Системное название: ExchangeWidget
    - \* url: /templates/futuristic/default/apps/exchange-widget/exchange.widget.js
  - **Телеканал**
    - \* Системное название: TVChannelWidget
    - \* url: /templates/futuristic/default/apps/tvchannel-widget/tvchannel.widget.js
    - \* **Атрибуты:**
      - number: номер канала по порядку в Smarty
  - **Баннер**
    - \* Системное название: PromoImageWidget
    - \* url: /templates/futuristic/default/apps/promo-image-widget/promo.image.widget.js
    - \* **Атрибуты:**
      - image\_url: адрес превью-картинки баннера
      - big\_image\_url: адрес полноэкранной картинки баннера
      - refresh\_interval: интервал для обновления картинки с сервера (в секундах)
  - **Новости**
    - \* Системное название: NewsWidget
    - \* url: /templates/futuristic/default/apps/news-widget/news.widget.js
    - \* **Атрибуты:**
      - rss\_url: адрес RSS-ленты
  - **Баннер-ссылка**
    - \* Системное название: LinkImageWidget
    - \* url: /templates/futuristic/default/apps/link-image-widget/link.image.widget.js
    - \* **Атрибуты:**

- `image_url`: адрес превью-картинки баннера
- `link_url`: адрес ссылки, открывающийся при запуске виджета

– **Поиск**

- \* Системное название: `SearchWidget`
- \* url: `/templates/futuristic/default/apps/search-widget/search.widget.js`

– **Промо фильма**

- \* Системное название: `PromoVodWidget`
- \* url: `/templates/futuristic/default/apps/promo-vod-widget/promo.vod.widget.js`
- \* **Атрибуты:**
  - `items__NUM__id`: идентификатор фильма/подписки в Smarty
  - `items__NUM__content_type`: тип контента, значения: 0 - фильм, 1 - подписка
  - `items__NUM__content_name`: название контента
  - `items__NUM__trailer_url`: url потока трейлера фильма, который запустится при нажатии на виджет
  - `items__NUM__preview_url`: url превью-картинки

– **Промо канала**

- \* Системное название: `PromoStreamWidget`
- \* url: `/templates/futuristic/default/apps/promo-stream-widget/promo.stream.widget.js`
- \* **Атрибуты:**
  - `items__NUM__trailer_url`: url потока трейлера канала, который запустится при нажатии на виджет
  - `items__NUM__preview_url`: url превью-картинки

– **ТВ Премьеры**

- \* Системное название: `PremieresWidget`
- \* url: `/templates/futuristic/default/apps/premieres-widget/premieres.widget.js`

Виджеты для шаблона *impuls*:

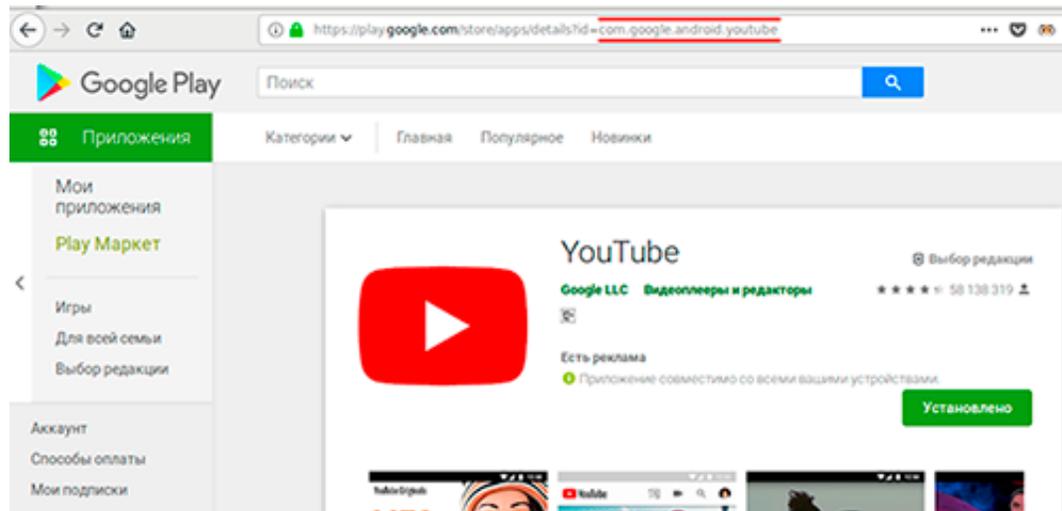
- **Тест скорости (корректная работа гарантирована на `tvip`, `redbox`, `tizen_tv`, прочие устройства в ра**

- Системное название: `SpeedTestWidget`
- url: `/templates/impuls/default/apps/speedtest-widget/speedtest.widget.js`
- **Атрибуты:**

- \* `speedtest_server_url`: url сервера, до которого измеряется скорость (на сервере должны находиться дополнительные файлы для корректной работы виджета, выдаются по запросу)

- Экран настроек Android - виджет, который при запуске открывает системные настройки Android
- Меню приложений Android - виджет, который при запуске открывает системное меню приложений Android

- Запуск приложения Android по AppId - виджет, открывающий заданное системное или установленное приложение Android. Для данного виджета в поле «URL / Название / ID приложения» задается системное имя приложения, которое можно узнать несколькими способами:
  - на некоторых версиях Android его можно узнать, открыв: Настройки -> Приложения -> Интересующее приложение
  - если приложение было скачано из Google Play, идентификатор можно посмотреть в строке браузера:



- если оба способа выше не подходят/не помогли, то можно установить специальное приложение Package Name Viewer, благодаря которому появится возможность просмотреть идентификатор всех установленных приложений.
- Ссылка на раздел ТВ - при создании данного типа виджета в приложении появится ещё одна ссылка на открытие меню ТВ (в главном меню или в списке сервисов, в зависимости от настройки виджета).
- Запуск системного приложения по ID - аналог виджета «Запуск приложения Android по AppId», но предназначенный для других устройств с возможностью запуска внешних приложений. На данный момент такая возможность доступна на приставках Tvip (идентификаторы приложений можно посмотреть в официальной документации Tvip [https://wiki.tvip.ru/stb/system\\_uri](https://wiki.tvip.ru/stb/system_uri)), телевизорах Samsung Smart TV 2015-2019 годов выпуска и приставках Imaqliq (идентификаторы *youtube* и *youtube\_kids*).

## 3.2.8 3.2.8. Раздел: Абоненты

### 3.2.8.1. Абоненты

В данном разделе производится заведение абонентов. В списке абонентов есть три специальные колонки: *Аккаунты*, *Платежи* и *Сообщения* - они содержат ссылки на соответствующие связанные с абонентом разделы.

При нажатии на имя абонента открывается карточка абонента и страница редактирования его параметров.

### 3.2.8.2. Аккаунты

В этом разделе производится заведение аккаунтов абонентов. При нажатии на номер аккаунта открывается карточка аккаунта, где также доступно редактирование его параметров.

### 3.2.8.3. Устройства

В этом разделе отображается список зарегистрированных устройств абонентов. Информация об устройствах добавляется в систему автоматически, при первом подключении абонента, но также допускается и ручное добавление устройств.

### 3.2.8.4. Сообщения и команды

В этом разделе можно создавать информационные рассылки на устройства абонентов. Такие сообщения могут быть добавлены как вручную из интерфейса личного кабинета, так и добавляются системой автоматически, например при поступлении платежа из личного кабинета, или подключении/отключении тарифного пакета, или при приближающейся дате окончания подписки.

---

**Примечание:** В большинстве абонентских приложений Microimpuls входящие сообщения реализованы как всплывающие окна.

---

|  |
|--|
| <p><b>Предупреждение:</b> Мобильные и нативные приложения могут не поддерживать HTML-форматирование.</p> |
|--|

#### 3.2.8.4.1. Массовая рассылка сообщений

Инструмент **Массовая рассылка** позволяет сформировать рассылку сообщений группе абонентов, которую можно выбрать по нескольким критериям:

- Период последней активности аккаунтов - позволяет выбрать абонентов, которые использовали сервис в определенный период.
- Тарифные пакеты - позволяет выбрать тарифные пакеты, в этом случае в выборку попадут абоненты, которым подключены выбранные тарифные пакеты.

При использовании массовой рассылки в теме и тексте сообщения можно использовать переменные, которые будут автоматически заменены на значения в момент создания сообщения: `$firstname` - имя абонента, `$lastname` - фамилия абонента.

### 3.2.8.5. Действия абонентов

В этой вкладке у администратора Smarty есть возможность проследить действия абонентов, такие как: изменение пароля, обновление данных профиля, подключение/отключение тарифов и так далее. Помимо действий также указывается их источник - виджет для интеграции с сайтом или приложение (TVMW API).

