VisionLabs
MACHINES CAN SEE

VisionLabs LUNA PLATFORM 5

Руководство администратора

v.5.92.0

Содержание

Глоссарий

Термин	Определение
Атрибуты	Базовые атрибуты и биометрический шаблон.
Базовые атрибуты	Возраст, пол и этническая принадлежность.
Бакет	Хранилище биометрических образцов с мета-данными, исходных изображений, объектов LP и результатов задач.
Биометрический образец	Нормализованное (центрированное и обрезанное) изображение, полученное после обнаружения лица или тела, предшествующее извлечению биометрического шаблона.
Биометрический шаблон	Набор данных в закрытом, двоичном формате, подготавливаемый системой распознавания на основе анализируемой характеристики.
Верификация	Сравнение двух фотоизображений лица с целью определения принадлежности одному и тому же человеку.
Детектор	Нейронная сеть, используемая для обнаружения либо лиц, либо тел, либо одновременно лиц и тел на исходном изображении.
Контрольные точки	Опорные точки на лице или теле, используемые алгоритмами распознавания для локализации лица или тела.
Кооперативный режим работы	Процесс получения исходных фотоизображений лиц или тел из потокового видео при осознанном взаимодействии человека с камерой.
Лица	Изменяемые объекты, содержащие информацию о лице человека.
Некооперативный режим работы	Процесс получения исходных фотоизображений лиц или тел из потокового видео, при этом человек может не взаимодействовать с камерой.
Ограничивающий прямоугольник	Прямоугольник, ограничивающий пространство изображения с обнаруженным лицом или телом.
Сравнение, матчинг	Операция сопоставления биометрических шаблонов, хранящихся в базе данных.

Термин	Определение
Параметры лиц	Характеристики лица (эмоции, параметры рта, положение головы и т.д.), определяемые на исходном изображения во время детекции.
Параметры тел	Характеристики тела (наличие рюкзака, головной убор, цвет одежды и т.д.), определяемые на исходном изображения во время детекции.
Параметры изображений	Характеристики изображения (ширина и высота, соотношение сторон, размер и т.д.), определяемые на исходном изображения во время детекции.
Перекрытие	Состояние объекта (глаз или рта), при котором он скрыт другим объектом.
Эстиматор	Нейронная сеть, используемая для оценивания определенного параметра лица или тела на исходном изображении.
Liveness	Программный способ, позволяющий подтвердить является ли человек на одном или нескольких изображениях «реальным» или мошенником, использующим поддельный идентификатор (распечатанное фото лица, видео, бумажную или 3D маску).
OneShotLiveness	Эстиматор, предназначенный для оценки Liveness.

Аббревиатура	Расшифровка
БО	Биометрический образец
БШ	Биометрический шаблон
БД, DB	База данных
LP	LUNA PLATFORM
MQ	Очередь сообщений
UI	Пользовательский интерфейс
GC	Сбор мусора
Accounts	LUNA PLATFORM Accounts
API	LUNA PLATFORM API
Faces	LUNA PLATFORM Faces
Image Store	LUNA PLATFORM Image Store
Matcher	LUNA PLATFORM Matcher
Events	LUNA PLATFORM Events
Sender	LUNA PLATFORM Sender
Remote SDK	LUNA PLATFORM Remote SDK
Handlers	LUNA PLATFORM Handlers
Python Matcher	LUNA PLATFORM Python Matcher
Video Manager	LUNA PLATFORM Video Manager
Video Agent	LUNA PLATFORM Video Agent
Streams Retranslator	LUNA PLATFORM Streams Retranslator
Python Matcher Proxy	LUNA PLATFORM Python Matcher Proxy
Backport 3	LUNA PLATFORM Backport 3
Backport 4	LUNA PLATFORM Backport 4
Admin	LUNA PLATFORM Admin
Configurator	LUNA PLATFORM Configurator
Tasks	LUNA PLATFORM Tasks
Licenses	LUNA PLATFORM Licenses
User Interface 3	LUNA PLATFORM User Interface 3

Аббревиатура	Расшифровка
User Interface 4	LUNA PLATFORM User Interface 4

1 Введение

LUNA PLATFORM — это автоматизированная система распознавания лиц и тел. LUNA PLATFORM предназначена для решения следующих задач:

- Обработка и анализ изображений:
  - обнаружение лиц и тел на фотографиях;
  - оценка базовых атрибутов (возраст, пол, расовая принадлежность) и параметров лиц, тел и изображений (эмоции, верхняя одежда, соотношение сторон изображения и др.);
  - проверка изображений на соответствие стандарту ISO/IEC 19794-5:2011 и по нестандартным условиям.
  - определение атак на биометрическое предъявление (проверка Liveness) и анализ изображений на предмет использования технологии Deepfake.
- Поиск схожих лиц в базе данных (1 к 1, 1 ко многим и M к N);
- Хранение получаемых биометрических шаблонов лиц/тел в базах данных;
- Создание списков для поиска;
- Сбор статистики;
- Гибкое управление запросами для соответствия требованиям обработки пользовательских данных.

> Некоторые из вышеперечисленных функций лицензируются отдельно. См. раздел «Информация о лицензии».

2 Общие сведения

Разделы в данной главе дают общее представление о работе LP, ее сервисах и создаваемых типах данных. Описания запросов, структур баз данных и другие технические детали в данной главе не приводятся.

2.1 Сервисы

LP состоит из нескольких сервисов. Сообщение между ними происходит посредством HTTP-запросов: сервис получает запрос и возвращает ответ. Некоторые сервисы также взаимодействуют друг с другом через Redis или веб-сокеты.

Информацию об архитектуре LUNA PLATFORM 5 можно найти в разделе «Взаимодействие сервисов».

Все сервисы можно разделить на **основные** и **дополнительные**. С помощью основных сервисов обеспечивается оптимальная работа LP. Использование всех основных сервисов включено по умолчанию в настройках сервиса API. При необходимости некоторые основные сервисы можно отключить (см. раздел «Отключаемые сервисы»).

У большинства сервисов имеется собственная база данных или файловое хранилище.

2.1.1 Основные сервисы

Сервис	Описание	База данных	Отключаемый
API	Основной интерфейс доступа для работы с LP. Получает запросы, распределяет задачи между другими сервисами LP.	-	Нет
Accounts	Создает аккаунты и управляет ими.	PostgreSQL/ Oracle	Нет
Remote SDK	Распознает лица и тела на изображениях, оценивает лица и тела по параметрам и создает биометрические образцы. Извлекает из биометрических образцов биометрические шаблоны. Извлекает базовые атрибуты изображений.	-	
Python Matcher	Выполняет операции сравнения биометрических шаблонов.	Нет	Нет

Сервис	Описание	База данных	Отключаемый
Faces	Создает лица, списки и атрибуты. Сохраняет эти объекты в базе данных. Позволяет другим сервисам получать требуемые данные из базы данных.	PostgreSQL/ Oracle, Redis	Да
Configurator	Хранит конфигурации всех сервисов в одном месте.	PostgreSQL/ Oracle	Нет
Licenses	Проверяет данные лицензии и возвращает информацию о них.	-	Нет
Handlers	Создает и хранит обработчики. Принимает запросы на детекцию, эстимацию и извлечение и перенаправляет их в сервис Remote SDK.	PostgreSQL/ Oracle	Да
Image Store	Хранит биометрические образцы, любые объекты, отчеты о длительном выполнении задач, создаваемые кластеры и дополнительные метаданные.	Local storage/ Amazon S3	Да
Events	Отвечает за сохранение событий в БД.	PostgreSQL	Да
Admin	Выполняет общие административные процедуры.	PostgreSQL/ Oracle	Да
Tasks	Выполняет длительные задачи, такие как Garbage collection, Additional extraction, Clustering и др.	PostgreSQL/ Oracle	Да
Tasks Worker	Выполняет внутреннюю работу сервиса Tasks.	-	Да
Sender	Отправляет уведомления о создаваемых событиях через веб-сокет.	Redis	Да

2.1.2 Дополнительные сервисы

Дополнительные сервисы предоставляют больше возможностей для работы системы. Запуск дополнительных сервисов не обязателен.

Сервис	Описание	База данных
Backport 3	Используется для обработки запросов LUNA PLATFORM 3 с использованием LUNA PLATFORM 5.	PostgreSQL/ Oracle
Backport 4	Используется для обработки запросов LUNA PLATFORM 4 с использованием LUNA PLATFORM 5.	-
User Interface 3	Пользовательский интерфейс, используемый для визуального представления возможностей, предоставляемых сервисом Backport 3. Он не включает в себя весь функционал LP 3.	Нет
User Interface 4	Пользовательский интерфейс, используемый для визуального представления возможностей, предоставляемых сервисом Backport 4. Он не включает в себя весь функционал LP 4.	-
Python Matcher Proxy	Управляет запросами сравнения и направляет их в Python Matcher или в плагины сравнения.	Нет
Lambda	Работает с пользовательскими модулями, имитирующими функционал отдельного сервиса.	PostgreSQL/ Oracle
Video Manager	Распределяет потоки между агентами с учетом полученной информации об аналитиках от агента.	PostgreSQL/ Oracle
Video Agent	Выполняет видеоаналитику подсчета количества людей.	PostgreSQL/ Oracle

Ниже представлена схема взаимодействия основных и дополнительных сервисов.

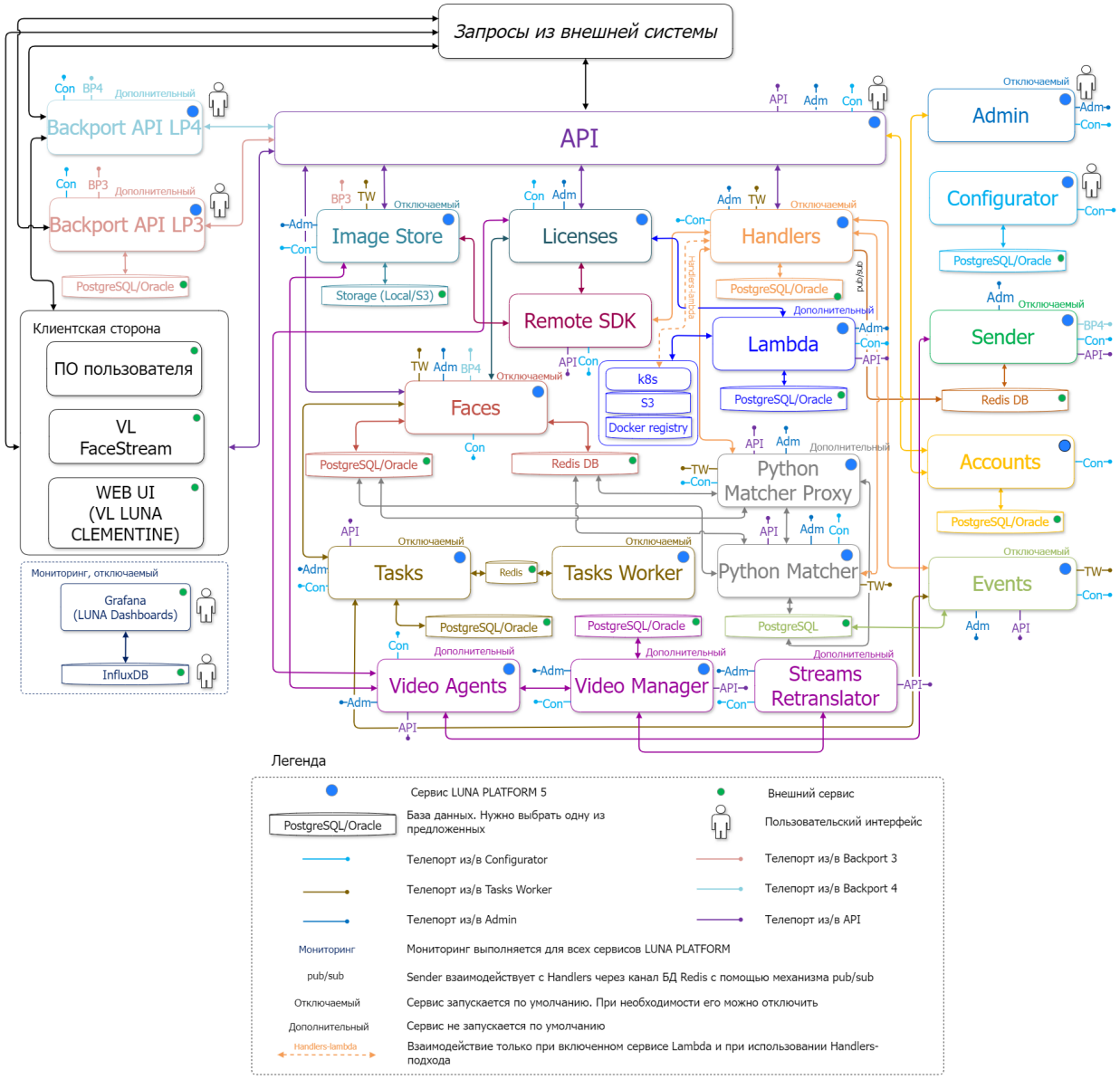


Рис. 1: Упрощенная схема взаимодействия основных и дополнительных сервисов

Данная схема не описывает линии связи между сервисами. Для полного описания взаимодействия основных сервисов см. раздел «Взаимодействие сервисов».

2.1.3 Сторонние сервисы

Существует несколько сторонних сервисов, обычно используемых с LP.

Сервис	Описание	Поддерживаемые сервисы
Балансировщик	Осуществляет балансировку запросов между сервисами LP, в случае если запущено несколько аналогичных сервисов. К примеру, можно балансировать запросы между сервисами API или между двумя контурами LP при масштабировании.	NGINX
Система мониторинга	Используется для сбора статистики о запросах и работе системы.	Supervisor
База данных мониторинга	База данных, используемая для хранения собранной информации по мониторингу системы.	InfluxDB
Визуализация мониторинга	Визуализация мониторинга представленная набором дашбордов. С ее помощью можно оценить общую нагрузку на систему или нагрузку на отдельные сервисы.	Grafana, Grafana Loki
Ротация логов	Все сервисы LP пишут логи, размер которых может очень сильно расти. Сервис распределения логов позволяет удалять старые файлы логов и освобождать место на диске.	Logrotate, etc.

> Описание этих сервисов не приведено в данном документе. См. соответствующую документацию, предоставляемую поставщиками сервисов.

2.2 Система авторизации

Для большинства запросов к LUNA PLATFORM требуется обязательное наличие аккаунта, кроме запросов, не предполагающих авторизации.

Аккаунты

Аккаунт необходим для разграничения областей видимости объектов для конкретного пользователя. Каждый созданный аккаунт имеет свой собственный уникальный идентификатор «account_id». Все данные аккаунтов сохраняются в БД сервиса Accounts под этим идентификатором.

Аккаунт можно создать с помощью POST запроса «create account» к сервису API, либо с помощью запроса «register account» к сервису Admin, либо с помощью пользовательского интерфейса Admin. При создании аккаунта необходимо указать следующие данные: login (email), password и account type (тип аккаунта).

