GS LABS
GS GROUP

Публичный - L0

DRE Advanced Media Platform SCRAMBLER

Руководство администратора

Индекс	2004-SCRAMBLER-AG
Секретность	Публичный - L0
Ревизия	1.0
Статус	Согласован
Подразделение	ДПРСУД
Компания	GS Labs

Содержание

1. Аннотация

Документ содержит настройки компонентов продукта DRE Advanced Media Platform SCRAMBLER и предназначен для сотрудников отдела мониторинга и инсталляции, а также для других технических специалистов, в обязанности которых входит настройка системы и поддержание её работоспособности.

2. Справочная документация

№	Наименование документа	Индекс документа (если есть)
1	DRE Advanced Media Platform SCRAMBLER. Руководство по установке	2004-SCRAMBLER-IG

3. Термины и сокращения

Термин	Определение
GS DRM	Система управления цифровыми правами DREPLUS (DRM DREPLUS).
Live	Вещание телеканала в режиме реального времени.
MDS	DRE Advanced Media Platform META DATA SERVER - сервер метаданных.
Scrambler Instance	Один экземпляр скремблера. Данный компонент предназначен для скремблирования файла или live потока.
Scrambler Manager	Компонент доступный для внешних пользователей. Предназначен для получения от пользователя задания на скремблирование, выбора свободного Scrambler Instance и отправки задания данному Scrambler Instance.
Скремблер	Устройство, выполняющее транскодирование и шифрование видеопотока по условиям обрабатываемой задачи.
Транскодирование	Преобразование видеофайла из одного цифрового формата в другой (преобразование формата файла, видео и аудио).

Сокращение	Расшифровка
CDN	Content Delivery Networks
DRM	Digital Rights Management
VOD	Video on Demand

4. Введение

4.1. Назначение

Программа DRE Advanced Media Platform SCRAMBLER (далее - SCRAMBLER или Система) предназначена для подготовки Live (прямой эфир) и VOD (видео по запросу) контента для последующего вещания в OTT-платформах. Реализует функции транскодирования, пакетирования и шифрования. При транскодировании обеспечивает возможность конвертации контента в разные уровни качества, соответствующие пропускной способности канала связи. При этом пакетирование выполняет в форматы HLS или DASH, а шифрование осуществляет с поддержкой технологий GS DRM, Google Widevine или Apple FairPlay. Поддерживает как последовательную обработку файлов (VOD контента) и видеопотоков (Live контента), так и задание их списком для пакетной обработки. Программа обеспечивает возможность использования ресурсов как одного, так и нескольких физических или виртуальных серверов (распределенная обработка), а также ускорение процесса транскодирования с помощью видеокарт, поддерживающих технологию Nvidia Cuda. Тип ЭВМ IBM PC – совместимый ПК; ОС Linux.

5. Обслуживание системы

Обслуживание SCRAMBLER заключается в выполнении следующих основных действий:

- Изменение настроек компонентов системы (**при необходимости**). После изменения настроек необходимо перезапустить соответствующий контейнер, чтобы изменения вступили в силу.
- Устранение ошибок в работе SCRAMBLER на основе его логов.

Необходимо отметить, что при установке SCRAMBLER с нуля необходимо настроить систему под нужды конкретного Заказчика. Начальные установки являются работоспособными, однако их стоит использовать для тестирования, обучения или примера.

5.1. Изменение настроек компонентов системы

5.1.1. Основные действия

В общем случае необходимо выполнить следующее:

1. Выполнить подготовительные действия:
    a. При установке SCRAMBLER "с нуля": скопировать содержимое папки проекта по развертыванию SCRAMBLER на машину, которая будет использоваться в качестве хоста для контейнеров, перейти в выбранную папку, предоставить пользователям права на чтение, изменение и запуск содержимого этой папки.
    b. При обновлении/перенастройке SCRAMBLER: остановить контейнеры Scrambler Instance, затем остановить контейнеры ScramblerManager, Nginx и т.д. (используется команда *docker-compose down*).
2. Отредактировать конфигурационные файлы компонентов (файлы docker-compose.yml).
3. Запустить docker-контейнеры с компонентами системы:
    a. Перейти в папку, где лежит файл ***docker-compose.yml*** для развертывания системы.
    b. Запустить скрипт установки:

    ```
    bash install_full.sh local_ip_addr
    ```

    , где local_ip_addr - адрес текущей машины.
    c. Дождаться окончания операции. В случае успеха стек контейнеров будет успешно поднят. Для проверки можно:
        i. выполнить команду:

        ```
        docker ps -a
        ```

        ii. в появившемся списке должны быть контейнеры scrambler_manager, scrambler_instance и др., развернутые с помощью docker-compose.