Для того, чтобы данный раздел стал доступен, необходимо включить опцию `TVMIDDLEWARE_CUSTOMER_LOG_ENABLED`.

### 3.2.8.6. Дилеры

Дилеры — это партнеры, которые могут предоставлять услуги и взаимодействовать с абонентами от имени оператора. В этом разделе указывается информация о дилерах, набор полей и структура раздела схожа со страницей *Абоненты*.

Отдельно стоит обратить внимание на поля *Имя пользователя* и *Пароль* - эти данные нужны для создания учетной записи дилера в панели администрирования Smarty. Такая учетная запись имеет ограниченные права и не имеет доступа к некоторым возможностям, однако позволяет создавать абонентов и аккаунтов, таким образом подключая их к сервису.

Созданные абоненты и аккаунты автоматически привязываются к дилеру.

### 3.2.9 3.2.9. Раздел: Мониторинг

Актуальная документация: <https://microimpuls.com/docs/smarty/admin-guide/monitoring>

## 3.3 3.3. Общие особенности работы с услугами и аккаунтами в Smarty

Актуальная информация об особенностях работы встроенного биллинга в Smarty в базе знаний продукта на сайте <https://microimpuls.com/docs/smarty/external-billing-integration/about-internal-billing>



---

## 4. Логирование работы Smarty

---

### 4.1 4.1. Логи Smarty

Записи в логах имеют следующий вид:

```
%(timestamp)% %(log level)% %(module)%:%(code line number)% %(function)%[% (pid)%.(thread id)%]  
↳%(message text)%
```

Сообщение (message text) состоит из названия события и дополнительного контекста.

Пример записи в логе:

```
Fri Mar 24 10:22:35+0300 2017 DEBUG api:621 get[83630.Thread-3] SOME_EVENT 'ip'='127.0.0.1' args=[  
↳'foo', 'bar']
```

В этом примере:

**Fri** день недели

**Mar** месяц

**24** дата

**10:22:35** локальное время в таймзоне, которая указана в settings.py в константе TIME\_ZONE

**+0300** смещение относительно UTC

**2017** год

**DEBUG** уровень лога

**api:621** название файла и номер строки

**get[83630.Thread-3]** название функции, pid и имя треда

**SOME\_EVENT** название события

**ip** и **args** дополнительный контекст

Уровень лога выставляется в настройках проекта(smarty/settings/settings\_<local>.py). Например, чтобы выставить уровень INFO для лога запросов к api, необходимо добавить следующий код:

```
LOGGING['handlers']['smarty_api_requests_handler']['level'] = "INFO"
```

События в зависимости от типа группируются в разных файлах. По умолчанию, логи сохраняются по адресу /var/log/microimpuls/smarty/.

### 4.1.1 4.1.1. smarty\_main - основной лог

Типы событий:

#### URLS\_INITIALIZED

Конфигурация Smarty инициализирована. Происходит во время старта приложения, дополнительный контекст:

- `available_apps` - доступные модули

#### События, необработанные другими логгерами

Происходит во время ошибки, необходимо обратиться к разработчику для решения проблемы.

Дополнительный контекст: локальные переменные функции, в которой произошла ошибка.

### 4.1.2 4.1.2. smarty\_accounts - лог абонентов

Типы событий:

#### LOGIN\_SUCCESS

Успешная авторизация, контекст:

- `login_type` - тип логина (мультилогин, базовое устройство, дополнительное устройство)
- `login_method` - метод логина (по абонементу и паролю, только по абонементу, только по UID устройства)
- `params` - параметры запроса

#### LOGIN\_ERROR

Ошибка авторизации, контекст:

- `reason` - причина ошибки:
  - `account_does_not_exist` - аккаунт не существует - абонемент или пароль или `device_uid` неверный
  - `account_is_not_active` - аккаунт не активен
  - `client_does_not_exist` - неверный `client_id` или `api_key`
  - `basic_device_sessions_limit` - превышен лимит сессий на базовом устройстве

- – `account_device_invalid` - такое базовое устройство уже существует
- – `external_api_error` - ошибка запроса к внешней системе (например, к внешнему биллингу)
- – `unknown_error` - неизвестная ошибка (сопровождается `stacktrace`'ом)
- `params` - параметры запроса

## ACCOUNT\_DEVICE\_ERROR

Ошибка при создании устройства аккаунта, контекст:

- `account` - аккаунт
- `device` - устройство
- `device_uid` - UID устройства
- `client` - оператор

## ACCOUNT\_ACTIVATED

Аккаунт был активирован, контекст:

- `activation_result` - ответ, который вернула внешняя система
- `params` - параметры запроса

## ACCOUNT\_CREATED

Создание аккаунта, контекст:

- `account_id` - идентификатор аккаунта

## ACCOUNT\_CHANGED

Изменение аккаунта, контекст:

- `account_id` - идентификатор аккаунта
- изменённые поля

## ACCOUNT\_TARIFFS\_ASSIGNED

Добавление тарифов аккаунту, контекст:

- `account_id` - идентификатор аккаунта
- `tariffs_ids` - список идентификаторов подключенных тарифов

## ACCOUNT\_TARIFFS\_REMOVED

Удаление тарифов у аккаунта, контекст:

- `account_id` - идентификатор аккаунта
- `tariffs_ids` - список идентификаторов удаленных тарифов

## CUSTOMER\_CREATED

Создание абонента, контекст:

- `customer_id` - идентификатор абонента

## CUSTOMER\_CHANGED

Изменение абонента, контекст:

- `customer_id` - идентификатор абонента
- изменённые поля

## CUSTOMER\_TARIFFS\_ASSIGNED

Добавление тарифов абоненту, контекст:

- `customer_id` - идентификатор абонента
- `tariffs_ids` - список идентификаторов подключенных тарифов

## CUSTOMER\_TARIFFS\_REMOVED

Удаление тарифов у абонента, контекст:

- `customer_id` - идентификатор абонента
- `tariffs_ids` - список идентификаторов удаленных тарифов

## ACCOUNT\_DEVICE\_REMOVED

Удаление устройства аккаунта, контекст:

- `account_id` - идентификатор аккаунта
- `device_uid` - идентификатор устройства

### 4.1.3 4.1.3. smarty\_billing\_out - запросы к внешним системам

## INIT\_ERROR

Ошибка инициализации обработчика API, контекст:

- `kwargs` - аргументы, переданные в класс обработчика
- `api_client_class` - название класса обработчика API
- `stacktrace`

## CUSTOMER\_BALANCE\_REQUEST\_ERROR

Ошибка при запросе баланса, контекст:

- `api_client_class` - название класса обработчика API
- `params` - параметры запроса
- `stacktrace`

## CUSTOMER\_BALANCE\_REQUEST\_SUCCESS

Успешный запрос баланса, контекст:

- `api_client_class` - название класса обработчика API
- `params` - параметры запроса
- `result` - результат запроса

## CUSTOMER\_PAYMENT\_LIST\_REQUEST\_ERROR

Ошибка при запросе списка транзакций, контекст:

- `api_client_class` - название класса обработчика API
- `params` - параметры запроса
- `stacktrace`

## CUSTOMER\_PAYMENT\_LIST\_REQUEST\_SUCCESS

Успешный запрос списка транзакций, контекст:

- `api_client_class` - название класса обработчика API
- `params` - параметры запроса
- `result` - результат запроса

## VIDEO\_ACTIONS\_LIST\_REQUEST\_ERROR

Ошибка при запросе вариантов действий с видео, контекст:

- `api_client_class` - название класса обработчика API
- `params` - параметры запроса
- `stacktrace`

## VIDEO\_ACTIONS\_LIST\_REQUEST\_SUCCESS

Успешный запрос вариантов действий с видео, контекст:

- `api_client_class` - название класса обработчика API
- `params` - параметры запроса
- `result` - результат запроса

## VIDEO\_ACTION\_REQUEST\_ERROR

Ошибка при попытке произвести действие с видео, контекст:

- `api_client_class` - название класса обработчика API
- `params` - параметры запроса
- `stacktrace`

## VIDEO\_ACTION\_REQUEST\_SUCCESS

Успешное действие с видео, контекст:

- `api_client_class` - название класса обработчика API
- `params` - параметры запроса
- `result` - результат запроса

### 4.1.4 4.1.4. smarty\_billing\_in - входящие запросы к Billing API

#### BILLING\_REQUEST\_ERROR

Ошибка при запросе к Billing API, контекст:

- `url` - URL метода, к которому производился запрос
- `ip` - IP-адрес, с которого производился запрос
- `args` - аргументы запроса
- `error_message` - сообщение об ошибке
- `error` - код ошибки

#### BILLING\_REQUEST\_SUCCESS

Успешный запрос в биллинг, контекст:

- `url` - URL метода, к которому производился запрос
- `ip` - IP-адрес, с которого производился запрос
- `args` - аргументы запроса

### 4.1.5 4.1.5. smarty\_cache - события, связанные с кешированием

#### OBJECT\_CACHED

Объект закеширован, контекст:

- `object` - кешируемый объект
- `timeout` - время, по истечении которого объект будет удален из кеша
- `key` - ключ объекта в кеше
- `deps` - объекты, при изменении которых кешируемый объект должен быть инвалидирован

## OBJECT\_INVALIDATED

Объект инвалидирован, контекст:

- `object` - объект, который был удален из кеша
- `deps_key` - ключ объекта, где находятся ключи связанных объектов, которые тоже должны быть инвалидированы

### 4.1.6 4.1.6. smarty\_messaging - лог отправленных сообщений для аккаунтов

## MESSAGE\_CREATED

Создано сообщение, контекст:

- `account` - аккаунт, которому было отправлено сообщение
- `subject` - тема сообщения
- `text` - текст сообщения

## MESSAGE\_SEND

Сообщение отправлено, контекст:

- `account` - аккаунт, которому было отправлено сообщение
- `subject` - тема сообщения
- `text` - текст сообщения
- `uuid` - идентификатор сообщения

## MESSAGE\_DELETED

Сообщение удалено, дополнительный контекст:

- `account` - аккаунт, которому было отправлено сообщение
- `subject` - тема сообщения
- `text` - текст сообщения
- `uuid` - идентификатор сообщения

### 4.1.7 4.1.7. smarty\_management - лог периодических команд

## MANAGEMENT\_COMMAND\_SUCCESS

Успешное выполнение команды, дополнительный контекст:

- `command` - название команды
- `execution_time` - время выполнения

## MANAGEMENT\_COMMAND\_ERROR

Ошибка выполнения команды, дополнительный контекст:

- `command` - название команды
- `stacktrace`

## 4.1.8 4.1.8. smarty\_epg - лог импорта EPG

### EPG\_CHANNEL\_IMPORTED

Программы для канала успешно импортированы, дополнительный контекст:

- `epg_channel` - канал
- `source` - источник EPG
- `programs_imported` - количество импортированных программ

### EPG\_CHANNEL\_IMPORT\_ERROR

Ошибка при импорте программ, дополнительный контекст:

- `epg_channel` - канал
- `source` - источник EPG
- `stacktrace`

### EPG\_IMPORT\_FINISHED

Импорт программ завершен, дополнительный контекст:

- `channels_processed` - количество обработанных каналов
- `programms_imported` - количество импортированных программ

### EPG\_REMOVED

В ходе парсинга были удалены старые записи, дополнительный контекст:

- `epg_channel` - канал
- `source` - источник epg
- `removed_objects` - удаленные объекты

### EPG\_TIME\_OVERLAP

Время окончания предыдущей программы больше времени начала текущей, дополнительный контекст:

- `epg_channel` - канал
- `source` - источник epg
- `program_name` - название программы
- `previous_program_name` - название предыдущей программы

- `program_time_begin` - время начала текущей программы
- `previous_time_end` - время окончания предыдущей программы

## EPG\_TIME\_HOLE

Время окончания предыдущей программы меньше времени начала текущей, дополнительный контекст:

- `epg_channel` - канал
- `source` - источник epg
- `program_name` - название программы
- `previous_program_name` - название предыдущей программы
- `program_time_begin` - время начала текущей программы
- `previous_time_end` - время окончания предыдущей программы

## EPG\_NAME\_DOUBLE

Название текущей программы совпадает с предыдущей, дополнительный контекст:

- `epg_channel` - канал
- `source` - источник epg
- `program_name` - название программы

### 4.1.9 4.1.9. smarty\_content\_requests - запросы на получение ссылки/адреса потока через TVMW API

#### CONTENT\_REQUEST\_FAIL

Произошла необработанная ошибка в процессе запроса, необходимо обратиться к разработчику.

Дополнительный контекст:

- `url` - URL метода, к которому производился запрос
- `params` - параметры запроса
- `stacktrace`

#### CONTENT\_REQUEST\_ERROR

Обработанная ошибка в процессе запроса, дополнительный контекст:

- `url` - URL метода, к которому производился запрос
- `params` - параметры запроса

Возможные причины:

- неверные параметры запроса
- устаревший ключ авторизации
- запрос к устаревшим данным (например, попытка воспроизвести слишком старую передачу из архива)

## CONTENT\_REQUEST\_SUCCESS

Успешный запрос, ссылка получена, дополнительный контекст:

- `url` - URL метода, к которому производился запрос
- `params` - параметры запроса
- дополнительная информация, в т.ч. адрес потока (в зависимости от метода)

## CLIENT\_CHANNELS\_NOT\_FOUND

В кеше не обнаружены каналы для данного Client ID, возможно был сброшен кеш или произошла ошибка выполнения команды `cache_channel_list`, дополнительный контекст:

- `client` - Client ID

### 4.1.10 4.1.10. smarty\_portal - лог событий портала

#### PORTAL\_EVENT

Событие в портале, дополнительный контекст:

- `event_description` - описание события
- `ip` - IP-адрес устройства абонента
- `device_uid` - идентификатор устройства
- `screen_name` - название экрана
- `user_agent` - User-Agent устройства

### 4.1.11 4.1.11. smarty\_stream\_services - лог стриминг-сервисов

#### STREAM\_SERVICE\_CHECKING\_ERROR

Ошибка при проверке доступности стриминг-сервиса, дополнительный контекст:

- `stream_service` - стриминг-сервис
- `was_available_before` - указывает, был ли стриминг-сервис доступен ранее
- `check_ping_success` - была ли успешной проверка пингом (опционально)
- `check_tcp_success` - была ли успешной проверка попыткой открыть сокет (опционально)
- `check_http_success` - была ли успешной проверка попыткой открыть URL (опционально)
- `check_is_alive_success` - была ли успешной проверка `is_alive` (опционально)
- `stacktrace`

## STREAM\_SERVICE\_CHECKING\_SUCCESS

Успешная проверка доступности стриминг-сервиса, дополнительный контекст:

- `stream_service` - стриминг-сервис
- `was_available_before` - указывает, был ли стриминг-сервис доступен ранее
- `check_ping_success` - проверка пингом была успешной (опционально)
- `check_tcp_success` - проверка попыткой открыть сокет была успешной (опционально)
- `check_http_success` - проверка попыткой открыть URL была успешной (опционально)
- `check_is_alive_success` - проверка `is_alive` была успешной (опционально)

### 4.1.12 4.1.12. smarty\_admin - лог панели администрирования Smarty

#### ADMIN\_REQUEST

Запрос к административному интерфейсу, дополнительный контекст:

- `user` - пользователь, осуществивший запрос
- `ip` - IP пользователя
- `path` - путь запроса
- `user_agent` - User-Agent браузера

### 4.1.13 4.1.13. smarty\_videoservices - лог обращений к видеосервисам

#### VIDEOSERVICES\_REQUEST

Запрос пользователя в Smarty на выполнение команды:

- `ip` - IP пользователя
- `user_agent` - User-Agent браузера
- `user` - пользователь, осуществивший запрос
- `path` - путь запроса

#### VIDEOSERVICES\_API\_REQUEST

Запрос Smarty к видеосервису:

- `host` - тип и адрес видеосервиса
- `args` - аргументы запроса
- `command` - вызываемый метод
- `message` - ответ сервера (только при ошибке)

#### 4.1.14 4.1.14. smarty\_payment - лог оплаты

##### NOTIFY\_ERROR

Ошибка обработки сообщения нотификации, дополнительный контекст:

- `client` - ID клиента в платёжном шлюзе.
- `transaction` - ID транзакции.
- `payment_source` - название платёжного шлюза.
- `params` - параметры запроса.
- `error` - описание ошибки.

Возможные причины ошибки:

- Неверная настройка платёжного шлюза.
- Передача неверных параметров платёжному шлюзу.
- Платёж не прошёл.

##### NOTIFY\_SUCCESS

Успешная нотификация, дополнительный контекст:

- `client` - ID клиента в платёжном шлюзе.
- `transaction` - ID транзакции.
- `payment_source` - название платёжного шлюза.
- `params` - параметры запроса.