Тип аккаунта определяет, какие данные доступны пользователю. Существует три типа аккаунтов:

- user — тип аккаунта, с помощью которого можно создавать объекты и использовать только данные своего аккаунта.
- advanced_user — тип аккаунта, для которого доступны права, аналогичные «user», а также есть доступ к данным всех аккаунтов. Доступ к данным других аккаунтов означает возможность получать данные (запросы GET), проверять их наличие (запросы HEAD) и выполнять запросы на сравнение по данным других аккаунтов.
- admin — тип аккаунта, для которого доступны права, аналогичные «advanced_user», а также есть доступ к сервису Admin

С помощью заголовка «Luna-Account-Id» в запросе «create account» можно задать желаемый идентификатор аккаунта.

Токены

Токен привязывается к существующему аккаунту любого типа и позволяет наложить расширенные ограничения на выполняемые запросы. Например, при создании токена можно дать пользователю разрешение только на создание и изменение всех списков и лиц, или можно запретить использование определенных обработчиков, указав их идентификатор.

Токен и все его разрешения сохраняются в БД и привязываются к аккаунту по параметру «account_id».

При создании токена доступно задание следующих параметров:

- expiration_time – время окончания действия токена в формате RFC 3339. Можно указать бесконечное время действия токена с помощью значения «null»
- permissions – разрешения, которые доступны пользователю
- visibility_area – видимость токеном данных других аккаунтов

Типы авторизации для доступа к ресурсам

В LUNA PLATFORM доступны следующие способы авторизации:

- **BasicAuth**. Авторизация по логину и паролю (задаются во время создания аккаунта).
- **BearerAuth**. Авторизация по JWT токену (выдается после создания токена).

Примечание. Также в рамках обратной совместимости с предыдущими версиями LUNA PLATFORM доступна авторизация **LunaAccountIdAuth**. Не рекомендуется использовать данную авторизацию в новых проектах, по умолчанию отключено.

См. подробную информацию о системе авторизации LUNA PLATFORM 5 в разделе «Аккаунты, токены и способы авторизации».

2.3 Подходы при работе

В LUNA PLATFORM есть три основных операции:

1. **Детекция** * процесс распознавания лица или тела на фотографии и нормализация изображения (создание биометрического образца) для дальнейшей работы. На этапе детекции также происходит **эстимация** (оценивание) параметров лица или тела, т.е. оценивание эмоций, направления взгляда, верхней одежды и др.
2. **Экстракция** — процесс извлечения пола и возраста по изображению лица и извлечения набора уникальных свойств лица или тела (биометрических шаблонов) по биометрическому образцу для выполнения дальнейшего сравнения.
3. **Матчинг** — процесс сравнения биометрических шаблонов.

Данные операции выполняются строго друг за другом. Невозможно сравнивать биометрические шаблоны лиц предварительно не выполнив их извлечение.

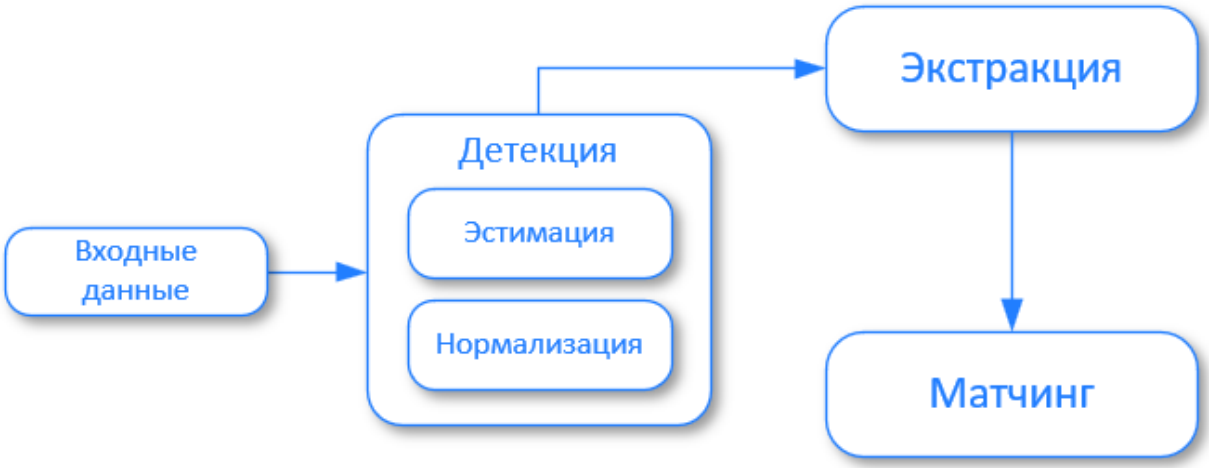


Рис. 2: Основные операции LUNA PLATFORM

Существует два основных подхода при выполнении вышеописанных операций.

2.3.1 Параллельное выполнение запросов

Первый, и основной, подход заключается в задании правил детекции, эстимации, экстракции, матчинга и др. в одном объекте — **обработчике**. После этого необходимо создать объект **событие**, который выдаст результат, основанный на всех правилах, указанных в обработчике. Использование такого подхода является самым оптимальным с точки зрения бизнес-логики.

При таком подходе выполняются следующие действия:

- с помощью запроса «create handler» создается обработчик, содержащий в себе информацию о правилах обработки изображения;

- в запросе «generate events» указывается полученный идентификатор обработчика, прикрепляется обрабатываемое изображение и генерируется событие, содержащее в себе информацию, полученную с помощью обработки правил обработчика.

Таким образом, при генерации события одновременно выполняется детекция, эстимация, экстракция, матчинг, сохранение в БД и прочее.

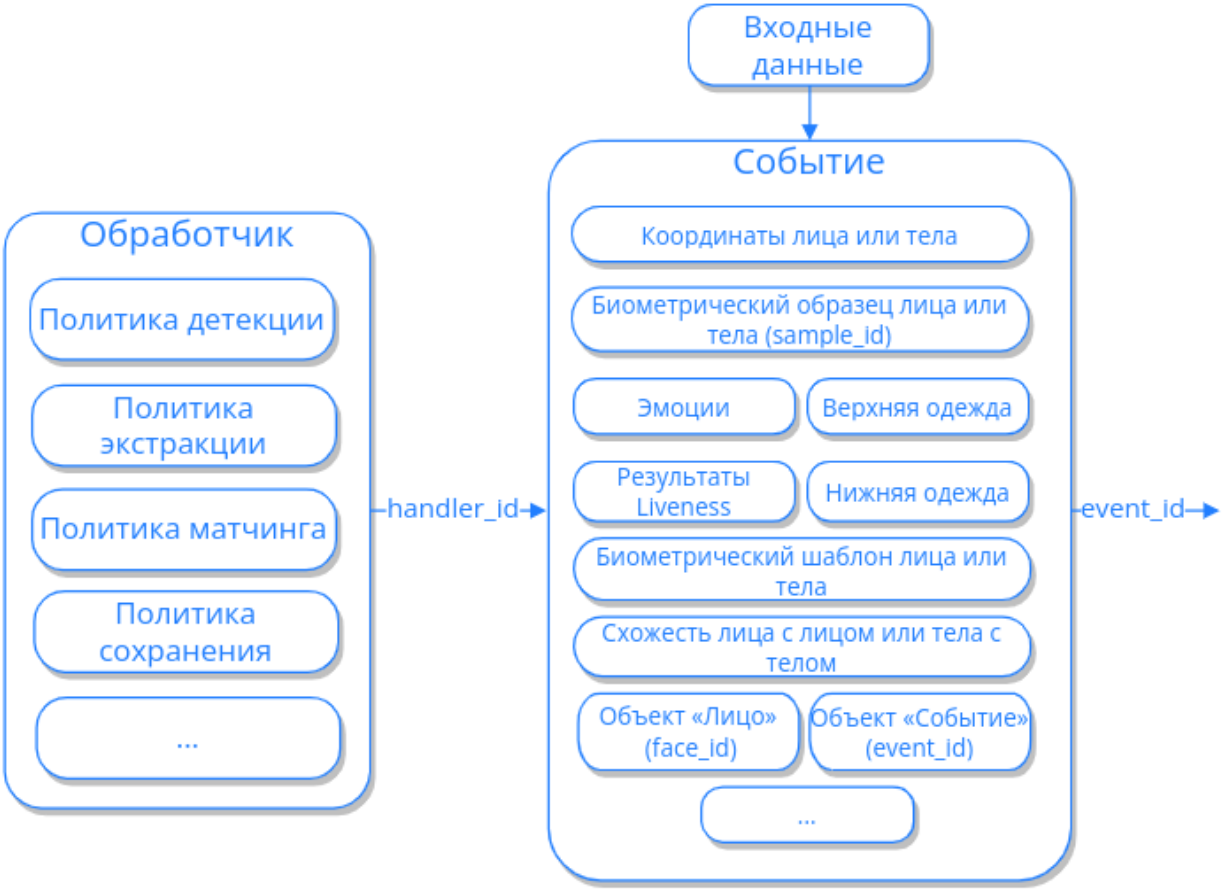


Рис. 3: Подход «Параллельное выполнение запросов»

Примеры распространенных сценариев использования обработчиков:

- регистрация эталонного биометрического шаблона с сохранением в список;
- биометрическая идентификация лиц по списку без сохранения;
- сохранение в БД лиц, идентифицированных в списке;
- сохранение событий только для уникальных лиц для последующего подсчета;
- пакетная идентификация;
- пакетный импорт.

Для некоторых сценариев может понадобиться предварительно создать определенные объекты в системе с помощью отдельных запросов. Например, для идентификации набора лиц по списку

сначала необходимо создать объект **список**, привязать к нему предварительно созданные объекты **лица**, а затем использовать обработчик и событие.

Преимущества подхода:

- все правила в одном месте; можно легко отредактировать существующий обработчик или создать новый для новой задачи. Следовательно, нет необходимости создавать и настраивать несколько разных запросов для выполнения базовых операций с изображениями;
- гибкое управление сохранением объектов в базы данных, настройка фильтров сохранения;
- возможность работы с пакетами изображений, расположенных в ZIP-архиве, на FTP-сервере, в S3-подобном хранилище и др.;
- возможность отправки уведомлений через веб-сокеты;
- сбор статистики.

Классический вариант использования такого подхода — в паре с модулем FaceStream. FaceStream анализирует видеопоток и отправляет лучшие изображения с видеопотока в LUNA PLATFORM на дальнейшую обработку. С помощью указанного обработчика, LUNA PLATFORM обрабатывает изображения по указанным правилам и сохраняет объекты в указанные базы данных.

2.3.2 Последовательное выполнение запросов

Второй подход заключается в выполнении отдельных запросов, т.е. в одном запросе нужно выполнить детекцию лица и получить его результат, затем использовать этот результат в запросе на извлечение и так далее.

При таком подходе выполняются следующие действия:

- нормализация изображения, а также детекция и эстимация лица с помощью запроса «detect faces»;
- извлечение пола, возраста и биометрического шаблона по нормализованному изображению с помощью запроса «extract attributes»;
- создание лица с помощью запроса «create face»;
- сравнение биометрических шаблонов лиц с помощью запроса «matching faces».

> Как правило, данный подход используется при работе с лицами, но при необходимости можно выполнять и отдельные запросы с телами, например, выполнять сравнение биометрических шаблонов тел с помощью запроса «matching bodies».

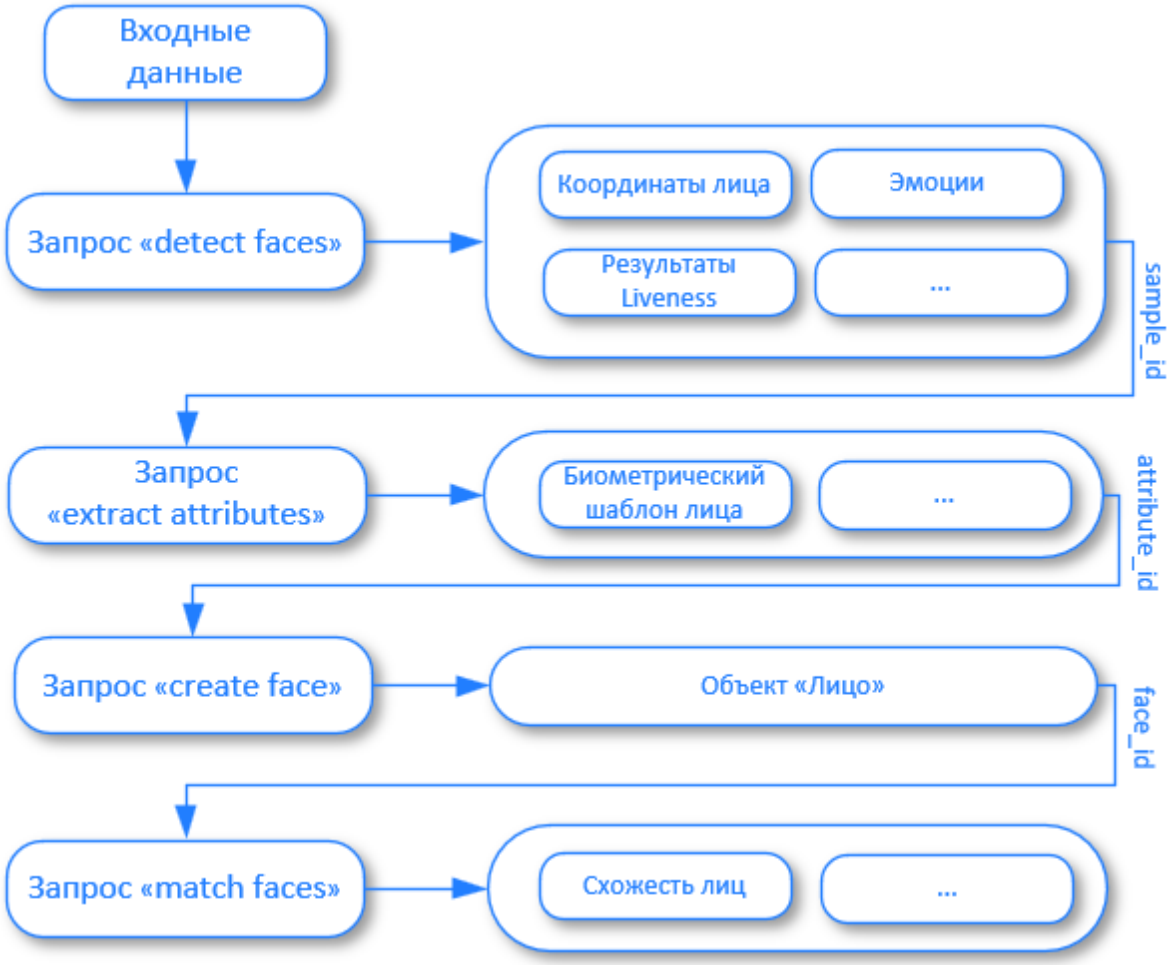


Рис. 4: Подход «Последовательное выполнение запросов»

Данный подход рекомендован к использованию:

- при первичном знакомстве с системой;
- если требуется замерить скорость выполнения каждого этапа обработки изображения отдельно;
- если требуется реализовать сценарий обработки, который затруднительно реализовать с помощью обработчиков и событий.