⚠ Установка и запуск docker-контейнеров, а также другие возможные действия (установка и запуск дополнительного Scrambler instance, запуск RTSP сервера + HAProxy) описаны в документе "DRE Advanced Media Platform SCRAMBLER. Руководство по установке" [1].

5.1.2. Конфигурационные файлы

Конфигурационный файл compose содержит настройки, определяющие перечень контейнеров и параметры их запуска.

Описание возможных параметров можно посмотреть здесь:

- version 2: https://docs.docker.com/compose/compose-file/compose-file-v2/

В SCRAMBLER используются следующие конфигурационные файлы:

1. docker-compose.yml для развертывания ScramblerManager, Publisher(nginx).
2. docker-compose.yml для развертывания сервиса ScramblerIntance (лежит в папке */scrambler_instance_deploy /scrambler_instance_deploy*).
3. docker-compose.yml сервиса rtsp_server. Находится в папке *rtsp_server_deploy*.

Формат yml/yaml имеет специфические требования к синтаксису (например, в yml/yaml файле нельзя использовать табуляцию). В связи с этим настоятельно рекомендуется проверить файл на корректность: https://codebeautify.org/yaml-validator

Использованные параметры описаны в таблице ниже.

Параметр	Описание
version:	Версия файла. В разных версиях файлов docker-compose некоторые опции могут быть доступны, отсутствовать, задаваться по-другому либо не работать с определенными версиями файла.
services:	**Перечень docker-контейнеров, работой которых управляет docker-compose. Параметры каждого контейнера задаются в отдельной секции.**
scrambler_manager:	**Настройки Scrambler Manager:**
image:	Образ для контейнера. Можно указать либо репозиторий / тег, либо идентификатор частичного изображения. Образы передаются вместе с инсталляционными файлами. **Примечание**. Если образ не существует, Compose пытается его вытащить (docker pull), если только вы не указали сборку, и в этом случае он строит её с использованием указанных опций и помечает её указанным тегом.

container_name:	Имя настраиваемого контейнера. В результате контейнер будет иметь указанное имя, в противном случае docker-compose генерирует имя по умолчанию. **Примечание**. Поскольку имена Docker-контейнеров должны быть уникальными, то в случае задания этого параметра вы не сможете масштабировать службу за пределами одного контейнера. Попытка сделать это приведет к ошибке.
ports:	Проброс портов. Либо укажите оба порта (*HOST: CONTAINER*), либо просто порт контейнера (в этом случае выбирается случайный порт на host-машине). **Примечание**. При сопоставлении портов в формате *HOST: CONTAINER* вы можете столкнуться с ошибочными результатами при использовании для контейнера порта ниже 60, поскольку YAML будет анализировать номера в формате *xx: yy* как sexagesimal (base 60). По этой причине рекомендуется явно указывать сопоставляемые порты как строки.
volumes:	Пути монтирования или именованные тома, опционально указывается путь на главной машине (*HOST: CONTAINER*) или режим доступа (*HOST: CONTAINER: ro*). Для файлов 2 версии тома должны быть указаны с помощью секции ***volumes*** верхнего уровня. Можно смонтировать относительный путь на хосте, который будет определяться относительно каталога, в котором используется конфигурационный файл Compose. Относительные пути всегда должны начинаться с **.** или **...**
environment:	Переменные среды:
FILES_EXPIRE_TIME	Время жизни файлов на файловом сервере в часах. По умолчанию 24 часа. Допустимы только целочисленные значения (любые буквы, знаки препинания и вещественные значения не подходят).
DELETE_OLD_TASKS	Удалять задачи по истечении FILES_EXPIRE_TIME. По умолчанию 1.
OUTPUT_DIR	Папка, в которую будут помещаться результаты обработки.
	Параметры базы данных SCRAMBLER DB, которая необходима для хранения задач сервиса ScramblerManager.
DATABASE_HOST	IP-адрес или имя хоста SCRAMBLER DB.
DATABASE_PORT	Номер порта для связи со SCRAMBLER DB.