##### PAYTURE\_REQUEST

Запрос к платёжному шлюзу Payture:

- `url` - URL API Payture, на который выполняется запрос.
- `args` - аргументы запроса.
- `response` - ответ от API в виде XML.

---

## 5. Интеграция Smarty с внешними системами и сервисами

---

### 5.1 5.1. API для разработчиков

Актуальная документация по API-методам в базе знаний продукта на сайте: <https://microimpuls.com/docs/smarty/extra/api>

#### 5.1.1 5.1.1. Примеры кода

- Библиотека на Python, реализующая Billing API: <https://github.com/microimpuls/smarty-billing-api-python>
- Скрипт интеграции с биллингом Гидра: [https://github.com/microimpuls/admin-tools/tree/master/hydra\\_billing\\_script](https://github.com/microimpuls/admin-tools/tree/master/hydra_billing_script)

---

**Примечание:** Смотрите также другие *дополнительные инструменты*

---

### 5.2 5.2. Интеграция с биллинговой системой

Актуальная документация о возможностях интеграции с внешним биллингом оператора в базе знаний продукта на сайте <https://microimpuls.com/docs/smarty/external-billing-integration>

### 5.3 5.3. Встраивание модулей в сайт

Актуальная документация по интеграции виджетов на сайт: <https://microimpuls.com/docs/smarty/extra-services-integration/site-widgets>

## 5.4 5.4. Интеграция с популярными видео-серверами

### 5.4.1 5.4.1. Интеграция с Astra

Для интеграции механизма авторизации видеопотоков (стриминг-сервисов) с [Astra](#) используется механизм генерации одноразовых токенов для ссылок на поток на стороне сервера Smarty. Astra в момент разбора HTTP Request от абонентского устройства выделяет токен и проверяет его на сервере Smarty на валидность.

Для генерации токена необходимо в маске URL стриминг-сервиса в Smarty добавить переменную `$token` в маску URL, например:

```
http://streamer.example.com:8080/mychannel/?token=$token
```

---

**Примечание:** Дополнительная информация:

- [StreamService API](#) - метод проверки токена `StreamServiceTokenCheck`
  - [Настройка стриминг-сервисов](#)
- 

### 5.4.2 5.4.2. Интеграция с Flussonic

Для интеграции механизма авторизации видеопотоков (стриминг-сервисов) с [Flussonic](#) используется механизм генерации одноразовых токенов для ссылок на поток на стороне сервера Smarty. Flussonic в момент разбора HTTP Request от абонентского устройства выделяет токен и проверяет его на сервере Smarty на валидность.

Для генерации токена необходимо в маске URL стриминг-сервиса в Smarty добавить переменную `$token` в маску URL, например:

```
http://streamer.example.com:8080/mychannel/?token=$token
```

На стороне Flussonic необходимо настроить авторизационный бекенд, указав адрес API-метода `StreamServiceTokenCheck` на стороне Smarty:

```
auth_backend main {
    backend https://smarty.example.com/tvmiddleware/api/streamservice/token/check/;
}
```

и в свойствах канала прописать параметр `auth`:

```
stream example {
    url hls://example.com/channel/index.m3u8;
    title "Channel Name";
    auth auth://main;
}
```

---

**Примечание:** Дополнительная информация:

- [StreamService API](#) - метод проверки токена `StreamServiceTokenCheck`
  - [Настройка стриминг-сервисов](#)
-

## 5.5 5.5. Интеграция с онлайн-кинотеатрами

### 5.5.1 5.5.1. Интеграция с MEGOGO

Smarty содержит модуль для интеграции с онлайн-кинотеатром MEGOGO, который включает в себя следующий функционал:

- Синхронизация фильмов и сериалов MEGOGO со встроенной видеотекой Smarty (наименования, описание, жанры, обложки и т.д.)
- Модель подписки (SVOD)
- Модель покупки отдельных фильмов и сериалов (TVOD)
- Все стандартные функции встроенной видеотеки Smarty (поиск, фильтрация, сортировка, группировка по жанрам и т.д.)
- Прозрачная интеграция в стандартные приложения Microimpuls на разных устройствах - единый интерфейс просмотра ТВ и VOD и других сервисов, привычный для абонента

#### 5.5.1.1. Настройка модуля megogo

Для подключения интеграции с MEGOGO необходимо проделать следующие шаги:

- Подключить модуль **megogo** в конфигурации Smarty в секции INSTALLED\_APPS, перезагрузить Smarty и выполнить миграцию данных.
- Создать скрытый тарифный пакет, который будет использоваться для возможности доступа абонентов к каталогу и покупке TVOD или подписке SVOD. Этот тарифный пакет будет автоматически включаться для всех импортированных фильмов для того, чтобы фильм был виден абоненту в каталоге.
- Создать тарифный пакет, который будет использоваться для подписки на пакет фильмов. При покупке SVOD-подписки этот пакет будет подключаться абоненту в Smarty, а при отключении пакета - будет происходить отписка от пакета SVOD.
- В панели администратора в разделе «Общие настройки» -> «Интеграция с API внешних систем» создать новую внешнюю систему:
  - – указать название (например, MEGOGO)
  - – выбрать из выпадающего списка класс API `megogo_api_client`
  - – выбрать тарифный пакет, созданный на втором шаге. Абоненты, у которых подключен данный тарифный пакет, будут видеть фильмы в видеотеке и иметь возможность их купить (по подписке или транзакционно, в зависимости от параметров фильма в MEGOGO)
- Указать дополнительные атрибуты внешней системы:
  - – `xml_url` - ссылка на XML-файл с каталогом фильмов MEGOGO. Для РФ: [http://xml.megogo.net/assets/files/ru/all\\_mgg.xml](http://xml.megogo.net/assets/files/ru/all_mgg.xml)
  - – `mobile_private_key` - закрытый ключ для мобильных устройств (предоставляется MEGOGO)
  - – `mobile_public_key` - открытый ключ для мобильных устройств (предоставляется MEGOGO)
  - – `smart_tv_private_key` - закрытый ключ для Smart TV (предоставляется MEGOGO)
  - – `smart_tv_public_key` - открытый ключ для Smart TV (предоставляется MEGOGO)

- – `stb_private_key` - закрытый ключ для STB (предоставляется MEGOGO)
- – `stb_public_key` - открытый ключ для STB (предоставляется MEGOGO)
- – `partner_id` - идентификатор партнера (предоставляется MEGOGO)
- – `salt` - ключ, используемый для формирования запроса авторизации в биллинге (предоставляется MEGOGO)
- – `svod_service_id` - идентификаторы сервиса SVOD, используемый для оформления услуги пользователю (предоставляется MEGOGO)
- – `available_tvod_collection_id` - идентификатор коллекции для получения доступных TVOD-объектов (предоставляется MEGOGO)
- – `tariff_id` - идентификатор тарифного пакета, созданного на 3 шаге, который будет подключен абоненту при активации подписки. Подключенность этого пакета означает наличие подписки на соответствующий пакет SVOD в MEGOGO. При отключении данного пакета у абонента/аккаунта в Smarty, будет автоматически вызван метод отписки от SVOD в MEGOGO.
- – `additional_tariffs_ids` - идентификаторы смешанных тарифных пакетов (объединяющих телеканалы и подписку MEGOGO), через запятую. При подключении одного из таких тарифов будет произведена подписка на MEGOGO также, как и для `tariff_id`. При отключении всех таких пакетов будет произведена отписка от MEGOGO также, как и для `tariff_id`.
- – `ignore_customer_balance_check` - при значении 1 при покупке контента не будет проверяться баланс абонента. Необходимо при интеграции покупок с внешней биллинговой системой оператора.
- – `typhoid_comment_category_id` - идентификатор жанра-категории, присваиваемый фильму, если он является фильмом с тифлокомментариями. Если данный атрибут указан и фильм имеет тифлокомментарии, то другие категории для него указаны не будут.
- – `sign_language_category_id` - идентификатор жанра-категории, присваиваемый фильму, если он является фильмом с сурдопереводом. Если данный атрибут указан и фильм имеет сурдоперевод, то другие категории для него указаны не будут.

После выполнения всех шагов необходимо произвести синхронизацию контента с помощью `management`-команды:

```
python manage.py megogo_sync_content --settings=settings.<settings filename>
```

Первая синхронизация может занять около получаса в связи со скачиванием обложек, последующие синхронизации проходят быстрее. Для регулярной синхронизации фильмов необходимо добавить вызов команды в `crontab`. Помимо синхронизации фильмов эта команда также создает подборки контента из базы MEGOGO для импортированных фильмов.