2.4 Детекция

LP выполняет поиск всех лиц и/или тел на каждом входящем фотоизображении.

Детекция выполняется с помощью нейронных сетей, расположенных в контейнере Remote SDK.

> Подробная информация о нейронных сетях описана в разделе «Нейросети».

На вход можно подать либо исходное изображение, либо изображение специального формата, полученное с помощью программного обеспечения VisionLabs (биометрический образец).

2.4.1 Требования к формату исходного изображения

Изображения должны отправляться только в разрешенных форматах. Формат изображения указывается в заголовке «Content-Type».

Поддерживаются следующие форматы изображений: JPG, PNG, BMP, PORTABLE PIXMAP, TIFF. Каждый формат имеет свои преимущества и предназначен для определенных задач.

Наиболее часто используемыми форматами являются **PNG** и **JPG**. Ниже приведена таблица с их преимуществами и недостатками:

Формат	Сжатие	Преимущества	Недостатки
PNG	Без потерь	Лучшее качество обработки изображения	Больший вес изображения
JPG	С потерями	Меньший вес изображения	Худшее качество обработки изображения

Таким образом, если не предполагается сохранять исходные изображения в хранилище Image Store, или же хранилище Image Store имеет достаточно большой объем, то для получения наилучших результатов обработки изображения рекомендуется использовать формат PNG.

> Не рекомендуется отправлять слишком сжатые изображения, т.к. уменьшается качество выполнения эстимации параметров лиц и/или и сравнение.

LUNA PLATFORM 5 поддерживает многие цветовые модели (например, RGB, RGBA, CMYK, HSV и др.), однако при обработке изображения, все они преобразуются в цветовую модель **RGB**.

Изображение также может может быть перекодировано в формат Base64.

> Дополнительная информация об исходных изображениях приведена в разделе «Исходные изображения».

2.4.2 Процесс детекции и эстимации лиц

Ниже представлен порядок детекции лица на изображении:

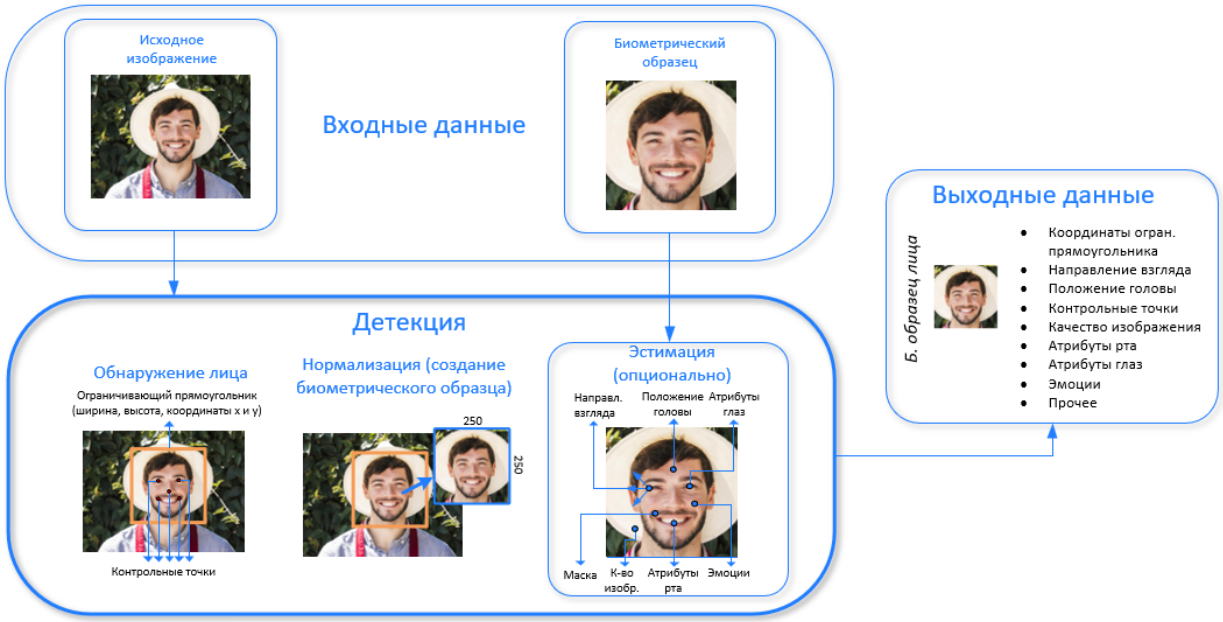


Рис. 5: Порядок обработки изображения лица

Основные шаги при детекции изображений лиц:

- **Обнаружение лица**. Определяется ограничивающий прямоугольник лица (высота, ширина, координаты «X» и «Y») и пять контрольных точек лица.
- **Нормализация**. Изображение центрируется определенным образом и обрезается до размера 250x250 пикс., необходимого для дальнейшей работы. Таким образом, все обрабатываемые изображения выглядят одинаково. Например, левый глаз всегда находится в рамке, определяемой некоторыми координатами. Такое нормализованное изображение называется **биометрическим образцом**. Сам биометрический образец сохраняется в хранилище Image Store. После создания биометрического образца, ему присваивается специальный идентификатор «sample_id», который используется для дальнейшей обработки изображения.

> Всем создаваемым объектам в LUNA PLATFORM присваиваются идентификаторы. Биометрический образец — «sample_id», лицо — «face_id», событие — «event_id» и т.д. Идентификаторы являются основным способом передачи данных между запросами и сервисами.

> Подробная информация о биометрических образцах приведена в разделе «Объект «Биометрический образец».

- **Эстимация**. Оцениваются следующие параметры лица:

- направление взгляда (углы поворота для обоих глаз);
- положение головы (наклон вперед, в сторону, поворот);
- шестьдесят восемь контрольных точек. Количество контрольных точек зависит от того, какие параметры лица подлежат оцениванию;
- качество изображения (свет, тень, размытость, освещенность и зеркальность);
- атрибуты рта (перекрытие, наличие улыбки);
- наличие маски (медицинская или тканевая маска на лице, маска отсутствует, рот перекрыт);
- эмоции (гнев, отвращение, страх, счастье, нейтральность, грусть, удивление);
- другие (см. раздел «Параметры лиц»).

При необходимости можно настроить фильтрацию по положению головы.

2.4.3 Процесс детекции и эстимации тел

Ниже представлен порядок детекции тела на изображении:

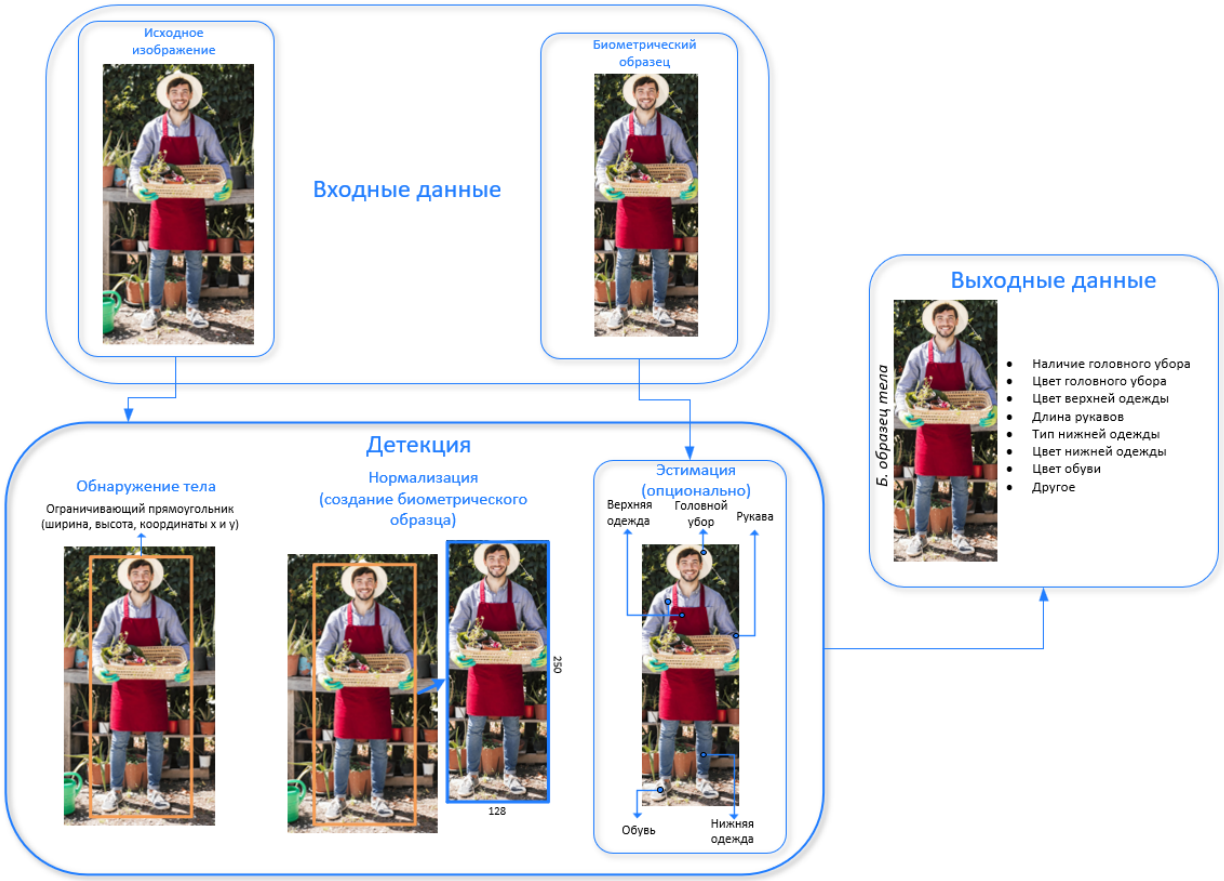


Рис. 6: Порядок обработки изображения тела

Основные шаги при детекции изображений тел:

- **Обнаружение тела**. Определяется ограничивающий прямоугольник тела (высота, ширина, координаты «X» и «Y»).
- **Нормализация**. Изображение центрируется определенным образом и обрезается до размера 128x250 пикс., необходимого для дальнейшей работы (в остальном принцип работы аналогичен процессу нормализации лица).
- **Эстимация**. Оцениваются следующие параметры тела:
  - верхняя часть тела (наличие и цвет головного убора, цвет верхней одежды);
  - длина рукавов (длинные рукава, короткие рукава, неизвестно);
  - нижняя часть тела (тип нижней одежды — брюки, шорты, юбка, неизвестно; цвет нижней одежды, наличие и цвет обуви);
  - аксессуары;
  - другие (см. раздел «Параметры тел»).

2.4.4 Взаимодействие сервисов при выполнении детекции

Ниже приведена схема взаимодействия сервисов при выполнении детекции.

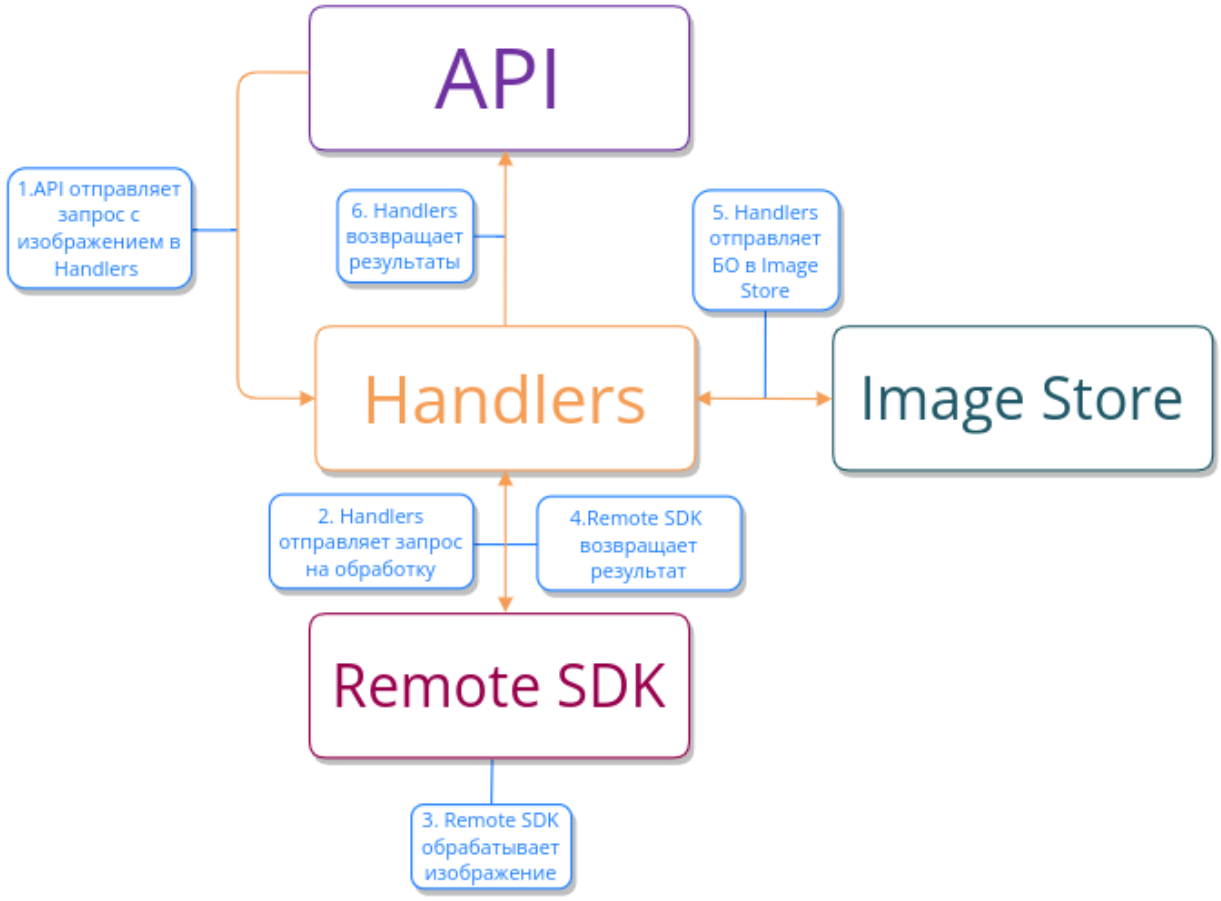


Рис. 7: Взаимодействие сервисов при выполнении детекции

2.4.5 Запросы на выполнение детекции

Ниже приведены основные запросы, где можно выполнить обнаружение лица или тела на изображении.

2.4.5.1 Запросы «create handler» и «generate events»

Данные запросы используются при подходе «Параллельное выполнение запросов».