DATABASE_USER	Имя пользователя SCRAMBLER DB.
DATABASE_PASSWORD	Пароль пользователя SCRAMBLER DB.
DATABASE_DBNAME	Имя для базы SCRAMBLER DB.
	Параметры http-сервера Scrambler Manager:
HTTP_HOST	Внутренний host Scrambler Manager.
HTTP_PORT	Внутренний port Scrambler Manager (внешний порт указан в секции ports).
HTTP_WRITE_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение (при обращении на запись данных).
HTTP_READ_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение (при обращении на чтение данных).
	Параметры логирования:
LOGGER_LEVEL	Степень логирования событий. Возможные значения: • 0 - trace, • 1 - debug, • 2 - info (значение по умолчанию), • 3 - warning, • 4 - error, • 5 - fatal. Для тестирования рекомендуется 1 или 0.
	Параметры http для системы публикации контента:
PUBLISHER_HTTP_SERVER	Адрес и порт http-сервера, который будет указываться системе публикации контента для забора результирующих файлов.
	Настройки взаимодействия с prometheus:
PROMETHEUS_ADDRESS	HTTP адрес сервера prometheus внутри контейнера.
PROMETHEUS_WRITE_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение (при обращении на запись данных).
PROMETHEUS_READ_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение (при обращении на чтение данных).
DISCOVERY_HEALTH_CHECK_TIMEOUT	Таймаут (в сек.), по которому scrambler_manager пингует известные ему scrambler_instance на предмет работоспособности.

CHECK_UNIQUE_RESOURCE_ID	Флаг, регулируеющий проверку уникальности resource_id в рамках одной задачи для разных профилей шифрования.
	Настройки взаимодействия с MDS:
MDS_ADDRESS	Адрес сервера авторизации MDS. Необходим для проверки токена при установлении websocket соединения с вебом.
MDS_TIMEOUT	Таймаут (в сек.) запросов к MDS.
	Настройки взаимодействия с rtsp:
RTSP_SERVER_HOST	Хост RTSP сервера, на приемники будут отправлять поток с камер видеонаблюдения. Здесь необходимо указать адрес haproxy, который балансирует rtsp соединения между rtsp серверами.
restart:	Условие перезапуска контейнера.
scrambler_instance:	**Настройки Scrambler Instance:**
image:	Образ для контейнера. Можно указать либо репозиторий / тег, либо идентификатор частичного изображения. Образы передаются вместе с инсталляционными файлами. **Примечание**. Если изображение не существует, Compose пытается его вытащить (docker pull), если только вы не указали сборку, и в этом случае он строит её с использованием указанных опций и помечает её указанным тегом.
container_name:	Имя настраиваемого контейнера. В результате контейнер будет иметь указанное имя, в противном случае docker-compose генерирует имя по умолчанию. **Примечание**. Поскольку имена Docker-контейнеров должны быть уникальными, то в случае задания этого параметра вы не сможете масштабировать службу за пределами одного контейнера. Попытка сделать это приведет к ошибке.
network_mode	Способ использования сети контейнером scrambler_instance. Контейнер должен использовать сеть host-машины. Значение не менять.

volumes:	Пути монтирования или именованные тома, опционально указывается путь на главной машине (*HOST: CONTAINER*) или режим доступа (*HOST: CONTAINER: ro*). Для файлов 2 версии тома должны быть указаны с помощью секции ***volumes*** верхнего уровня. Можно смонтировать относительный путь на хосте, который будет определяться относительно каталога, в котором используется конфигурационный файл Compose. Относительные пути всегда должны начинаться с **.** или **...**
environment:	Переменные среды:
instance_type	Тип скремблера. Может принимать значения: cpu/gpu. По умолчанию cpu. (тип gpu имеет смысл использовать только на машине с драйверами nvidia)
transcoding_last_transcode_files	Папка в контейнере scrambler_instance, в которой лежат последние файлы транскодирования. **Не менять!**
manager_host manager_available	Scrambler Instance может работать как в связке со Scrambler Manager, так и без него. Чтобы работать отдельно от ScramberManager, нужно установить переменную manager_available: 0. В противном случае нужно указать manager_host и manager_available: 1. **ВНИМАНИЕ!** Параметр manager_host должен быть указан в формате *hostname:port*
output_dir	Директория, в которой будут храниться результаты работы скремблера. Обратите внимание, что это путь до директории внутри контейнера, для доступа к файлам необходимо сделать docker volume для этой директории.
	Параметры http сервера Scrambler Instance:
http_address	IP-адрес Scrambler Instance.
http_port	Порт Scrambler Instance.
http_external_address	Параметры для отправки запросов на Scrambler Instance (hostname:порт).
kms_host	Адрес сервиса KMS.
	Настройки шифрования:
encryptor_Widevine_provider	Имя провайдера Widevine.
encryptor_GsDrm_hls_time	Продолжительность блока контента (чанка), сек.
encryptor_GsDrm_key_rotation_time	Период смены ключей шифрования GsDrm, сек.