Для команды `megogo_sync_content` можно указать флаг `--load_actor_info` для загрузки данных об актёрах со стороны сервиса MEGOGO, однако это потребует больше времени для синхронизации.

После первой синхронизации будут созданы жанры фильмов MEGOGO. Затем необходимо выполнить финальный шаг:

- Создать требуемые жанры видеотеки в Smarty и произвести маппинг жанров MEGOGO к жанрам Smarty в служебной панели администратора по адресу <http://smarty.example.com/admin/megogo/megogogenremap/>.

При последующей синхронизации фильмов произойдет привязка жанров.

---

**Примечание:** Для возможности покупки фильмов, доступных в TVOD, для каждого ключа должна быть подключена данная услуга на стороне MEGOGO.

---

## 5.5.2 5.5.2. Интеграция с tvzavr (deprecated)

---

**Примечание:** Кинотеатр tvzavr более недоступен в рамках Smarty по причине его официального закрытия.

---

Smarty содержит модуль для интеграции с онлайн-кинотеатром tvzavr, который включает в себя следующий функционал:

- Синхронизация фильмов и сериалов tvzavr.ru со встроенной видеотекой Smarty (наименования, описание, жанры, обложки и т.д.)
- Модель подписки (SVOD)
- Все стандартные функции встроенной видеотеки Smarty (поиск, фильтрация, сортировка, группировка по жанрам и т.д.)
- Прозрачная интеграция в стандартные приложения Microimpuls на разных устройствах - единый интерфейс просмотра ТВ и VOD и других сервисов, привычный для абонента

### 5.5.2.1. Настройка модуля tvzavr

Для подключения интеграции с tvzavr необходимо проделать следующие шаги:

- Подключить модуль **tvzavr** в конфигурации Smarty в секции `INSTALLED_APPS`, перезагрузить Smarty и выполнить миграцию данных.
- Создать скрытый тарифный пакет, который будет использоваться для возможности доступа абонентов к каталогу SVOD. Этот тарифный пакет будет автоматически включаться для всех импортированных фильмов для того, чтобы фильм был виден абоненту в каталоге.
- Создать тарифный пакет, который будет использоваться для подписки на пакет фильмов. При покупке SVOD-подписки этот пакет будет подключаться абоненту в Smarty, а при отключении пакета - будет происходить отписка от пакета SVOD.
- В панели администратора в разделе «Общие настройки» -> «Интеграция с API внешних систем» создать новую внешнюю систему:
  - – указать название (например, tvzavr)
  - – выбрать из выпадающего списка класс API `tvzavr_api_client`
  - – выбрать тарифный пакет, созданный на втором шаге. Абоненты, у которых подключен данный тарифный пакет, будут видеть фильмы в видеотеке и иметь возможность купить подписку на них.
- Указать дополнительные атрибуты внешней системы:
  - – `tvzavr_tariff_id` - значение этого параметра выдаёт tvzavr
  - – `plf` - значение этого параметра выдаёт tvzavr
  - – `secret` - значение этого параметра выдаёт tvzavr

- – `subscription_tariff_id` - идентификатор тарифного пакета, созданного на 3 шаге, который будет подключен абоненту при активации подписки. Подключенность этого пакета означает наличие подписки на соответствующий пакет SVOD в tvzavr. При отключении данного пакета у абонента/аккаунта в Smarty, будет автоматически вызван метод отписки от SVOD в tvzavr.

После выполнения всех шагов необходимо произвести синхронизацию контента с помощью `management`-команды:

```
python manage.py tvzavr_sync_content --settings=settings.<settings filename>
```

Первая синхронизация может занять около получаса в связи со скачиванием обложек, последующие синхронизации проходят быстрее. Для регулярной синхронизации фильмов необходимо добавить вызов команды в `crontab`.

После первой синхронизации будут созданы жанры фильмов tvzavr. Затем необходимо выполнить финальный шаг:

- Создать требуемые жанры видеотеки в Smarty и произвести маппинг жанров tvzavr к жанрам Smarty в служебной панели администратора по адресу <http://smarty.example.com/admin/tvzavr/tvzavrgenremap/>.

При последующей синхронизации фильмов произойдет привязка жанров.

---

**Примечание:** Для того, чтобы оформленные подписки продлевались на стороне сервера tvzavr, необходимо также по расписанию вызывать команду `check_accounts`.

---

### 5.5.3 5.5.3. Интеграция со Start

Smarty содержит модуль для интеграции с онлайн-кинотеатром `Start`, который включает в себя следующий функционал:

- Синхронизация фильмов и сериалов `start.ru` со встроенной видеотекой Smarty (наименования, описание, жанры, обложки и т.д.)
- Модель подписки (SVOD)
- Все стандартные функции встроенной видеотеки Smarty (поиск, фильтрация, сортировка, группировка по жанрам и т.д.)
- Прозрачная интеграция в стандартные приложения `Microimpuls` на разных устройствах - единый интерфейс просмотра ТВ и VOD и других сервисов, привычный для абонента

#### 5.5.3.1. Настройка модуля Start

Для подключения интеграции с tvzavr необходимо проделать следующие шаги:

- Подключить модуль `cinema_start` в конфигурации Smarty в секции `INSTALLED_APPS`, перезагрузить Smarty и выполнить миграцию данных.
- Создать скрытый тарифный пакет, который будет использоваться для возможности доступа абонентов к каталогу SVOD. Этот тарифный пакет будет автоматически включаться для всех импортированных фильмов для того, чтобы фильм был виден абоненту в каталоге.
- Создать тарифный пакет, который будет использоваться для подписки на пакет фильмов. При покупке SVOD-подписки этот пакет будет подключаться абоненту в Smarty, а при отключении пакета - будет происходить отписка от пакета SVOD.

- В панели администратора в разделе «Общие настройки» -> «Интеграция с API внешних систем» создать новую внешнюю систему:
  - – указать название (например, Start)
  - – выбрать из выпадающего списка класс API `cinema_start_api_client`
  - – выбрать тарифный пакет, созданный на втором шаге. Абоненты, у которых подключен данный тарифный пакет, будут видеть фильмы в видеотеке и иметь возможность купить подписку на них.
- Указать дополнительные атрибуты внешней системы в правом блоке страницы.

После выполнения всех шагов необходимо произвести синхронизацию контента с помощью `management`-команды:

```
python manage.py cinema_start_sync_content --settings=settings.<settings filename>
```

Первая синхронизация может занять около получаса в связи со скачиванием обложек, последующие синхронизации проходят быстрее. Для регулярной синхронизации фильмов необходимо добавить вызов команды в `crontab`.

После первой синхронизации будут созданы жанры фильмов Start. Затем необходимо выполнить финальный шаг:

- Создать требуемые жанры видеотеки в Smarty и произвести маппинг жанров Start к жанрам Smarty в служебной панели администратора по адресу [http://smarty.example.com/admin/cinema\\_start/startgenremap/](http://smarty.example.com/admin/cinema_start/startgenremap/).

При последующей синхронизации фильмов произойдет привязка жанров.

## 5.6 5.6. Интеграция с CAS CMS

Smarty поддерживает интеграцию с некоторыми системами CAS по модели единой системы управления подписками, при этом порталные приложения и клиенты для приставок и других устройств поддерживают работу с любыми CAS, которые поддерживаются конкретным устройством.

### 5.6.1 5.6.1. Интеграция с Irdeto

Настройки интеграции задаются в файле конфигурации Smarty:

**IRDETO\_NATIONALITY** str Значение по умолчанию: „RUS“

**IRDETO\_REGION\_TAG** str Значение по умолчанию: „MO“

**IRDETO\_HOST** str Адрес сервера Irdeto с SOAP API, по умолчанию „http://127.0.0.1:80“

## 5.7 5.7 Интеграция с платежными системами

Актуальная документация: <https://microimpuls.com/docs/smarty/payments-integration>

### 5.7.1 5.7.1. Интеграция с Payture

Актуальная документация: <https://microimpuls.com/docs/smarty/payments-integration/payture>

## 5.8 5.8. Дополнительные инструменты

**Скрипт миграции данных smarty между БД по client\_id** [https://github.com/microimpuls/admin-tools/tree/master/smarty\\_migrate\\_tool](https://github.com/microimpuls/admin-tools/tree/master/smarty_migrate_tool)

**Скрипт миграции с OFT Middleware на Microimpuls Middleware** [https://github.com/microimpuls/admin-tools/tree/master/oft\\_db\\_migrate\\_tool](https://github.com/microimpuls/admin-tools/tree/master/oft_db_migrate_tool)

**Скрипт миграции аккаунтов и MAC-адресов с Hydra Billing в Microimpuls Middleware** [https://github.com/microimpuls/admin-tools/tree/master/hydra\\_migrate](https://github.com/microimpuls/admin-tools/tree/master/hydra_migrate)

**Скрипт массового создания аккаунтов через Billing API** [https://github.com/microimpuls/admin-tools/tree/master/mass\\_customer\\_creator](https://github.com/microimpuls/admin-tools/tree/master/mass_customer_creator)

---

**Примечание:** Другие полезные скрипты и утилиты см. в репозитории Microimpuls на Github: <https://github.com/microimpuls/admin-tools>

---

---

## 6. Установка и настройка портала для STB и Smart TV

---

Портал - это набор Javascript-приложений (JS/HTML/CSS) в разных стилевых и функциональных вариантах (шаблонов), представляющих собой оболочку (интерфейс) для абонента. Портал загружается встроенным в устройство (приставку или Smart TV) браузером.