- Запрос «create handler»:

Название параметра в теле запроса	Тело ответа	Сохранение в БД
«detect_face» или «detect_body» политики «detect_policy»	Идентификатор обработчика, содержащий правила детекции	Информация об обработчике сохраняется в базу данных Handlers

- Запрос «generate events»:

Тело запроса	Тело ответа	Сохранение в БД
Изображение	Результат детекции	БО сохраняется в хранилище Image Store, можно отключить сохранение с помощью параметра «store_sample». Если в обработчике включен параметр «store_event», то результаты детекции будут занесены в базу данных Events в таблицу «face_detect_result» или «body_detect_result»

> В запросе доступно использование схемы «multipart/form-data». Использование схемы позволяет отправить несколько изображений одновременно, указать дополнительную информацию для события и пр. См. дополнительную информацию в разделе «Использование схемы»multipart/form-data"при генерации события".

2.4.5.2 Запрос «detect faces»

Запрос «detect faces» используется при подходе «Последовательное выполнение запросов». Данный запрос не может быть использован для изображения тела.

Тело запроса	Тело ответа	Сохранение в БД
Изображение	Результат детекции	БО сохраняется в хранилище Image Store, невозможно отключить сохранение

В запросе можно отправить изображение или указать URL-адрес изображения. Можно использовать схему «multipart/form-data» для отправки нескольких изображений в запросе. Каждое из отправленных изображений должно иметь уникальное имя. Заголовок «Content-Disposition» должен содержать фактическое имя файла.

В запросе также можно указать параметры ограничивающего прямоугольника с помощью схемы «multipart/form-data». Это позволяет обозначить определенную зону с лицом на изображении. Для ограничивающего прямоугольника можно указать следующие свойства:

- координаты «x» и «y» верхнего левого угла,
- высота,
- ширина.

2.4.5.3 Запрос «sdk»

Запрос «sdk» также позволяет выполнить детекцию лица или тела на исходном изображении. В отличие от других запросов, его данные не могут быть использованы в дальнейшнем. См. подробную информацию в разделе «Ресурс sdk».

2.4.6 Запросы на выполнение эстимации

Эстимация параметров лиц и тел выполняется совместно с детекцией. Чаще всего, параметры для оценивания имеют название вида `estimate_<estimation_name>`. Например, `estimate_glasses`.

В разделе «Оцениваемые данные» приведен перечень всех возможных параметров, которые можно оценить в LUNA PLATFORM 5.

2.4.7 Результаты детекции

В ответах на запросы на обнаружение возвращается следующая информация:

- адрес до сохраненного биометрического образца лица/тела
- идентификатор биометрического образца лица/тела
- координаты ограничивающего прямоугольника лица/тела
- пять контрольных точек лица

Все эти данные используются для дальнейшего выполнения экстракции и матчинга.

Данная информация дополняется в зависимости от включенных параметров эстимации лиц/тел.

Больше об детекции лиц, тел и сервисе Remote SDK можно прочитать в разделе «Сервис Remote SDK».

2.4.8 Доверенные детекции

В LUNA PLATFORM можно выполнить детекцию лиц и тел. При необходимости можно вручную задать детекцию, явно передав координаты детекции лица или тела, полученные с помощью алгоритмов VisionLabs, и включив режим доверенной детекции. В этом случае детекция или редетекция производиться не будет.

Работа с доверенными детекциями доступна в следующих запросах:

- «detect faces»
- «sdk»
- «iso»
- «generate events»
- «perform verification»

- «generate stream events (beta)»

Необходимо выполнить следующие действия:

1. Включить режим указания доверенной детекции. Это можно сделать с помощью схемы «application/json» с параметром «trusted_detections» или с помощью схемы «multipart/form-data» с заголовком «X-Luna-Trusted-Detections».
2. Передать координаты доверенной детекции («x», «y», «width», «height») в параметрах «face_bounding_boxes» или «body_bounding_boxes».

Примечание. Обратите внимание, что если отключить стандартную детекцию и оставить активным только режим доверенной детекции, то использование обычного детектора становится нецелесообразным.

Примечание. Не рекомендуется указывать сторонние детекции, полученные другими алгоритмами, так как это может привести к потере точности или искажению результатов анализа.

2.5 Экстракция

В LUNA PLATFORM под словом «экстракция» понимается:

- извлечение биометрических шаблонов и базовых атрибутов (пол, возраст, этническая принадлежность) лиц;
- извлечение биометрических шаблонов тел.

> По изображению тела тоже можно выполнить оценку пола и возраста человека, однако такой способ определения является менее точным и происходит на этапе эстимации. К тому же, выполнять оценку базовых атрибутов по телу рекомендуется совместно с извлечением базовых атрибутов по лицу.

Биометрические шаблоны — это наборы характеристик, считываемых с лиц или тел на изображениях. Для БШ требуется гораздо меньше памяти по сравнению с исходным изображением.

Восстановить исходное изображение лица или тела из БШ невозможно по соображениям безопасности персональных данных.

Биометрические шаблоны используются для сравнения лиц с лицами и тел с телами. Нельзя сравнить два лица или тела при отсутствии извлеченных БШ. К примеру, при необходимости сравнить лицо из базы данных с входным изображением лица, необходимо извлечь биометрический шаблон для данного изображения.

Биометрический шаблон извлекается с помощью нейронной сети, расположенной в контейнере Remote SDK.

> При необходимости можно зашифровать биометрический шаблон для обеспечения дополнительной безопасности. См. раздел «Шифрование биометрических шаблонов» для более подробной информации.

> Подробная информация о нейронных сетях описана в разделе «Нейросети».

> См. дополнительную информацию о биометрических шаблонах в разделе «Биометрический шаблон».

2.5.1 Процесс экстракции

Для извлечения данных требуется следующая информация, полученная при выполнении детекции:

- биометрический образец лица или тела;
- координаты ограничивающего прямоугольника для лица или тела;
- пять контрольных точек лица.

Ниже представлен порядок выполнения экстракции:

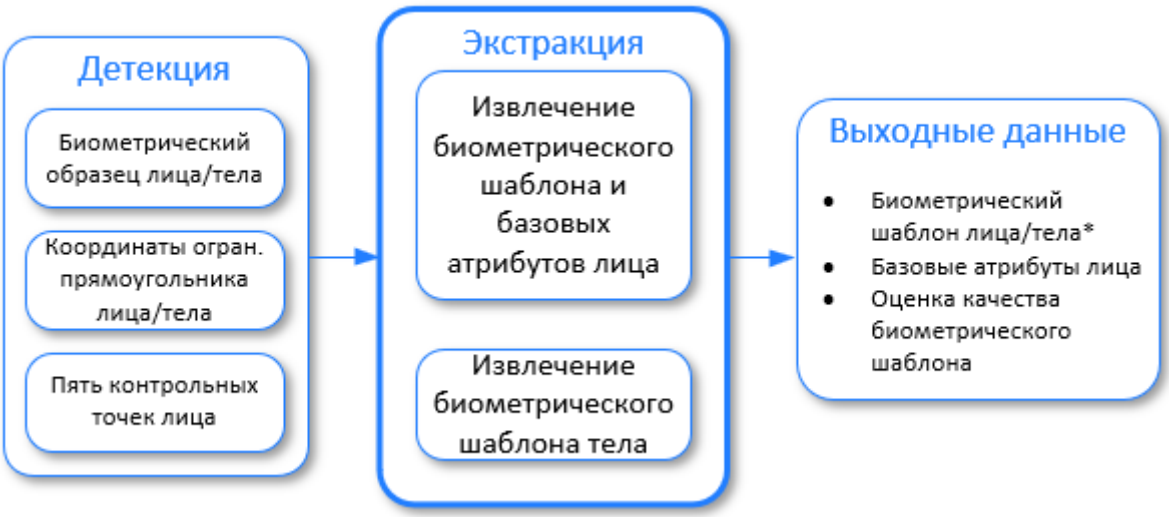


Рис. 8: Порядок выполнения экстракции

* в большинстве запросов на извлечение, биометрический шаблон записывается в базу данных, не выдаваясь в теле ответа (см. «Запросы на извлечение биометрических шаблонов»).

2.5.2 Особенности извлечения данных из лиц

Биометрический шаблон лица и базовые атрибуты лица, полученные из одного и того же изображения, сохраняются в базе данных как объект **атрибут**. Этот объект может включать и БШ и базовые атрибуты, или только одну из этих сущностей.

Можно извлекать БШ и базовые атрибуты с использованием нескольких биометрических образцов одного и того же лица одновременно. Таким образом можно получить **агрегированный** биометрический шаблон и агрегированные базовые атрибуты. Точность сравнения и извлечения базовых атрибутов посредством агрегированного БШ выше. Агрегацию следует использовать при работе с изображениями, полученными с веб-камер.

> Агрегированный БШ можно получить из изображений, но не из уже созданных БШ.

Подробная информация об агрегации описана в разделе «Агрегирование».

2.5.2.1 Временные атрибуты

После извлечения атрибутов с помощью соответствующих запросов (см. «Запросы на извлечение данных»), полученные данные сохраняются в БД Redis в качестве временных атрибутов.

Временные атрибуты имеют TTL (время существования) и удаляются из базы данных по истечении указанного периода. В запросе можно указать период от 1 до 86400 секунд. По умолчанию TTL составляет 5 минут.

> Можно получить информацию о временном атрибуте по его идентификатору, пока не истечет его TTL.

Для того, чтобы сохранить атрибуты в базу данных, необходимо создать объект **лицо**, привязав к нему атрибуты. Лицо создается либо отдельным запросом после извлечения атрибутов (лицо сохраняется в БД Faces), либо во время генерации события (лицо сохраняется в БД Faces или в БД Events). Можно создать несколько лиц и объединить их в список, который можно использовать при сравнении.

> См. подробную информацию в разделах «Объект «Лицо» и «Объект «Список».

Хранить атрибуты можно во внешней базе данных и использовать их в LP только тогда, когда требуется. См. раздел «Создание объектов с использованием внешних данных».

> Подробная информация об атрибутах приведена в разделе «Объект «Атрибут».

2.5.3 Особенности извлечения данных из тел

Для тел не существует атрибутов. Вся информация о телах может быть сохранена только как объект **событие**. Соответственно, извлеченные данные не сохраняются в базу данных Redis. Информация о телах может быть сохранена только в базу данных сервиса Events при включении соответствующего параметра.

> Также как и лица, события можно агрегировать. Подробная информация об агрегации описана в разделе «Агрегирование».

2.5.4 Взаимодействие сервисов при выполнении экстракции

Ниже приведена схема взаимодействия сервисов при выполнении экстракции.

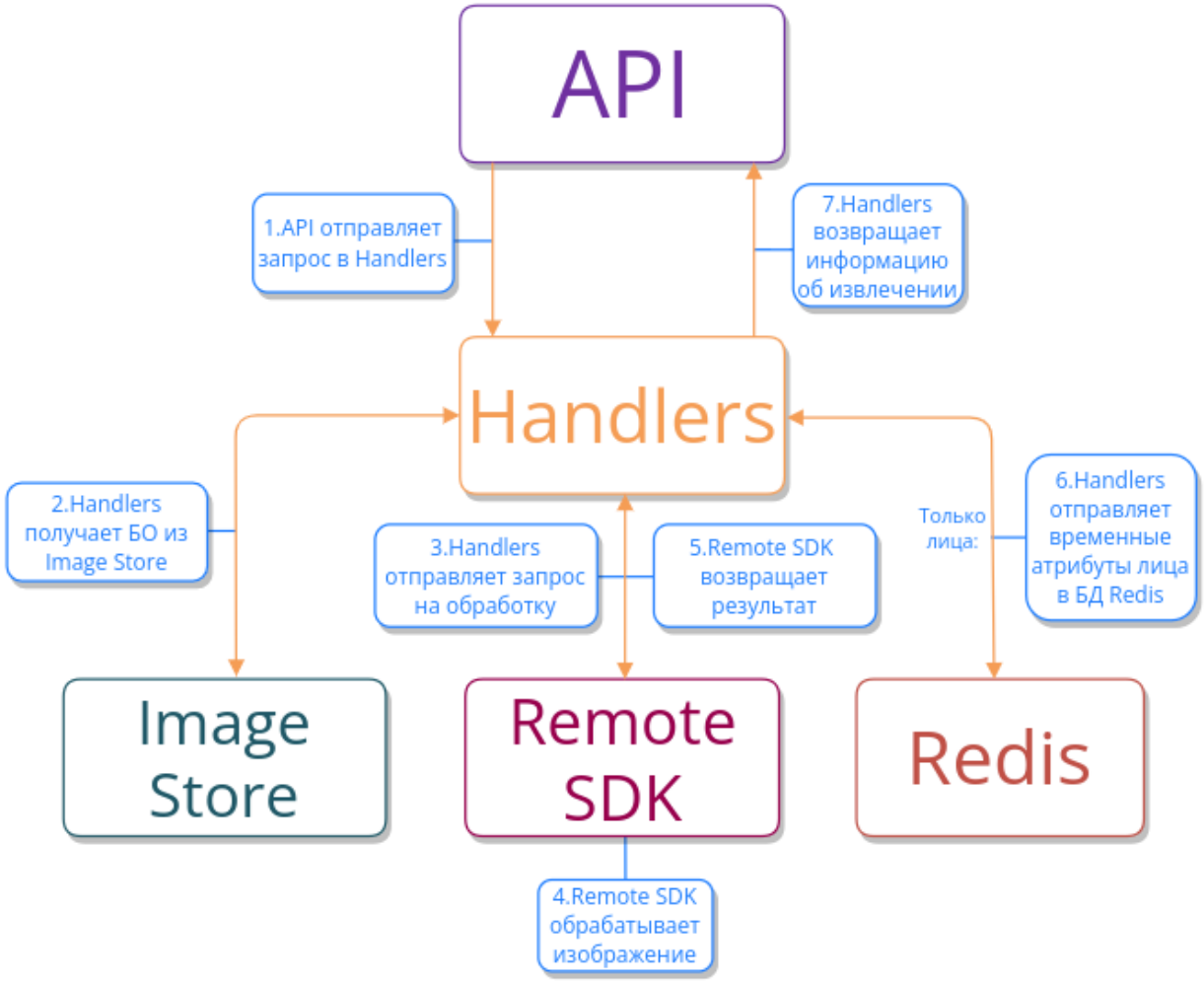


Рис. 9: Схема взаимодействия сервисов при выполнении экстракции

2.5.5 Запросы на выполнение экстракции

Ниже приведены основные запросы, где можно выполнить извлечение данных из изображений лиц или тел.

2.5.5.1 Запросы «create handler» и «generate events»

Данные запросы используются при подходе «Параллельное выполнение запросов».

- Запрос «create handler»:

Название параметров в теле запроса	Тело ответа	Сохранение в БД
БШ лица: «extract_face_descriptor» политики «extract_policy»	Идентификатор обработчика, содержащий правила извлечения	Информация об обработчике сохраняется в базу данных Handlers
БШ тела: «extract_body_descriptor» политики «extract_policy»	Аналогично описанию выше	Аналогично описанию выше
Базовые атрибуты лица: «extract_basic_attributes» политики «extract_policy»	Аналогично описанию выше	Аналогично описанию выше

- Запрос «generate events»:

Тело запроса	**Тело ответа**	**Сохранение в БД**
Изображение	Результаты извлечения	Если в обработчике включен параметр «store_event», то БШ лица или тела будет занесен в базу данных Events, в таблицу «face_descriptor» или «body_descriptor», а базовые атрибуты лица в таблицу «event». Если в обработчике включен параметр «store_face», то будет создано лицо и соответствующий БШ будет занесен в базу данных Faces в таблицу «descriptor», а базовые атрибуты лица в таблицу «event». Если в обработчике включен параметр «store_attribute», то БШ и базовые атрибуты лица будут занесен в базу данных Redis на период TTL.