	Параметры http-сервера для системы публикации контента:
publisher_http_server	Адрес и порт http-сервера, который будет указываться системе публикации контента для забора результирующих файлов.
publisher_call_back_timeout	Время ожидания ответа на callback-запрос к системе публикации контента, сек.
	Настройки взаимодействия с утилитой zbx-statsd (через которую осуществляется взаимодействие с Zabbix):
live_time_shift_buffer_depth	Период хранения чанков для live (в секундах). То есть если задано 100сек и время чанка 5сек, то будут храниться последние 20 чанков.
transcoding_overrun_nonfatal	Значение флага overrun_fatal при обработке live контента с помощью ffmpeg. Если задан 1, то передается overrun_fatal=1, иначе флаг не передается в ffmpeg.
thumbnails_enabled	1 - thumbnails генерируются для VOD контента. 0 - thumbnails не генерируются.
thumbnails_interval	Интервал генерации thumbnails в сек.
thumbnails_resolution	Разрешение thumbnails.
live_timeout	Таймаут в секундах для ожидания поступления live потока
rtsp_transport	Параметр rtsp_transport, который использует ffmpeg для чтения rtsp потока
transcoding_threads	Значение опции ffmpeg -threads. По факту означает количество потоков процессора, которые будут использоваться при транскодировании VOD.
transcoding_gpu_number	Номер видеокарты, которую scrambler_instance будет использовать при транскодировании на GPU.
non_transcoding_target_bitrate	Целевой битрейт выходного потока для live задач без транскодирования
restart:	Условие перезапуска контейнера.
nginx:	**Настройки nginx для доступа к результатам обработки через http:**
	ВНИМАНИЕ! Если установка производится с помощью установочного скрипта (см. ниже), значения параметров менять не требуется.

image:	Образ для контейнера. Можно указать либо репозиторий / тег, либо идентификатор частичного изображения. Образы передаются вместе с инсталляционными файлами. **Примечание**. Если изображение не существует, Compose пытается его вытащить (docker pull), если только вы не указали сборку, и в этом случае он строит её с использованием указанных опций и помечает её указанным тегом.
container_name:	Имя настраиваемого контейнера. В результате контейнер будет иметь указанное имя, в противном случае docker-compose генерирует имя по умолчанию. **Примечание**. Поскольку имена Docker-контейнеров должны быть уникальными, то в случае задания этого параметра вы не сможете масштабировать службу за пределами одного контейнера. Попытка сделать это приведет к ошибке.
ports:	Проброс портов. Либо укажите оба порта (*HOST: CONTAINER*), либо просто порт контейнера (в этом случае выбирается случайный порт на host-машине). **Примечание**. При сопоставлении портов в формате *HOST: CONTAINER* вы можете столкнуться с ошибочными результатами при использовании для контейнера порта ниже 60, поскольку YAML будет анализировать номера в формате *xx: yy* как sexagesimal (base 60). По этой причине рекомендуется явно указывать сопоставляемые порты как строки.
volumes:	Пути монтирования или именованные тома, опционально указывается путь на главной машине (*HOST: CONTAINER*) или режим доступа (*HOST: CONTAINER: ro*). Для файлов 2 версии тома должны быть указаны с помощью секции ***volumes*** верхнего уровня. Можно смонтировать относительный путь на хосте, который будет определяться относительно каталога, в котором используется конфигурационный файл Compose. Относительные пути всегда должны начинаться с **.** или **...** **Примечание**. Контейнер nginx отдает файлы по http из примонтированной с помощью cifs директории, поэтому на хосте монтируется cifs директория, а в nginx она пробрасывается через volume.
command: "nginx -g 'daemon off;'"	Переопределение дефолтной команды для исполняемого контейнера. В данном случае это команда запуска веб-сервера nginx НЕ в режиме демона (фонового процесса).