### 6.1 6.1. Установка портала

Поскольку портал это статические файлы, не требующие сервера приложений с поддержкой выполнения каких-либо скриптов, то для его хостинга достаточно любого веб-сервера. Также, при необходимости, портал может быть размещен на CDN-сервисе или встроен в прошивку устройства.

С сервером Smarty портал взаимодействует через HTTP API посредством выполнения XMLHttpRequest-запросов.

Для хостинга портала рекомендуется использовать веб-сервер nginx.

Настройки портала задаются в файле `client.js`, размещенном в корневой директории. Для удобства деплоя и администрирования `client.js`, обычно, является симлинком на специфичный для данного провайдера конфиг, размещенный в `/etc/microimpuls/portal/`.

---

**Примечание:** Из-за политики ограничения выполнения кроссдоменных запросов на некоторых устройствах рекомендуется для выполнения запросов к Smarty использовать тот же домен, где размещен портал. В стандартном конфиге nginx для портала, который идет в комплекте с установочным пакетом портала, указан специальный location `/api`, осуществляющий редирект на Smarty.

---

### 6.2 6.2. Параметры конфигурации `client.js`

В конфиге `client.js` задаются как основные параметры (например, данные для подключения к Smarty), так и специфичные для каждого приложения (шаблона).

### 6.2.1 Обязательные параметры

Актуальная документация по обязательным параметрам: <https://microimpuls.com/docs/smarty/portal-and-apps-settings/portal-settings>

### 6.2.2 Дополнительные параметры

Актуальная документация по дополнительным параметрам: <https://microimpuls.com/docs/smarty/portal-and-apps-settings/portal-settings>

### 6.2.3 Специфичные для шаблонов параметры

Актуальная документация по дополнительным параметрам шаблона impuls: <https://microimpuls.com/docs/smarty/portal-and-apps-settings/impuls-settings>

Актуальная документация по дополнительным параметрам шаблона focus: <https://microimpuls.com/docs/smarty/portal-and-apps-settings/focus-settings>

Актуальная документация по дополнительным параметрам шаблона futuristic: <https://microimpuls.com/docs/smarty/portal-and-apps-settings/futuristic-settings>

Актуальная документация по дополнительным параметрам шаблона infinitely: <https://microimpuls.com/docs/smarty/portal-and-apps-settings/infinitely-settings>

## 6.3 6.3. Механизм событий

События позволяют добавлять или переопределять некоторый функционал портала в разные моменты его жизненного цикла. Обработчики событий задаются в конфиге client.js, что позволяет сохранить кастомные настройки и функции даже при обновлении портала на новую версию.

Доступные события:

- *OnDataRequestError* - ошибка выполнения запроса к Smarty, в качестве аргумента - код ошибки
- *OnAppInitBegin* - старт инициализации приложения
- *OnAppInitEnd* - завершение инициализации приложения
- *OnDeviceInitBegin* - старт инициализации драйвера устройства
- *OnDeviceInitEnd* - завершение инициализации драйвера устройства
- *OnDeviceKeyEvent* - событие нажатия клавиши пульта, в качестве аргумента - код клавиши
- *OnAccountLoginSuccessful* - успешная авторизация аккаунта

Жизненный цикл: *OnAppInitBegin* > *OnDeviceInitBegin* > *OnDeviceInitEnd* > *OnAppInitEnd*.

Пример создания обработчиков событий приведен ниже.

---

**Примечание:** Внимание! Использование этого механизма требует навыков программирования на JavaScript, знания архитектуры и API портала и API устройств.

---

## 6.4 6.4. Пример конфигурации

Пример ниже предназначен для шаблона `impuls` и кроме стандартного поведения добавляет дополнительный пункт в меню настроек абонента - «Режим ожидания», включающий или отключающий обработку событий подключения/отключения HDMI-кабеля на приставке MAG, а также проверяет версию прошивки и запускает обновление в случае необходимости при старте приложения и добавляет некоторые другие функции.

```
var CLIENT_SETTINGS = {
  'client_id': 1,
  'api_key': '***',
  'api_url': '/api',
  'template_name': 'impuls',
  'template_size': {
    'impuls': {
      'default': [1280, 720]
    },
    'classic': {
      'default': [1280, 720],
      '720x576': [720, 576]
    },
  },
  'available_templates': ['futuristic'],
  'template_styles': {
    'futuristic': ['futuristic', 'futuristic_x']
  },
  'settings_filename': 'example.dat',
  'site_url': 'www.example.com',
  'signup_auto_activation_period': 0,
  'show_welcome_message': false,
  'registration_available': false,
  'settings_menu_custom_items': [
    'handle-hdmi-events'
  ],
  'auth_mode': 'device_uid',
  'default_timezone': 12,
  'default_buffersize': 3,
};

OnAppInitBegin = function()
{
  // Автоматическое выключение плеера ночью в 01:30 по локальному времени
  setInterval(function(){
    var date = new Date();
    if (date.getHours() == 1 && date.getMinutes() == 30) {
      App.player.stop();
      App.display.showScreen('tvchannels');
    }
  }, 35000);

  // Обновление прошивки для определенных старых версий
  try {
    var fver = parseInt(gSTB.RDir('ImageVersion'));
    var desc = gSTB.GetDeviceImageDesc();
    if (desc == 'default-214') {
      stbUpdate.startAutoUpdate("http://update.example.com/imageupdate", true);
    }
  }
}
```

(continues on next page)

```
    } catch (e) {}
};

OnAppInitEnd = function()
{
    // Подключение custom css верстки
    var el = document.createElement('link');
    el.rel = 'stylesheet';
    el.type = 'text/css';
    el.href = '/custom/css/custom.css';
    document.getElementsByTagName('body')[0].appendChild(el);

    // Подключение режима overscan для шаблона impuls
    Helper.addBodyClass('overscan');

    // Переопределение кнопки EPG на красную кнопку
    App.playerScreen.key_epg = App.playerScreen.key_red;
    App.tvChannelsScreen.key_epg = App.tvChannelsScreen.key_red;

    // Переопределение обработчика кнопки Power
    App.display.setGlobalKeyCodeHandler('power', function(){
        App.player.stop();
        App.display.showScreen('mainmenu');
    }, App.playerScreen);
};

OnDeviceInitBegin = function()
{
    // Добавление меню настройки режима ожидания
    var handleHdmiEventsMenu = new BaseMenu({
        menuTag: 'ul',
        itemIdPrefix: 'settings-handle-hdmi-events-value',
        useItemIdWithoutIndex: true,
        itemTag: 'span',
        displayItemsNumber: 1,
        sourceItemsNumber: 2,
        useFastRefresh: false,
        getItemNameFunc: function(sourceItemIndex, displayItemIndex) {
            var names = [
                'Отключен',
                'Включен'
            ];
            return names[sourceItemIndex];
        }
    });
    App.settings.setCustomSetting('handle-hdmi-events', 1);
    App.lang.set('ru', 'label-settings-handle-hdmi-events', 'Режим ожидания');
    App.settingsScreen.addCustomSettingsMenu('handle-hdmi-events', handleHdmiEventsMenu);
};

OnDeviceInitEnd = function()
{
    // Установка произвольной прозрачности интерфейса
    App.device.setWindowTransparencyLevel(230);

    // Установка режима 16:9 по умолчанию

```

(continues on next page)