> В запросе доступно использование схемы «multipart/form-data». Использование схемы позволяет отправить несколько изображений одновременно, указать дополнительную информацию для события и пр. См. дополнительную информацию в разделе «Использование схемы»multipart/form-data"при генерации события".

2.5.5.2 Запрос «extract attributes»

Запрос «extract attributes» используется при подходе «Последовательное выполнение запросов». Данный запрос не может быть использован для биометрического образца тела.

Параметр запроса	Тело запроса	Тело ответа	Сохранение в БД
«extract_descriptor» и «extract_basic_-attributes»	Изображени	Результаты извлечения	Временные атрибуты сохраняются в базу данных Redis и удаляются оттуда по истечении периода TTL. В момент создания лица, БШ сохраняется в в таблицу «descriptor» базы данных Faces, а базовые атрибуты сохраняются в таблицу «attributes»

Для сохранения информации в БД Faces, нужно привязать атрибуты к лицу с помощью запроса «create face». В противном случае, по истечению TTL временные атрибуты лица будут удалены из базы данных Redis. Невозможно сохранить лицо в базу данных Events, используя комбинацию запросов «extract attributes» + «create face». Для этого нужно воспользоваться подходом «Параллельное выполнение запросов» (см. выше).

2.5.5.3 Запрос «sdk»

Запрос «sdk» также позволяет выполнить извлечение базовых атрибутов или биометрического шаблона. Данный запрос позволяет вернуть биометрический шаблон лица или тела в особом формате, который можно использовать при сравнении «сырых» биометрических шаблонов. См. подробную информацию в разделе «Ресурс sdk».

2.5.6 Результаты извлечения

В ответах на запросы на извлечения возвращается следующая информация:

- идентификатор временного атрибута лица (не возвращается если в обработчике выключен параметр «store_attribute» политики «attribute_policy»)
- адрес до сохраненного в БД Redis временного атрибута лица (не возвращается если в обработчике выключен параметр «store_attribute» политики «attribute_policy»)
- базовые атрибуты лица
- качество БШ лица/тела
- набор идентификаторов биометрических образцов по которым было выполнено извлечение

2.6 Матчинг

При наличии биометрических шаблонов, LP позволяет искать похожие лица или тела в базе данных посредством сравнения данного биометрического шаблона с биометрическими шаблонами, хранящимися в базах данных Redis, Faces, Events или с биометрическими шаблонами, переданными напрямую.

Сравнение выполняется с помощью сервиса Python Matcher. В ответе на сравнение выдается степень схожести, значение которой лежит в интервале от 0 до 1. Высокая степень схожести означает, что два БШ принадлежат одному и тому же человеку.

Существует два типа биометрических шаблонов — биометрический шаблон лица и биометрический шаблон тела. Сравнение биометрических шаблонов тел не такое точное, как сравнение биометрических шаблонов лиц. При большом количестве событий-кандидатов, вероятность ложных определений лучших совпадений выше, чем при большом количестве лиц-кандидатов.

Исходники сравниваемых объектов представлены в виде **эталонов** (объектов, подлежащих сравнению) и **кандидатов** (набора объектов, с которыми производится сравнение). Каждый эталон сопоставляется с каждым из заданных кандидатов.

> Сравнение не может быть выполнено при отсутствии БШ для любого из сравниваемых лиц.

Можно выбрать следующие объекты в качестве эталонов и кандидатов.

Эталоны:

- Атрибуты
- Лица
- Внешние ID лиц и событий
- События (лицо или тело)
- ID треков событий
- Биометрические шаблоны

Кандидаты:

- Лица
- События (лицо или тело)
- Атрибуты
- Биометрические шаблоны

Эталоны указываются с помощью ID соответствующих объектов. Если задается несуществующий эталон (например, указан несуществующий ID в поле «event_id» или «face_id»), возвращается соответствующая ошибка.

Кандидаты указываются с помощью фильтров. Результаты сравнения возвращаются для тех кандидатов, которые соответствуют заданным фильтрам. Если таковых не обнаружено (например,

указан несуществующий ID в поле «event_ids» или «face_ids»), не будет ни результата сравнения, ни ошибки. То есть поле результата будет пустым.

2.6.1 Процесс выполнения матчинга

Ниже представлен процесс выполнения матчинга с использованием лиц в качестве кандидатов и эталонов:

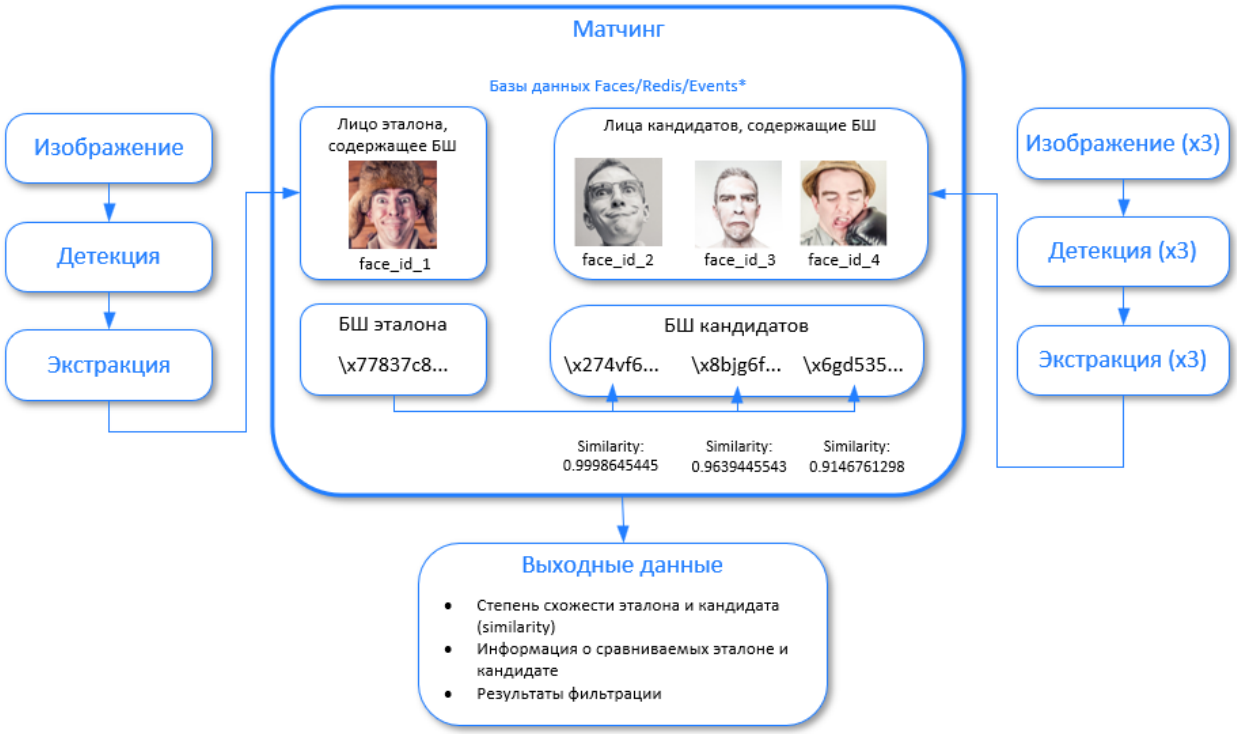


Рис. 10: Процесс выполнения матчинга

* Биометрические шаблоны можно также передать напрямую с помощью запроса «raw matching».

2.6.2 Взаимодействие сервисов при выполнении матчинга

Ниже приведена схема взаимодействия сервисов при выполнении матчинга.

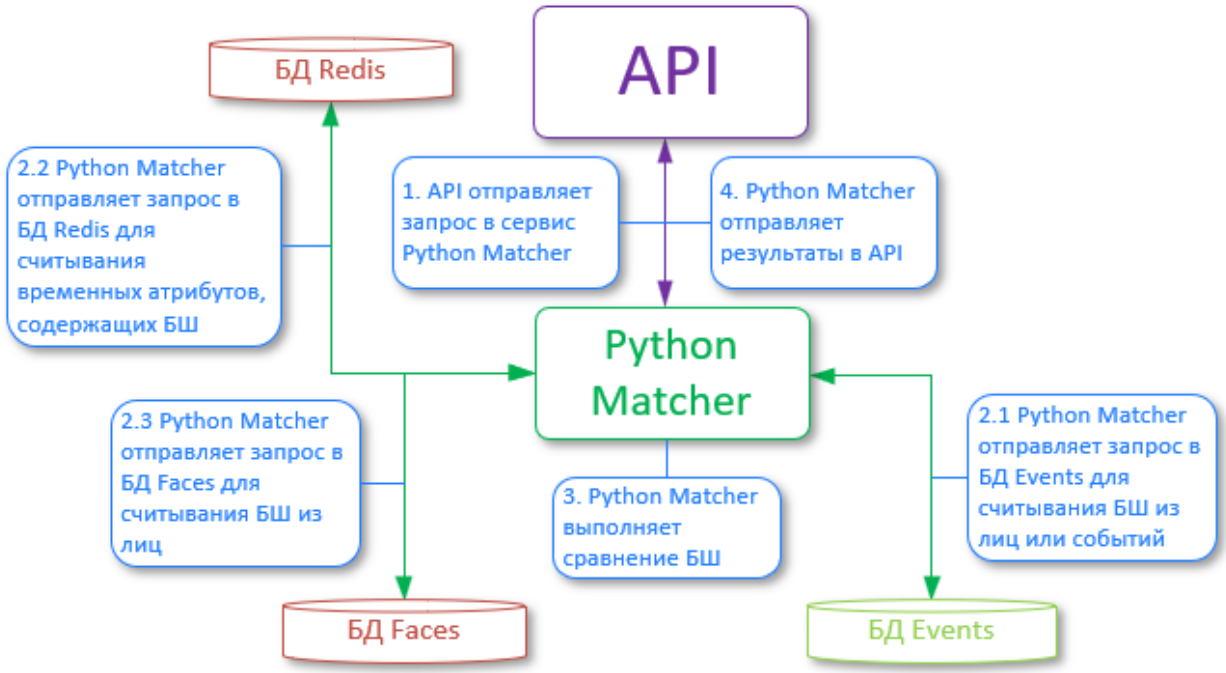


Рис. 11: Взаимодействие сервисов при выполнении матчинга

См. дополнительные сценарии взаимодействия сервисов при выполнении сравения в разделе «Диаграммы сравнения».

2.6.3 Фильтрация результатов матчинга

Результаты матчинга можно фильтровать. Для фильтрации нужно указать соответствующие параметры запроса.

Ниже приведены некоторые примеры фильтров событий и лиц.

Для фильтрации событий можно:

- Указать камеру (поле «source») и период (поля «create_time__gte» и «create_time__lt» или «end_time__gte» и «end_time__lt»);
- Указать теги (поле «tags») для событий в качестве фильтров и выполнить сравнение для событий с помощью одних этих тегов;
- Указать географическую зону и выполнить сравнение с помощью событий, созданных только в этой зоне. Можно указать конкретную локацию (город, район, улица, дом) или область, заданную географическими координатами.

Для фильтрации лиц можно:

- Указать список внешних ID лиц для выполнения сравнения по ним;
- Указать список ID и выполнять сравнение по нему.

См. запросы «matching faces», «human body matching» и политику «match_policy» запроса «generate events» в справочном руководстве сервиса API для более подробной информации обо всех доступных фильтрах.

2.6.4 Запросы на матчинг

Ниже приведены основные запросы, где можно сравнить биометрические шаблоны.

2.6.4.1 Запросы «create handler» и «generate events»

Данные запросы используются при подходе «Параллельное выполнение запросов».

- Запрос «create handler»:

Название политики в теле запроса	Тело ответа	Сохранение в БД
«match_policy»	Идентификатор обработчика, содержащий правила сравнения.	Информация об обработчике сохраняется в базу данных Handlers

- Запрос «generate events»:

Тело запроса	Тело ответа	Сохранение в БД
Изображение	Результаты матчинга	Если в обработчике включен параметр «store_event», то результаты матчинга будут занесены в таблицы «face_match_result» или «event_match_result» базы данных Events

> В запросе доступно использование схемы «multipart/form-data». Использование схемы позволяет отправить несколько изображений одновременно, указать дополнительную информацию для события и пр. См. дополнительную информацию в разделе «Использование схемы»multipart/form-data”при генерации события”.

2.6.4.2 Запросы «matching faces» и «matching bodies»

Запросы «matching faces» и «matching bodies» используются при подходе «Последовательное выполнение запросов».

Тело запроса	Тело ответа	Сохранение в БД
кандидаты, эталоны и фильтры указываются в параметрах «candidates» и «references»	Результаты матчинга	Нет

2.6.4.3 Запрос «raw matching»

Можно задавать в качестве эталонов и кандидатов биометрические шаблоны в форматах SDK, Raw и XPK-файлах с помощью запроса «raw matching».

Все данные по БШ предоставляются посредством запроса, таким образом можно использовать этот запрос при необходимости производить сравнение БШ, не хранимых в базах данных LUNA PLATFORM.

> В LUNA PLATFORM есть возможность извлечь биометрический шаблон в формате SDK (см. раздел «Форматы биометрических шаблонов»).

2.6.5 Результаты матчинга

В ответах на запросы на матчинг возвращается следующая информация:

- информация о эталоне
- информация о кандидате
- степень схожести каждого кандидата с эталоном
- отфильтрованные кандидаты

Для более подробной информации о сервисах сравнения БШ см. раздел «Сервисы сравнения».

2.6.6 Верификация

Можно использовать ресурс «/verifiers» для создания специального обработчика для верификации. Он включает несколько политик для обработки входных изображений. См. подробную информацию о верификаторе в разделе «Описание верификаторов».

Этот запрос используется для сравнения одного заданного объекта со входным объектом. Для задач идентификации необходимо использовать другие запросы сравнения биометрических шаблонов.

2.6.7 Сравнение большого набора биометрических шаблонов

Иногда требуется сравнивать очень большое количество кандидатов. При классическом сравнении методом последовательного перебора большого количества биометрических шаблонов с помощью сервиса Python Matcher, невозможно получить малую задержку при высоком количестве

запросов в секунду. Поэтому требуется использовать методы аппроксимации, реализованные в дополнительном модуле LUNA Index Module, которые обменивают некоторую точность на высокую скорость. Эти методы ускоряют сравнение за счет построения индекса, содержащего данные предварительной обработки.

LUNA Index Module лицензируется отдельно и поставляется по запросу к VisionLabs. Поставка модуля содержит всю необходимую документацию и скрипт Docker Compose.

2.7 Сохраняемые данные и объекты LP

В этом разделе описываются данные, сохраняемые в LUNA PLATFORM 5.