cap_add: - SYS_ADMIN	Добавление Linux возможностей для контейнера. Подробнее см. https://docs.docker.com/engine/reference/run/#runtime-privilege-and-linux-capabilities. Значение SYS_ADMIN позволяет выполнять ряд операций по системному администрированию. **Примечание**. Совместное использование *cap-add* и *devices: /dev /fuse* позволяет использовать примонтированный директорий.
devices: - /dev/fuse	Проброс устройств. Аналогичен параметру *--device* в команде *docker run*. **Примечание**. Совместное использование *cap-add* и *devices: /dev /fuse* позволяет использовать примонтированный директорий.
security_opt: ["apparmor:unconfined"]	В данном случае - отключает ограничение apparmor для контейнера. **Примечание**. Используется совместно с *cap-add* и *devices: /dev /fuse* - без apparmor не получится использовать примонтированный директорий.
restart:	Условие перезапуска контейнера.
fas	**Настройки сервиса fas:**
EXTERNAL_ADDRESS	Адрес, который fas будет прописывать, как URl для получения ключа контента.
PROMETHEUS_ADDRESS	HTTP адрес сервера prometheus внутри контейнера.
PROMETHEUS_WRITE_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение (при обращении на запись данных).
PROMETHEUS_READ_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение (при обращении на чтение данных).
HTTP_ADDRESS	Внутренний адрес сервиса.
HTTP_WRITE_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение (при обращении на запись данных).
HTTP_READ_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение (при обращении на чтение данных).
LOGGER_LEVEL	Уровень логирования.
SECURE_LINK_ENABLED	Доступность режима secure link. По умолчанию 1.
SECURE_LINK_SECRET	Secret ключ, использующийся при формировании secure link.

SECURE_LINK_EXPIRES	Время жизни secure link.
ENCRYPTION_KEY	32 байтный ключ, использующийся для шифрования ключей, выдаваемых сервисом.
rtsp_server	**Настройки сервиса rtsp_server:**
RTSP_RTSPADDRESS	Внутренний порт, на котором сервис слушает rtsp соединения.
RTSP_EXTERNALADDRESS	Внешний адрес rtsp сервера, который он присылает в scrambler_manager при поступлении потока.
RTSP_SCRAMBLERMANAGERADDRESS	Адрес сервиса scrambler_manager.
RTSP_SCRAMBLERMANAGERTIMEOUT	Таймаут при обращениях к scrambler_manager.
RTSP_LOGLEVEL	Уровень логирования сервиса.
RTSP_METRICS	yes - выдавать метрики по порту 9998, no - метрики выдаваться не будут.

6. Ведение логов

6.1. Режимы ведения логов

Компоненты SCRAMBLER ведут логи, информация из которых может быть использована для решения возникающих проблем. Логи могут вестись с разной степенью подробности.

Уровни логирования настраиваются в **docker-compose.yml**, с помощью параметра LOGGER_LEVEL (для rtsp - с помощью параметра RTSP_LOGLEVEL).

Доступны следующие режимы ведения логов:

- 0 - trace: подробная информация по любым действиям;
- 1 - debug: конфигурационные данные (при запуске системы), другая информация, необходимая для отладки, + сообщения уровня Info;
- 2 - info (значение по умолчанию): базовая информация (сообщения о запуске, работе, выключении системы) + сообщения уровня Warning;
- 3 - warning: системные предупреждения + сообщения уровня Error;
- 4 - error: все ошибки, возникающие в процессе работы, в том числе ошибки уровня Fatal;
- 5 - fatal: критические ошибки, приводящие к сбоям системы.

6.2. Формат записей

Все логи представлены в формате JSON. Каждое сообщение лога представлено в формате отдельной JSON-структуры с исчерпывающим набором полей.

В зависимости от уровня логирования изменяется набор данных внутри JSON либо добавляются новые элементы (записи лога).

6.3. Просмотр логов

Логи пишутся в std:out.

Для просмотра логов определенного сервиса можно использовать команду:

```
docker logs [OPTIONS] CONTAINER
```

где:

- [OPTIONS] - дополнительные параметры при выполнении команды. См. https://docs.docker.com/engine/reference/commandline/logs/
- CONTAINER - название docker-контейнера с сервисом.