(продолжение с предыдущей страницы)

```
App.device.setAspectRatioMode('16:9');

// Обработка событий HDMI для MAG
if (App.device.getDeviceKind() === 'mag') {
    App.device.standBy = false;
    App.device.standByTimer = null;
    App.device.onEvent = function(code) {
        switch (code) {
            default:
                break;
            case 0x20: // HDMI connected
                if (Helper.toInt(App.settings.getCustomSetting('handle-hdmi-events')) == 1) {
                    clearTimeout(App.device.standByTimer);
                    if (App.device.standBy) {
                        App.device.stb.StandBy(false);
                        App.device.standBy = false;
                    }
                }
                break;
            case 0x21: // HDMI disconnected
                if (Helper.toInt(App.settings.getCustomSetting('handle-hdmi-events')) == 1) {
                    clearTimeout(App.device.standByTimer);
                    App.device.standByTimer = setTimeout(function () {
                        if (!App.device.standBy) {
                            if (App.player.isActive()) {
                                App.player.stop();
                                App.display.showScreen('tvchannels');
                            }
                            App.device.stb.StandBy(true);
                            App.device.standBy = true;
                        }
                    }, 60 * 5 * 1000);
                }
                break;
        }
    }
}
};

OnAccountLoginSuccessful = function()
{
    // Изменение формата отображения оставшихся дней активации на dd/mm/yyyy
    var value = App.data.getActivationDaysLeft();
    if (parseInt(value) <= 0) {
        value = App.lang.get('auto-renewal');
    } else {
        var d = new Date(new Date().getTime()+(parseInt(value)*24*60*60*1000));
        var dd = d.getDate();
        var mm = d.getMonth() + 1;
        var y = d.getFullYear();
        value = dd + '/' + mm + '/' + y;
    }
    Helper.setHtml('info-menu-activation-days-left', value);
};
```

## 6.5 6.5 Кастомизация стилей оформления портала

Smarty позволяет для каждого устройства задать внешний css-файл для кастомного оформления портала, например, установить своё фоновое изображение, сменить цвет шрифта или фокуса.

Для этого необходимо открыть в панели администрирования «Общие настройки» -> «Настройки STB и приложений» -> <устройство для настройки> и прописать в поле «Внешний CSS» путь до файла с новыми стилями. Как правило, данный файл располагают на том же веб-сервере, что и портал.

Для создания файла с внешними стилями достаточно с помощью отладчика любого браузера проследить какие классы и идентификаторы отвечают за тот или иной элемент экрана в портале, после чего перезаписать/дописать нужные свойства к ним. Ниже представлены самые часто встречающиеся примеры кастомизации для шаблонов `futuristic` и `impuls`.

Пример внешнего css-файла для кастомизации шаблона `futuristic`.

```

/* Установка кастомного фонового изображения */
.screen, .android_stb .screen {
    background: url('example-bg.jpg') no-repeat;
}

/* Установка кастомного фонового изображения для верхней панели с лого оператора */
#statusbar-screen {
    background: transparent url('statusbar-bg.png') no-repeat;
}

/* Установка кастомного изображения для фокуса главного меню */
div#main-menu-selection {
    background: transparent url('example-selection-bg.png') no-repeat center 0px;
}

/* Установка кастомного шрифта */
@font-face {
    font-family: 'Nunito';
    src: url('../fonts/Nunito/nunito-v12-latin_cyrillic.eot');
    src: url('../fonts/Nunito/nunito-v12-latin_cyrillic.eot?#iefix') format('embedded-opentype'),
    url('../fonts/Nunito/nunito-v12-latin_cyrillic.woff2') format('woff2'),
    url('../fonts/Nunito/nunito-v12-latin_cyrillic.woff') format('woff'),
    url('../fonts/Nunito/nunito-v12-latin_cyrillic.ttf') format('truetype');
    font-weight: normal;
    font-style: normal;
}

body {
    font-family: "Nunito", "HelveticaNeue-Light", "Helvetica Neue Light", "Helvetica Neue",
    ↪Helvetica, Arial, "Lucida Grande", sans-serif;
}

```

Пример внешнего css-файла для кастомизации шаблона `impuls`.

```

/* Установка кастомного ладера */
.screen-container.loader, #player-mode-icon.loader, #loading-bar {
    background: transparent url("loader-example.gif") center center no-repeat;
}

```

(continues on next page)

(продолжение с предыдущей страницы)

```
}

#firstloading-loader {
    background: url('loader-example.gif') no-repeat;
    height: 65px;
    width: 120px;
}

.loader#video-actions-panel {
    background: transparent url("loader-example.gif") center left no-repeat;
    height: 120px;
}

/* Установка кастомного фонового изображения */

.screen, body, .play .screen, .png-transparency .screen, .play.png-transparency .screen,
.transparent.png-transparency .screen, .transparent .screen {
    background: url('example-bg.jpg') no-repeat;
}

body.play {
    background: transparent;
}

/* Установка кастомных координат для логотипа оператора */

.client-logo {
    margin-top: 75px !important;
    margin-right: 30px;
}
```

---

**Примечание:** Нужно учесть, что для корректной работы внешних стилей, необходимо, чтобы все дополнительные ресурсы (изображения, шрифты), используемые в них, были доступны и имели правильные пути.

---

---

**Примечание:** После установки обновлений шаблонов необходимо внимательно читать changelog и тщательно тестировать портал на предмет корректности работы внешнего CSS, так как внутренние css- и html-файлы могут претерпевать изменения, что так или иначе влияет на отображение кастомных стилей.

---



---

## А. Решение проблем и рекомендации

---

Компания «Майкроимпульс» оказывает услуги профессиональной технической поддержки по проектам, включая услуги по администрированию и настройке серверов Smarty, порталов, систем управления БД, а также осуществляет настройку репликации данных, балансировки нагрузки, кеширования, резервирования и прочее.

Подробную информацию о стоимости технической поддержки можно получить у своего менеджера.

Кроме того, задать свой вопрос или найти решение той или иной проблемы можно на официальном техническом сообществе <http://forum.micro.im>

### 7.1 А.1. Проблемы в работе сервера Middleware и сопутствующих системах и их решение

#### 7.1.1 Не отображается список телеканалов на устройстве абонента

**Возможные причины и решения:**

- Для абонента/аккаунта не установлены тарифные пакеты, содержащие необходимый набор каналов и стриминг-сервисов.
- В составе подключенных для абонента/аккаунта тарифных пакетов не найдено ни одного доступного стриминг-сервиса.
- Кеш списка телеканалов пуст, необходимо выполнить команду `cache_channel_list`, см. в *Кеширование списка телеканалов*.
- После установки обновления не выполнена команда `flushall`, что привело к порче кеша. См. в *Установка обновлений Smarty*.

### 7.1.2 В работе приложения у абонента возникают ошибки и/или при включении приставки возникает ошибка авторизации

#### Возможные причины и решения:

- После установки обновления не выполнена команда **flushall**, что привело к порче кеша. См. в *Установка обновлений Smarty*.
- Если используется единственный сервер Redis и в директории для сохранения дампа закончилось место, то это может приводить к ошибкам «*MISCONF Redis is configured to save RDB snapshots, but is currently not able to persist on disk*». Необходимо освободить свободное место и отключить опцию *stop-writes-on-bgsave-error*, см. в *Настройка кеширования*.

### 7.1.3 В телепрограмме на устройстве абонента не отображаются значки доступности записи телепередач

**Проблема:** при включенном и настроенном сервисе PVR абоненту недоступно включение передач из записи.

#### Возможные причины и решения:

- Для абонента/аккаунта не установлен тарифный пакет, содержащий соответствующий и доступный стриминг-сервис.
- Созданы технические работы, затрагивающие работу сервиса, см. в *Технические работы*.

### 7.1.4 Высокая нагрузка на CPU на сервере MongoDB

**Проблема:** потребление CPU на сервере MongoDB близко к 100% по всем ядрам, недоступны отчеты телесмотра.

#### Возможные причины и решения:

- Сервер MongoDB запущен без поддержки NUMA (такое также может быть при использовании виртуализации). Решение: <https://docs.mongodb.com/manual/administration/production-notes/#mongodb-and-numa-hardware>
- В процессе установки Smarty и настройки сервера MongoDB не были добавлены индексы. Решение: запустить команду *migrate*, которая добавляет нужные индексы в настроенную БД MongoDB, см. в *Установка обновлений Smarty*.

### 7.1.5 В ходе установки обновления Smarty возник всплеск нагрузки на сервер Middleware и СУБД

#### Возможные причины и решения:

- Если в момент обновления планировщиком была запущена команда импорта EPG, то попытка выполнить команду *flushall* может привести к конфликту инвалидации объектов, т.к. один процесс их создает, а другой инвалидирует. Это может привести к массовой инвалидации объектов в кеше. Для предотвращения такой ситуации при установке обновления необходимо завершать фоновые команды *epg\_import* и *cache\_channel\_list*. См. в *Установка обновлений Smarty*.