Эта информация может быть полезна при использовании LUNA PLATFORM 5 в соответствии с правовой системой Европейского Союза и Общими положениями о защите (GDPR).

В этом разделе не описываются правовые аспекты использования персональных данных. Следует учитывать, какие сохраненные данные могут интерпретироваться как персональные в соответствии с местными законами и постановлениями.

> Обратите внимание, что комбинации данных LUNA PLATFORM могут быть расценены в соответствии с законом как персональные данные, даже если отдельные поля данных не содержат персональных данных. Например, объект лицо, содержащий параметр «user_data» и БШ, может считаться персональными данными.

Объекты и данные сохраняются в LP при выполнении определённых запросов. Поэтому следует отключать сохранение ненужных данных при выполнении этих запросов.

Рекомендуется ознакомиться с этим разделом, прежде чем принимать решение о том, какие данные следует получать и хранить в LUNA PLATFORM.

> В этом документе рассматривается использование ресурсов, приведенных в спецификации OpenAPI, и создание только объектов LUNA PLATFORM 5. Данные, созданные с помощью сервисов Backport 3 и Backport 4, в этом разделе не рассматриваются.

2.7.1 Исходные изображения

Фотоизображения являются основными источниками данных для LP. Они необходимы для создания биометрических образцов и проверки Liveness.

Можно отправить в LP сами изображения или URL-адреса изображений.

> Обратите внимание, что после нормализации исходных изображений, биометрические образцы по умолчанию сохраняются в формате JPG, даже если исходное изображение отправляется в формате PNG. При необходимости можно изменить формат сохраняемых биометрических образцов с помощью настройки «default_image_extension» сервиса Image Store.

Не рекомендуется отправлять повернутые изображения в LP, поскольку они будут обработаны неверно и потребуется выполнить их поворот. Изображение можно повернуть, используя внутренние механизмы LP, двумя способами:

- включить параметр автоориентации «use_exif_info» по данным EXIF, имеющимся в параметрах запроса;
- включить настройку автоориентации «LUNA_REMOTE_SDK_USE_AUTO_ROTATION» в настройках сервиса Configurator.

Более подробная информация о работе с повернутыми изображениями приведена в разделе Особенности работы с сервисами.

> См. подробную информацию о требованиях к исходным изображениям в разделе «Требования к формату исходного изображения».

2.7.1.1 Использование исходных изображений

Исходные изображения можно указать для обработки при выполнении POST запросов к следующим ресурсам:

- «/sdk»;
- «/detector»;
- «/handlers/{handler_id}/events»;
- «/verifiers/{verifier_id}/verifications»;
- «/liveness».

2.7.1.2 Сохранение исходных изображений

Как правило, исходные изображения не требуется сохранять после обработки. При необходимости их можно сохранить для целей тестирования системы или в рамках бизнес-логики, например, когда исходное изображение необходимо отобразить в пользовательском интерфейсе.

Исходные изображения можно сохранять в LP:

- используя POST запрос к ресурсу «/images».
- используя POST запрос к ресурсу «/handlers/{handler_id}/events». Исходные изображения сохраняются, если включён параметр «store_image» для «image_origin_policy» в «storage_policy» во время создания обработчика с помощью ресурса «/handlers».

Исходные изображения хранятся в бакете «visionlabs-image-origin» сервиса Image Store.

Сохранение метаданных с изображением

В ресурсе «/images» вместе с изображением можно сохранять пользовательские метаданные, используя заголовок `X-Luna-Meta-<user_defined_key>` со значением `<user_defined_value>`. В бакете исходных изображений хранилища Image Store, метаданные сохраняются в отдельный файл `<image_id>.meta.json`, который расположен рядом с исходным изображением.

В ответе на запрос на получение изображения (запрос «get image») нужно указать заголовок `with_meta=1` для получения метаданных изображения в заголовке ответа.

Чтобы сохранять значения метаданных для нескольких ключей, необходимо задать несколько заголовков.

2.7.1.3 Удаление исходных изображений

Исходные изображения хранятся в бакете в течение неограниченного времени.

Исходные изображения можно удалить из бакета следующим образом:

- с помощью запроса DELETE к ресурсу «/images/{image_id}»,
- вручную, удалив соответствующие файлы из бакета.

2.7.2 Объект «Биометрический образец»

2.7.2.1 Использование биометрического образца

Для лица и тела создаются отдельные биометрические образцы.

БО необходимы:

- для оценки базовых атрибутов.
- для оценки параметров лица, тела и изображения.
- для извлечения БШ для лиц и тел.
- при изменении версии нейросети БШ.

> При изменении версии нейросети нельзя использовать БШ предыдущей версии. БШ новой версии можно извлечь, только если сохранен исходный биометрический образец.

БО также можно использовать в качестве аватаров для лиц и событий. Например, если необходимо отобразить аватар в пользовательском интерфейсе.

БО хранятся в бакетах хранилища Image Store неограниченное время.

БО хранятся в следующих бакетах:

- «visionlabs-samples» – бакет для БО лиц.
- «visionlabs-bodies-samples» – бакет для БО тел.

Пути к бакетам указываются в параметрах «bucket» в разделах «LUNA_IMAGE_STORE_FACES_SAMPLES_ADDRESS» и «LUNA_IMAGE_STORE_BODIES_SAMPLES_ADDRESS» в сервисе Configurator.

БО следует хранить до тех пор, пока не будут выполнены все необходимые запросы по оценке параметров лица, тела, оценке базовых атрибутов и извлечению БШ.

2.7.2.2 Создание и сохранение биометрических образцов

БО создаются при обнаружении лица и/или тела на изображении. Они создаются при выполнении следующих запросов:

- запроса POST к ресурсу «/detector». БО создаются и сохраняются в неявной форме. Пользователь не влияет на их создание.

- запроса POST к ресурсу «/handlers/{handler_id}/events». Исходные изображения сохраняются при активации параметра «store_sample» для «face_sample_policy» и «body_sample_policy» в «storage_policy» во время создания обработчика с помощью ресурса «/handlers».
- запроса POST к ресурсу «/verifiers/{verifier_id}/verifications». БО сохраняются при активации параметра «store_sample» для «face_sample_policy» в «storage_policy» при создании верификатора с помощью ресурса «/verifiers». БО тела не сохраняются с помощью этого ресурса.

Сохранение внешних биометрических образцов

БО можно передать непосредственно в LP из внешнего программного обеспечения VisionLabs (например, FaceStream). Программное обеспечение самостоятельно создает биометрический образец из исходного изображения.

Можно вручную сохранить внешние БО в LP (внешний БО должен быть передан в запросе) с помощью следующих запросов:

- С помощью POST запроса к ресурсу «/samples/{sample_type}».
- Если установлен параметр «warped_image» в POST запросе к ресурсу «/detector».
- Если для параметра «image_type» установлено значение «1» (БО лица) или «2» (БО тела) в параметрах POST запроса к ресурсу «/handlers/{handler_id}/events». Для «face_sample_policy» и «body_sample_policy» из «storage_policy» должен активироваться параметр «store_sample».

2.7.2.3 Отключение сохранения биометрических образцов

Отключение сохранения БО при выполнении запроса к ресурсу «/detector» невозможно. Созданные БО можно удалить вручную после выполнения запроса.

В ресурсе «/handlers» предусмотрен параметр «storage_policy», с помощью которого можно отключить сохранение:

- БО лиц. Необходимо установить «face_sample_policy» > «store_sample» на «0».
- БО тел. Необходимо установить «body_sample_policy» > «store_sample» на «0».

В ресурсе «/verifiers» предусмотрен параметр «storage_policy», с помощью которого можно отключить сохранение БО лиц. Необходимо установить «face_sample_policy» > «store_sample» на «0».

2.7.2.4 Удаление биометрических образцов

Можно воспользоваться следующими способами удаления БО лица или тела:

- Выполнить запрос DELETE к ресурсу «/samples/faces{sample_id}», чтобы удалить БО лица по его идентификатору.
- Выполнить запрос DELETE к ресурсу «/samples/bodies{sample_id}», чтобы удалить БО тела по его идентификатору.
- Вручную удалить из бакета необходимые БО лица или тела.

2.7.2.5 Получение информации о биометрических образцах

Биометрический образец лица или тела можно получить по его идентификатору.

- Выполнить запрос GET к ресурсу «/samples/faces/{sample_id}», чтобы получить БО лица по его идентификатору.
- Выполнить запрос GET к ресурсу «/samples/bodies/{sample_id}», чтобы получить БО тела по его идентификатору.

> При удалении БО лица или тела система выдает ошибку при выполнении GET запроса.

2.7.3 Биометрический шаблон

Базовое описание и применение биометрических шаблонов описано в разделе «Экстракция» выше.

См. дополнительную информацию о шифровании биометрических шаблонов в разделе «Шифрование биометрических шаблонов».

См. дополнительную информацию в разделе «Форматы биометрических шаблонов».

2.7.4 Объект «Атрибут»

Данный объект предназначен только для работы с лицами.

> Перед прочтением рекомендуется ознакомиться с разделом «Экстракция».

Атрибуты – это временные объекты, которые включают в себя базовые атрибуты и биометрические шаблоны лица. Эти данные можно получить после обработки БО.

Базовые атрибуты содержат следующие личные данные:

- **возраст**. Возраст определяется в годах.
- **пол**. Предполагаемый пол: 0 — женский, 1 — мужской.
- **этническая принадлежность**. Предполагаемая этническая принадлежность.

> Для того, чтобы персональные данные не хранились в системе, можно отключить извлечение базовых атрибутов.

БШ не считается персональными данными. С его помощью невозможно восстановить исходное лицо.

2.7.4.1 Использование атрибутов

Объект «Атрибут» может использоваться в следующих случаях:

- при сравнении с помощью атрибутов. Оно может быть выполнено с использованием:

- ресурса «/matcher/faces»,
- ресурса «/tasks/cross_match»,
- ресурса «/verifiers/{verifier_id}/verifications».

Поскольку у атрибутов есть срок существования (по умолчанию он равен 5 минутам), их удобно использовать для проверки или идентификации. Они удаляются вскоре после получения результата.

- для создания ROC-кривых с помощью ресурса «/tasks/roc» (см. раздел «Задача ROC-curve calculating»),
- для сохранения данных существующего атрибута для лица с помощью ресурса «/faces».

  Это не единственный способ сохранить БШ и базовые атрибуты лица. Можно использовать ресурс «/handlers/{handler_id}/events» для создания лица и прикрепления к нему извлечённого БШ и базовых атрибутов.

Базовые атрибуты, сохраненные в лицах или объектах событий, могут использоваться для фильтрации соответствующих объектов при выполнении запросов.

2.7.4.2 Создание и сохранение атрибутов

Атрибуты можно создать при отправке запросов к следующим ресурсам:

- «/extractor»,

  Необходимо включить параметры запроса «extract_basic_attributes» и «extract_descriptor», чтобы извлечь соответствующие данные. Атрибуты сохраняются в неявной форме после выполнения запроса.

- «/handlers/{handler_id}/events»,

  Параметры «extract_basic_attributes», «extract_face_descriptor» и «extract_body_descriptor» включаются в обработчике для извлечения соответствующих данных. Необходимо включить опцию в обработчике для хранения атрибутов: «storage_policy» > «attribute_policy» > «store_attribute».

- «/verifiers/{verifier_id}/verifications».

  Параметр «extract_basic_attributes» следует активировать в верификаторе для извлечения соответствующих данных. Необходимо активировать опцию «storage_policy» > «attribute_policy» > «store_attribute» в верификаторе для хранения атрибутов.

Атрибуты можно создать с использованием внешних БШ и внешних базовых атрибутов с помощью следующего ресурса:

- «/attributes».

2.7.4.3 Время существования атрибутов

У атрибутов есть время существования (TTL). По истечении TTL атрибуты автоматически удаляются. Поэтому нет необходимости удалять атрибуты вручную.

Значение TTL по умолчанию можно задать в параметре «default_ttl» в сервисе Configurator. Максимальное значение TTL можно задать в параметре в сервисе Configurator.

TTL можно указывать непосредственно в запросах к следующим ресурсам:

- «/extractor» в параметре «ttl».
- «/handlers» в «storage_policy» > «attribute_storage» в параметре «ttl»

2.7.4.4 Отключение извлечения атрибутов

Можно отключить извлечение базовых атрибутов, установив значение параметра «extract_basic_-attributes» равным «0» в следующих ресурсах:

- «/extractor»
- «/handlers»
- «/verifiers»

Можно отключить извлечение БШ, установив значение параметра «extract_descriptor» равным «0» в следующих ресурсах:

- «/extractor»
- «/handlers»

2.7.4.5 Отключение сохранения атрибутов

Можно отключить параметр «storage_policy» > «attribute_policy» > «store_attribute» в ресурсе «/handlers» для отключения сохранения атрибутов. При использовании этого обработчика для ресурса «/handlers/{handler_id}/events» атрибуты не сохраняются даже в течение указанного периода TTL.

Можно отключить параметр «storage_policy» > «attribute_policy» > «store_attribute» в ресурсе «/verifiers» для отключения сохранения атрибутов. При использовании этого верификатора для ресурса «/verifiers/{verifier_id}/verifications», атрибуты не сохраняются даже в течение указанного периода TTL.

2.7.4.6 Удаление атрибутов

Атрибуты автоматически удаляются по истечении времени существования.

Для удаление атрибутов по их идентификатору необходимо выполнить запрос DELETE к ресурсу «/attributes/{attribute_id}».

2.7.4.7 Получение информации об атрибутах

Можно получить информацию о существующих временных атрибутах до истечения времени существования. Для этого необходимо:

- Выполнить запрос GET к ресурсу «/attributes/{attribute_id}», чтобы получить информацию о временном атрибуте по его идентификатору.
- Выполнить запрос GET к ресурсу «/attributes», чтобы получить информацию о ранее созданных атрибутах по их идентификаторам.
- Выполнить запрос GET к ресурсу «/attributes/{attribute_id}/samples», чтобы получить информацию обо всех биометрических образцах временных атрибутов по идентификатору атрибута.

> Если время существования TTL какого-либо атрибута не истекло, выдаются данные атрибута. В противном случае данные для этого атрибута в ответе не выдаются.

2.7.5 Объект «Лицо»

Лица – это изменяемые объекты, содержащие информацию об одном человеке.

В объекте «Лицо» хранятся следующие общие данные:

- Биометрический шаблон («descriptor»)
- Базовые атрибуты («basic_attributes»)
- Аватар («avatar»)
- Данные пользователя («user_data»)
- Внешний идентификатор («external_id»)
- Идентификатор события («event_id»)
- Идентификатор биометрического образца («sample_id»)
- Идентификатор списка («list_id»)
- Идентификатор учетной записи («account_id»)

См. раздел «Описание базы данных Faces» для получения дополнительной информации об объекте «Лицо» и данных, хранящихся в нем.

Данные атрибутов могут храниться в лице. Данные базовых атрибутов, данные БШ и информация о БО сохраняются в базе данных Faces и связаны с объектом «лицо».

При удалении лица, данные связанных атрибутов также удаляются.

- **Биометрический шаблон**. Необходимо указать БШ для лица, если необходимо использовать лицо для операций сравнения.

  Одно лицо нельзя связать с несколькими БШ.

- **Базовые атрибуты**: Базовые атрибуты могут использоваться для отображения информации в пользовательском интерфейсе.

Объект «Лицо» также может содержать идентификаторы биометрических образцов, используемых для создания атрибутов.

Описание общих полей событий:

- «user_data». Это поле может содержать любую информацию о человеке.
- «avatar». Аватар – это визуальное представление лица, которое можно использовать в пользовательском интерфейсе.

  Это поле может содержать URL-адрес внешнего изображения или биометрический образец, который используется в качестве аватара для лица.
- «external_id». Внешний идентификатор лица.

  Можно использовать внешний идентификатор для работы с внешними системами.

  Также можно использовать внешний идентификатор, чтобы указать, что несколько лиц принадлежат одному и тому же человеку. Для этих лиц устанавливается одинаковый внешний идентификатор при их создании.

  По запросу «faces» > «get faces» можно посмотреть все лица с одинаковым внешним идентификатором.
- «event_id». В этом поле может содержаться идентификатор события, в результате которого появилось это лицо.
- «list_id». В этом поле может содержаться идентификатор списка, с которым связано лицо.

  Лицо можно связать с одним или несколькими списками.
- «account_id» – идентификатор учетной записи, которой принадлежит лицо.
- «sample_id» – идентификатор биометрического образца. К лицу можно привязать один или несколько БО. Это должны быть БО, используемые для извлечения атрибутов. Все БО должны принадлежать одному человеку. БШ невозможно будет обновить до новой версии нейросети, если для лица не будут сохранены БО.

> Идентификаторы обычно не содержат персональную информацию. Однако они могут быть связаны с объектом, содержащим персональные данные.

2.7.5.1 Использование лица

Лица хранят информацию о людях, которые зарегистрированы в LP. Следовательно, они обычно требуются для верификации (полученный БШ лица сравнивается с БШ, прикреплённым к существующему лицу) и идентификации (входной БШ сравнивается с несколькими БШ лиц из указанного списка).

Если в системе нет сохраненных лиц, следующие операции с лицами невозможны:

- Сравнение по лицам и спискам, если лица указаны в качестве кандидатов или эталонов в запросе к ресурсу «/matcher/faces».
- Сравнение по лицам и спискам, когда лица указаны в качестве кандидатов в политике сравнения к ресурсу «/handlers».
- Задачи Cross-matching, когда лица указываются в качестве кандидатов или эталонов в запросе к ресурсу «/tasks/cross_match».
- Задачи Clustering, если лица заданы как фильтры для кластеризации в запросе к ресурсу «/tasks/clustering».
- Запросы на верификацию, если идентификаторы лиц задаются в параметрах запроса в запросе к ресурсу «/verifiers/{verifier_id}/verifications».
- Задачи ROC-curve calculation (ресурс «/tasks/roc»).

2.7.5.2 Создание и сохранение лиц

Лица могут быть созданы с помощью следующих запросов:

- «create face»

  Можно указать атрибуты для лица одним из следующих способов:

  - путем указания идентификатора временного атрибута;
  - путем указания БШ и базовых атрибутов (с БО или без них);
  - путем указания БШ (с БО или без них);
  - путем указания базовых атрибутов (с БО или без них);

  Последние три способа используются, когда нужно создать лицо с помощью данных, хранящихся во внешнем хранилище.

- «generate events». В используемом обработчике должен быть активирован параметр «storage_policy» > «face_policy» > «store_face».

2.7.5.3 Запросы для работы с лицами

Ниже приведены запросы для работы с лицами:

Запрос	Описание
«create face»	Создать лицо.
«get faces»	Получить информацию о лицах в соответствии с установленными фильтрами. Можно указать идентификатор списка в качестве фильтра для получения информации обо всех лицах, связанных со списком.
«delete faces»	Удалить несколько лиц по их идентификаторам.

Запрос	Описание
«get face count»	Получить информацию о количестве ранее созданных лиц в соответствии с установленными фильтрами.
«get count of faces with attributes»	Получить информацию о количестве ранее созданных лиц с прикрепленными атрибутами.
«get face»	Получить информацию о лице по его идентификатору.
«patch face»	Обновить поля «user_data», «external_id», «event_id», «avatar» уже созданного лица.
«remove face»	Удалить лицо по его идентификатору.
«check if face exists»	Проверить существует ли лицо с указанным идентификатором.

2.7.6 Объект «Список»

Список – это объект, в котором могут находиться лица, относящиеся к одной категории, например, клиенты или сотрудники.

В списках имеется поле «description», в котором могут храниться любые необходимые данные, позволяющие отличать списки друг от друга.

2.7.6.1 Использование списка

Можно добавлять лица в списки для разделения людей на группы.

> В списки можно добавлять только лица.

Идентификаторы списков обычно указываются для операций сравнения в качестве фильтра для лиц.

2.7.6.2 Запросы для работы со списками

Ниже приведены запросы для работы со списками:

Запрос	Описание
«create list»	Создать список.
«get lists»	Получить информацию обо всех ранее созданных списках в соответствии с фильтрами.
«delete lists»	Удалить несколько списков по их идентификаторам.

Запрос	Описание
«get list count»	Получить информацию о количестве ранее созданных списков в соответствии с установленными фильтрами.
«get list»	Получить информацию о списке по его идентификатору.
«check if list exists»	Проверить существует ли список с указанным идентификатором.
«update list»	Обновить поле «user_data» уже созданного списка.
«delete list»	Удалить список по его идентификатору.
«attach/detach faces to the list»	Прикрепить/открепить определенные идентификаторы лиц к списку.

2.7.7 Объект «Событие»

События используются при подходе «Параллельное выполнение запросов».

События – это неизменяемые объекты, которые содержат информацию об одном лице и/или теле. Их можно получить после обработки изображений с помощью обработчиков или создать вручную.

> В отличие от лиц, события невозможно изменить после создания. Единственным исключением является БШ, прикреплённый к событию. Он может быть обновлён на новую версию нейронной сети.

Обычно событие создается для каждого лица/тела, обнаруженного на изображении. Если обнаруженные лицо и тело принадлежат одному и тому же человеку, они сохраняются в одном событии.

В LP также есть возможность создавать новые события вручную без обработки с помощью обработчиков. Используется в тех случаях, когда логика заполнения полей событий должна отличаться от логики, используемой обработчиками. Например, если нужно извлечь БШ только для части, а не для всех обнаружений.

Событие может быть связано с БШ, хранящимся в базе данных Events.

> Поскольку в базе данных Events могут быть миллионы событий, рекомендуется использовать высокопроизводительную колоночную базу данных. Также можно использовать реляционную базу данных, но для этого понадобятся дополнительные настройки.

В объекте «Событие» хранятся следующие основные данные:

- «source». В этом поле может быть указан источник, из которого было получено лицо или тело человека. Значение указывается при выполнении запроса «generate events».

- «location». В этой группе параметров приведена информация о месте, где произошло событие. Значения указываются при выполнении запроса «generate events». В эту группу входят следующие поля:
  - «city»
  - «area»
  - «district»
  - «street»
  - «house_number»
  - «geo_position» – широта и долгота.
- «tag». Это поле содержит тег (или несколько тегов), применяемый при выполнении «conditional_tags_policy». Теги можно указать при выполнении запроса «create handler» или «generate events».
- «emotion». В этом поле отображается преобладающая эмоция, оцениваемая для лица. При необходимости можно отключить параметр «estimate_emotions» в ресурсе «/handlers» или «create verifier».
- «insert_time». В этом поле содержится время, когда лицо появилось в видеопотоке. Эти данные обычно предоставляются внешними системами.
- «top_similar_object_id». В этом поле содержится идентификатор наиболее похожего объекта.
- «top_similar_external_id». В этом поле содержится внешний идентификатор наиболее похожего кандидата (события или лица), с которым выполняется сравнение лица.
- «top_matching_candidates_label». В этом поле содержится метка, применяемая при выполнении «match_policy». Метки задаются в этой политике в ресурсе «/handlers».
- «face_id». События содержат идентификатор лица, появившегося в результате события.
- «list_id». События содержат идентификатор списка, с которым связано созданное лицо.
- «external_id». Этот внешний идентификатор указывается для лица, созданного при обработке запроса события. Значение указывается при выполнении запроса «generate events».
- «user_data». Это поле может содержать любую пользовательскую информацию. Указываются данные пользователя для лица, созданного при обработке запроса события. Значение указывается при выполнении запроса «generate events».
- базовые атрибуты «age», «gender» и «ethnic_group» можно сохранить в событии. Извлечение базовых атрибутов задается в «extract_policy» ресурса «/handlers».
- БШ лица и тела. События могут использоваться для операций сравнения, поскольку они содержат БШ.
- информация о лицах, телах и событиях, используемых для сравнения с событиями.

- информация о результатах обнаружения лиц и тел людей, включая:
  - идентификаторы созданных биометрических образцов.
  - информация об ограничивающих прямоугольниках обнаруженных лиц/тел.
  - «detect_time» – время обнаружения события лица/тела. Значение выдается от внешней системы.
  - «image_origin» – URL исходного изображения, на котором было обнаружено лицо/тело.

> Идентификаторы обычно не содержат персональную информацию. Однако они могут быть связаны с объектом с персональными данными.

См. раздел «Описание базы данных Events» для получения дополнительной информации об объекте «Событие» и данных, хранящихся в нем.

Агрегирование атрибутов для событий

Можно включить агрегирование атрибутов для входных изображений при создании события.

В соответствии с указанными настройками на входных изображениях обнаруживаются лица и тела. Если включено обнаружение лиц и должна быть выполнена агрегация БШ, то единый БШ создаётся на основе всех найденных лиц. Такая же логика используется и для создания агрегированного БШ тел. Кроме того, агрегируются базовые атрибуты, полученные значения определения Liveness, масок, эмоций, верхняя и нижняя части тела и аксессуары тела.

Вся информация об обнаруженном лице/теле и предполагаемых свойствах выдается в ответе отдельно для каждого изображения. При сохранении события в БД заносится агрегированные результаты. Информация о детекции лица/тела добавляется в базу данных Events в виде отдельной записи для каждого изображения из запроса.

При выполнении агрегации изображения в запросе к ресурсу «/handlers/{handler_id}/events» должны содержать только одно лицо/тело и это лицо/тело должны принадлежать одному и тому же человеку.

2.7.7.1 Использование событий

События необходимы для хранения информации о появлении людей в видеопотоке для последующего анализа. Внешняя система обрабатывает видеопоток и отправляет кадры или биометрические образцы с обнаруженными лицами и телами.

LP обрабатывает эти кадры или БО и создает события. Поскольку события включают статистику о времени обнаружения лиц, местоположении, базовых атрибутах и т.д., их можно использовать для сбора статистики об интересующем человеке или сбора общей статистики с использованием всех сохраненных событий.

События предоставляют следующие возможности:

- **Сравнение по событиям**

Поскольку в событиях хранятся БШ, они могут использоваться для сравнения. Можно сравнить БШ человека с существующими событиями для получения информации о передвижениях человека.

Для осуществления сравнения нужно установить события в виде эталонов или кандидатов для операций сравнения (см. раздел «Сравнение биометрических шаблонов»).

- **Уведомления через веб-сокеты**

  Можно получить уведомления о событиях через веб-сокеты. Уведомления отправляются в соответствии с заданными фильтрами. Например, при распознавании сотрудника в приложение турникета может быть отправлено уведомление, и турникет откроется.

  См. раздел «Отправка уведомлений через сервис Sender».

- **Сбор статистики**

  Статистику можно собирать по существующим событиям с помощью специального запроса. Ее можно использовать для:

  - Группирования событий по частоте или временным интервалам.
  - Фильтрации событий на основе их свойств.
  - Подсчета количества созданных событий в соответствии с фильтрами.
  - Определения минимального, максимального и среднего значения свойств для существующих событий.
  - Группирования существующих событий на основании заданных значений свойств.

  Сгенерированные события необходимо сохранить в базе данных для последующей сборки статистики.

  Дополнительную информацию по запросу «events» > «get statistics on events» см. в «APIReferenceManual.html».

- **Пользовательские данные**

  В события можно добавлять пользовательскую информацию, которую в дальнейшем можно использовать для фильтрации событий. Информация передается в формате JSON и записывается в базу данных Events.

  См. подробную информацию в разделе «Метаинформация события».

2.7.7.2 Создание и сохранение событий

События создаются при выполнении запроса к ресурсу «/handlers/{handler_id}/events». Для параметра «event_policy» > «store_event» необходимо установить значение «1» при создании обработчика с использованием ресурса «/handlers».

Для создания и сохранения событий вручную используйте ресурс «/handles/{handler_id}/events/raw».

Формат сгенерированного события аналогичен формату, выдаваемому с помощью ресурса «/handlers/{handler_id}/events». Поля «event_id» и «url» не указываются при создании запроса. Они возвращаются в ответе после создания события.

> Уведомления с использованием веб-сокетов отправляются при создании событий через ресурс «/handlers/{handler_id}/events».

2.7.7.3 Удаление событий

События удаляются только с помощью задачи Garbage collection. Необходимо установить фильтры для удаления всех событий, соответствующих этим фильтрам.

Чтобы удалить событие, нужно отправить запрос POST к ресурсу «/tasks/gc».

Также можно вручную удалить необходимые события из базы данных.

2.7.7.4 Получение информации о событиях

Можно получить информацию об имеющихся событих.

- С помощью запроса «events» > «get event» можно получить информацию о событии по его идентификатору.

> При удалении события система выдает ошибку при выполнении GET запроса.

- С помощью запроса «events» > «get events» можно получить информацию обо всех ранее созданных событиях в соответствии с установленными фильтрами. По умолчанию события фильтруются за последний месяц начиная с текущей даты. Если задан какой-либо из следующих фильтров, то фильтрация по умолчанию не будет использоваться.
  - список идентификаторов событий (event_ids);
  - нижнее пороговое значение идентификатора события (event_id__gte);
  - верхнее пороговое значение идентификатора события (event_id__lt);
  - нижнее пороговое значение времени создания (create_time__gte);
  - верхнее пороговое значение времени создания (create_time__lt).

2.7.7.5 Использование схемы «multipart/form-data» при генерации события

В запросе «generate events» можно отправить изображение, указать URL-адрес изображения или отправить «сырой» биометрический шаблон. Можно также использовать схему «multipart/form-data» для отправки нескольких изображений в запросе. Каждое из отправленных изображений должно иметь уникальное имя. Заголовок «Content-Disposition» должен содержать фактическое имя файла.

В запросе также можно указать следующие параметры с помощью схемы «multipart/form-data»:

- параметры ограничивающего прямоугольника лица или тела. Это позволяет обозначить определенную зону с лицом или телом на изображении
- время детекции изображения
- временная метка детекции относительно начала видеозаписи
- исходное изображение
- пользовательская метаинформация

Все вышеперечисленная информация будет записана в событие при его генерации.