### 7.1.6 При массовом количестве абонентов возникает повышенная нагрузка на CPU и она не уменьшается

#### Возможные причины и решения:

- Недостаточно воркеров uwsgi или nginx. Необходимо провести оптимальную настройку сервера приложений и веб-сервера согласно общим рекомендациям под высокую нагрузку.
- Недостаточно оперативной памяти или CPU. См. в Системные требования. Временным решением проблемы может быть включение кеша nginx, для этого необходимо для наиболее частых запросов задействовать обработчик *@cached* в конфиге nginx, например:

```
location /tvmiddleware/api/channel/list/ {
    try_files $uri @cached;
}
location /tvmiddleware/api/program/list/ {
    try_files $uri @cached;
}
```

### 7.1.7 Периодически возникает всплеск нагрузки на сеть на сервере Middleware

#### Возможные причины и решения:

- Включена опция «*Включить автообновление данных без перезагрузки устройства*» в настройках устройства (см. в *Настройки STB и виджетов*). Если данная нагрузка нежелательна, то необходимо отключить опцию. Тогда полный список телеканалов и EPG не будет осуществляться.
- При некорректной работе сети или использовании виртуализации может возникать ситуация задерживания пакетов в очередях, что при освобождении очереди может привести к всплеску запросов, и как следствие - к всплескам трафика. Необходимо устранить сетевые проблемы и обеспечить быструю передачу данных.

### 7.1.8 Некоторые страницы панели администратора открываются с существенной задержкой, также возникают задержки в работе устройств

#### Возможные причины и решения:

- Проверьте доступность сервера СУБД для сервера Smarty. Необходимо обеспечить минимальное время отклика для быстрой работы системы. Также может помочь отключение DNS resolving на сервере БД, например для MySQL: <http://pe-kay.blogspot.ru/2011/08/problem-of-high-number-of.html>

### 7.1.9 Сервер приложений uwsgi не загружается, Smarty недоступна

#### Возможные причины и решения:

- Нет прав на запись для пользователя www-data в директорию /var/log/microimpuls или /var/log/microimpuls/smarty. Необходимо разрешить запись в эти директории.
- Другая причина - см. в логи /var/log/uwsgi/.

### 7.1.10 Команда импорта EPG `epg_import` останавливается, EPG не импортируется из панели администратора

Возможные причины и решения:

- Нет прав на запись в директорию `/usr/share/nginx/html/microimpuls/smarty/media`. Необходимо назначить пользователя и группу `www-data` на директорию `smarty`, выполнив команду: `chown -R www-data:www-data /usr/share/nginx/html/microimpuls`.
- Другая причина - см. в логи `/var/log/microimpuls/smarty/smarty_epg.log`.

### 7.1.11 В приложении абонента не отображаются иконки телеканалов и картинки передач

Возможные причины и решения:

- Некорректно установлено значение опции `MEDIA_BASE_URL` (см. в *Описание основных параметров*), либо некорректно настроен `nginx`. Для отладки необходимо открыть портал в браузере и с помощью инструментов разработчика (например, Firebug) отследить запросы к картинкам.

### 7.1.12 В приложении абонента спустя некоторое время сбиваются часы и EPG

Возможные причины и решения:

- Если на устройстве используется синхронизация времени с NTP, то при разрыве соединения с NTP-серверов время может испортиться. Необходимо обеспечить корректную работу и доступность NTP-сервера. Время на сервере `Middleware` также должно быть синхронизировано со временем на серверах `PVR`.

### 7.1.13 Не работает опрос серверов `MicroTS`, возникают ошибки при взаимодействии `Smarty` с видео-серверами `Microimpuls`

**Проблема:** данные мониторинга не собираются, не отображается мониторинг транскодирования потоков, заданий записи и др.

Возможные причины и решения:

- Порт `JSON-RPC API` соответствующих сервисов недоступен для сервера `Smarty`. Необходимо обеспечить их доступность.
- Установлена неверная версия пакета `python-jsonrpcserver`. Необходимо установить пакет из репозитория `Microimpuls`, см. в *Установка Smarty и модулей*.

### 7.1.14 Стороннее приложение, использующее `TVMiddleware API`, не может получить доступ с другого домена из-за политики `CORS`

Возможные причины и решения:

- В файле конфигурации `Smarty` необходимо прописать исключение `CORS` для этого домена в опции `CORS_ORIGIN_WHITELIST`, тогда в заголовках `HTTP` для запросов с этого домена будут выданы необходимые разрешения. Пример: `CORS_ORIGIN_WHITELIST = („example.com“, )`

### 7.1.15 Не загружаются обложки и описание фильма с сервиса Кинопоиск

#### Возможные причины и решения:

- При частом использовании функции скачивания информации с сервиса Кинопоиск в панели администратора IP-адрес сервера Smarty может быть заблокирован системой защиты от ботов Кинопоиска. Для решения этой проблемы необходимо снизить активность запросов либо обратиться в службу технической поддержки сервиса Кинопоиск.

### 7.1.16 Медленно обрабатываются запросы к Billing API, медленно сохраняются изменения в панели администратора при редактировании тяжелых объектов

#### Возможные причины и решения:

- При длительной работе Smarty накапливает много данных в кэше для быстрой работы приложений. При редактировании данных администратором, при обработке API-запросов от биллинга может возникнуть необходимость инвалидации данных по большому числу связанных объектов, что вызывает длительную работу сервера для обработки такого запроса. Чтобы данные операции производились в фоновом режиме и не тормозили работу необходимо включить поддержку потоков, установив опцию `enable-threads` в значение `true` в файле конфигурации `uwsgi`, после чего перезапустить `uwsgi`.

## 7.2 А.2. Рекомендации

### 7.2.1 Рекомендуемые параметры ядра

Изменения нужно вносить в файл `/etc/sysctl.conf`:

```
kernel.shmmax = 2473822720
kernel.shmall = 4097152000
net.core.rmem_default = 8388608
net.core.rmem_max = 16777216
net.core.wmem_default = 8388608
net.core.wmem_max = 16777216
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_tw_recycle = 0
net.ipv4.tcp_tw_reuse = 0
net.ipv4.tcp_keepalive_time = 10
net.ipv4.tcp_fin_timeout = 5
```

Затем выполнить команду для применения изменений:

```
sysctl -p
```



---

## В. Дополнительные материалы

---

### 8.1 В.1. Установка драйвера cx\_Oracle на Debian и пример настройки подключения Smarty к Oracle

1. Установить пакет `oracle-instantclient11.2-basic`.

При установке на Debian необходимо сконвертировать rpm-пакет, предоставляемый Oracle, в deb-пакет с помощью утилиты `alien`:

```
alien oracle-instantclient11.2-basic-11.2.0.4.0-1.x86_64.rpm
dpkg -i oracle-instantclient11.2-basic_11.2.0.4.0-2_amd64.deb
```

2. Установить пакет `oracle-instantclient11.2-devel`:

```
alien oracle-instantclient11.2-devel-11.2.0.4.0-1.x86_64.rpm
dpkg -i oracle-instantclient11.2-devel_11.2.0.4.0-2_amd64.deb
```

3. Установить пакет `oracle-instantclient11.2-sqlplus`:

```
alien oracle-instantclient11.2-sqlplus-11.2.0.4.0-1.x86_64.rpm
dpkg -i oracle-instantclient11.2-sqlplus_11.2.0.4.0-2_amd64.deb
```

4. Установить python-модуль `cx_Oracle`:

```
pip install cx_Oracle
```

5. Установить `libaiol`:

```
apt-get install libaiol libaio-dev
```

#### 8.1.1 В.1.1. Настройка подключения Smarty к Oracle

В файл конфигурации Smarty в секции настроек подключения к БД необходимо прописать:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.oracle',
        'NAME': "smarty",
        'USER': "smarty", # Имя схемы/пользователя
        'PASSWORD': "password", # Пароль пользователя
        'HOST': "10.0.0.10", # Адрес сервера Oracle
        'PORT': '1521',
        'OPTIONS': {
            'threaded': True,
            'use_returning_into': False,
        },
        'CONN_MAX_AGE': 600,
    }
}
```

### 8.1.2 В.2. Настройка подключения Smarty к PostgreSQL

В файл конфигурации Smarty в секции настроек подключения к БД необходимо прописать:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'smarty',
        'HOST': 'localhost',
    }
}
```

Параметр Host является обязательным. Также для PostgreSQL появляется новая зависимость: psycopg2. Устанавливается через pip:

```
pip install psycopg2-binary
```