2.7.8 Объект «Обработчик»

Обработчики используются при подходе «Параллельное выполнение запросов».

Обработчик – это объект, в котором хранятся точки входа для обработки изображений. Точки входа называются политиками. Они характеризуют процесс обработки изображения, следовательно, определяют сервисы LP, используемые для обработки. Обработчик создается с помощью запроса «create handler».

> При создании обработчика важно убедиться, что включены только необходимые политики. Лишние политики могут увеличивать время выполнения запросов и создавать гигабайты лишних данных на диске.

Политики обработчика

В таблице ниже представлены все существующие политики обработчиков. Каждая политика соответствует одному из сервисов LP, приведенных в столбце «Сервис».

Таблица 18: Политики обработчика

Политика	Описание	Сервис
detect_policy	Задаются оцениваемые параметры лица, тела и изображения.	Remote SDK
extract_policy	Задается необходимость извлечения БШ и базовых атрибутов (пол, возраст, этническая принадлежность). Также определяется пороговое значение качества БШ.	Remote SDK
match_policy	Задается массив списков для сравнения с текущим лицом и дополнительные фильтры для сравнения для каждого списка. Результаты сравнения можно использовать в «create_face_policy» и «link_to_lists_policy»	Python Matcher

Политика	Описание	Сервис
storage_policy	Активируется сохранение данных в базе данных для: • БО лиц и тел (подполитики «face_sample_policy» и «body_sample_policy») • исходные изображения (подполитика «image_origin_policy») • атрибуты (подполитика «attribute_policy») • лица (подполитика «face_policy») • события (подполитика «events_policy») Можно указать фильтры для сохранения объектов. Можно выполнять автоматическую привязку к спискам для лиц и устанавливать TTL для атрибутов. Можно включать и отключать отправку уведомлений через веб-собкеты или через вебхуки. См. подробную информацию в разделе «Отправка событий в сторонний сервис».	Image Store, Faces, Events, Handlers
conditional_-tags_policy	Задаются фильтры для присвоения тегов событиям.	Handlers

Если какие-то из политик не требуются, их можно отключить или не указывать в обработчике. Не указанные политики будут выполняться в соответствии с заданным для них значениями по умолчанию. Например, если сохранение БО не требуется, то следует явно отключить их в политике. Если просто не указать «storage_policy» в обработчике, то БО будут сохранены в соответствии с настройками по умолчанию.

Все доступные политики описаны в справочном руководстве сервиса API.

Фильтрация выходных данных

Во многих политиках есть фильтры. Фильтры могут указываться как явно в поле «filters», так и в параметрах с названием вида ”‹estimation›_states«. Если фильтрация будет выполнена в первой политике, то это приведет к прекращению выполнения последующих политик, поскольку они выполняются последовательно. В таком случае событие не будет сохранено в БД, а отфильтрованные объекты отобразятся в подблоке»filtered_detections”в теле ответа запроса на генерацию события.

Использование обработчиков

При использовании обработчиков можно:

- Указать параметры обнаружения лица/тела и предполагаемые параметры лица/тела (см. «Обработка исходного изображения и создание биометрических образцов»).

- Активировать извлечение базовых атрибутов и БШ (см. «Извлечение биометрических шаблонов и создание атрибутов»).
- Выполнить сравнение БШ (см. «Сравнение биометрических шаблонов») для получения результата сравнения и использования результата в качестве фильтров для политик.
- Настроить автоматическое создание лиц с помощью фильтров. Можно указать фильтры в соответствии с предполагаемыми базовыми атрибутами и результатами сравнения.
- Настроить автоматическое прикрепление созданных лиц к спискам на основании фильтров. Можно указать фильтры в соответствии с предполагаемыми базовыми атрибутами и результатами сравнения.
- Указать объекты, которые необходимо сохранить после обработки изображения.
- Настроить автоматическое добавление тегов к событию на основании фильтров. Можно указать фильтры в соответствии с предполагаемыми базовыми атрибутами и результатами сравнения.

Кроме того, доступна возможность обработки пакета изображений с помощью обработчика (см. «Задача Estimator»).

Существует три типа обработчиков (параметр «handler_type»):

- статический
- динамический
- lambda

2.7.8.1 Статический обработчик

Параметры обработчика такого типа указываются при его создании, а затем при генерации события или создании задачи Estimator указывается созданный идентификатор обработчика.

2.7.8.2 Динамический обработчик

Параметры обработчика такого типа указываются при запросе на генерацию события или на создание задачи Estimator. Динамические обработчики позволяют разделить технические параметры (пороговые значения и параметры качества, которые необходимо скрыть от пользователей frontend приложений) и бизнес-логику. Политики динамического обработчика задаются с помощью схемы «multipart/form-data» в запросе на генерацию события.

Верификаторы могут быть только статическими.

Пример использования динамических обработчиков:

Например, необходимо отделить пороговые значения углов положения головы от других параметров обработчика.

Можно сохранить пороговые значения во внешней базе данных и реализовать логику автоматической подстановки этих данных при создании событий (например, во frontend приложении).

Пользователь frontend приложения отправляет запросы на создание событий и указывает необходимые списки и другие параметры. Пользователь не знает пороговых значений и не может их изменить.

2.7.8.3 Lambda обработчик

Такой обработчик используется при работе с сервисом Lambda, предназначенным для работы с пользовательскими модулями, имитирующими функционал отдельного сервиса.

Все запросы будут отправлены в обработчик lambda с использованием «lambda_id».

Если пользовательская Handlers-lambda поддерживает возможность генерации события, то формат тела запроса и ответа к ресурсам «/handlers/{handler_id}/events» или «/tasks/estimator» будет соответствовать спецификации OpenAPI. Если же Handlers-lambda не поддерживает возможность генерации события, то должен быть использован запрос к ресурсу «/lambdas/{lambda_id}/proxy» с соответствующей формой запроса.

Логика поведения запроса полностью зависит от lambda, написанной пользователем.

См. подробную информацию в разделе «Сервис Lambda».

2.7.8.4 Запросы для работы с обработчиками

Ниже приведены запросы для работы с обработчиками:

Запрос	Описание
«create handler»	Создать обработчик.
«get handlers»	Получить информацию обо всех ранее созданных обработчиках в соответствии с установленными фильтрами.
«get handler count»	Получить информацию о количестве ранее созданных обработчиков в соответствии с установленными фильтрами.
«validate handler policies»	Проверить политики обработчика, прежде чем использовать их для создания или обновления обработчика.
«get handler»	Получить информацию об обработчике по его идентификатору.
«replace handler»	Обновить параметры уже созданного обработчика.
«check if handler exist»	Проверить существует ли обработчик с указанным идентификатором.
«remove handler»	Удалить обработчик по его идентификатору.

2.7.9 Объект «Верификатор»

Верификаторы используются при подходе «Параллельное выполнение запросов».

Сервис Handlers также обрабатывает запросы на создание верификаторов, необходимых для процесса верификации. Они создаются с помощью запроса «create verifier».

Верификатор содержит ограниченное количество политик обработчика — detect_policy, extract_-policy и storage_policy.

В верификаторе можно задать «verification_threshold».

Созданный верификатор следует использовать при отправке запросов в:

- ресурс «/verifiers/{verifier_id}/verifications». Можно задать ID объектов, по которым должна производиться верификация.

- ресурс «/verifiers/{verifier_id}/raw». Можно задать биометрические шаблоны в чистом виде в качестве эталонов и кандидатов для сравнения. Т. к. обрабатываются БШ в чистом виде, «verification_threshold» — это основной параметр, используемый из указанного верификатора.

Ответ включает в себя поле «status». Оно показывает, успешно ли прошла верификация для каждой пары сравниваемых объектов. Она успешна, если схожесть двух объектов превышает значение, заданное в «verification_threshold».

2.7.9.1 Запросы для работы с верификаторами

Ниже приведены запросы для работы с верификаторами:

Запрос	Описание
«create verifier»	Создать верификатор.
«get verifiers»	Получить информацию обо всех ранее созданных верификаторах в соответствии с установленными фильтрами.
«raw verification»	Выполнить верификацию «сырых» биометрических шаблонов.
«perform verification»	Выполнить верификацию по заданным объектам.
«count verifiers»	Получить информацию о количестве ранее созданных верификаторов в соответствии с установленными фильтрами.
«get verifier»	Получить информацию о верификаторе по его идентификатору.
«replace verifier»	Обновить параметры уже созданного верификатора.
«check if verifier exists»	Проверить существует ли верификатор с указанным идентификатором.

Запрос	Описание
«remove verifier»	Удалить верификатор по его идентификатору.

2.7.10 Прочие объекты

В сервисе Image Store можно хранить любые файлы в качестве объектов (например, видеофайл).

Загрузить файлы можно с помощью запроса «create objects». Байты файла должны быть указаны в теле запроса, а заголовок `Content-Type` должен содержать MIME-тип файла (например, `video/mp4`). Ответ на запрос «get object» содержит заголовок Content-Disposition. Этот заголовок содержит имя файла объекта вложения (например, `video_1.mp4`). Имя файла генерируется на основе `object_id` и MIME-типа.

Если MIME-тип файла не получилось определить, то расширение файла будет установлено как `.bin`.

2.8 Отправка уведомлений через сервис Sender

Можно отправлять уведомления о созданных событиях сторонним приложениям через веб-сокеты. Например, можно сконфигурировать LP для отправки уведомлений на мобильный телефон о прибытии VIP-гостей.

Когда LP создает новое событие, оно добавляется в специальную базу данных. Затем событие можно отправить в сервис Sender, если он включен.

Стороннее приложение должно быть подключено к сервису Sender через веб-сокет. Фильтр интересующих событий необходимо устанавливать для каждого приложения. Таким образом, клиент будет получать уведомления только об интересующих событиях.

Уведомление о новом событии отправляется сервисом Sender.

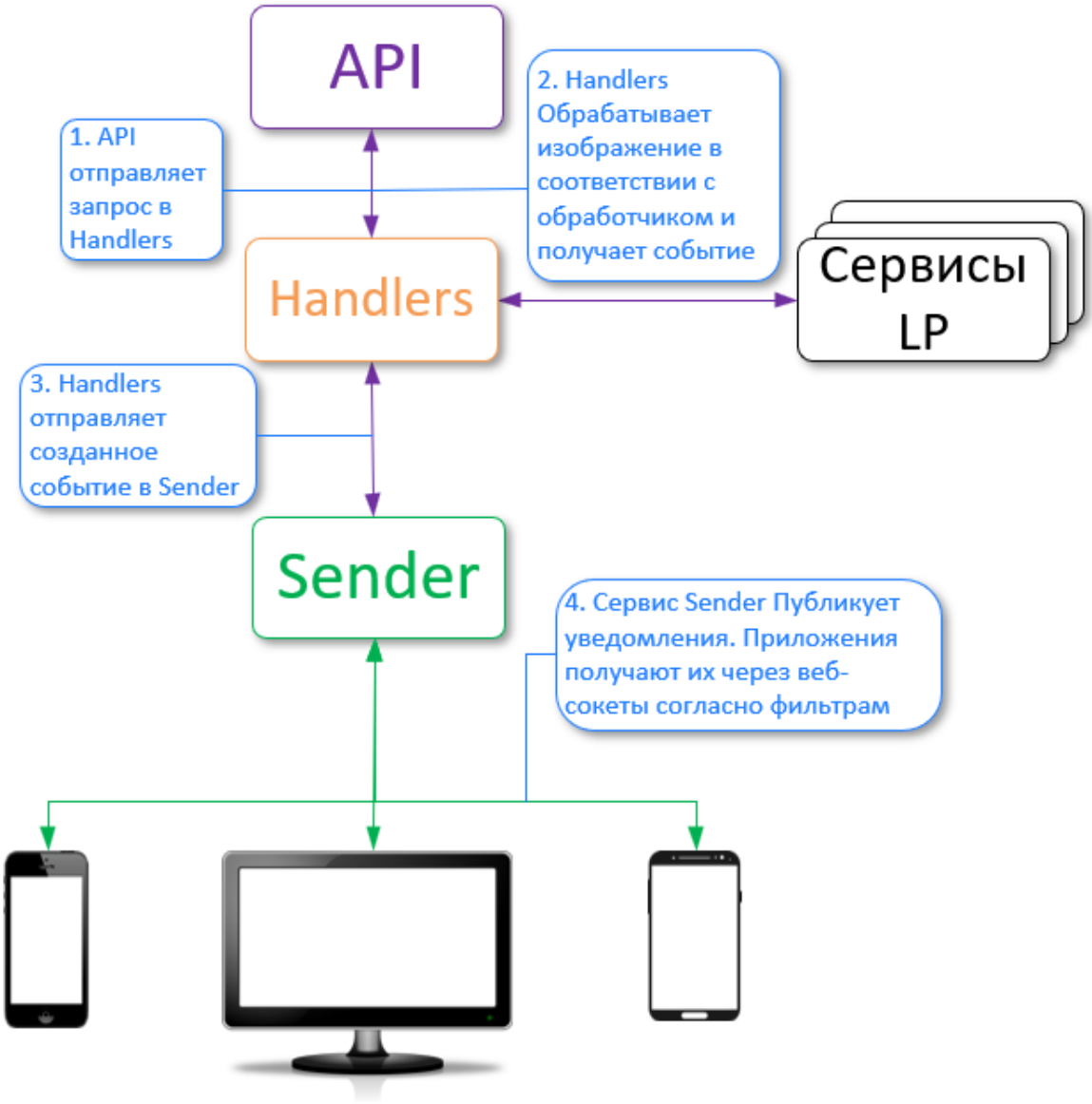


Рис. 12: Работа сервиса Sender

Для более подробной информации о работе сервиса Sender см. раздел «Сервис Sender».

2.9 Общие сведения сервиса Licenses

Сервис Licenses предоставляет информацию об условиях лицензии сервисам LP.

См. раздел «Сервис Licenses» для более подробной информации.

2.10 Общие сведения сервиса Admin

Сервис Admin обеспечивает инструменты для административных процедур.

Все запросы сервиса Admin описаны в спецификации OpenAPI сервиса Admin.

См. раздел «Сервис Admin» для более подробной информации о сервисе Admin.

2.11 Общие сведения сервиса Configurator

Сервис Configurator включает в себя настройки всех сервисов LP. Таким образом он обеспечивает возможность конфигурации всех сервисов в одном месте.

См. раздел «Сервис Configurator» для более подробной информации о сервисе Configurator.

2.12 Общие сведения сервиса Tasks

Задачи предоставляют дополнительные возможности для обработки больших объемов данных.

Чем больше размер массива обрабатываемых данных, тем больше времени занимает выполнение задачи. Когда задача создана, в результатах запроса отображается идентификатор задачи.

См. раздел «Сервис Tasks» для более подробной информации об обработке задач.

Задача Clustering

Задача Clustering дает возможность группировать события и лица, принадлежащие одному и тому же человеку, в кластеры.

Можно задать фильтры для выбора обрабатываемых объектов.

С помощью задачи Clustering можно, например, получать все ID событий, принадлежащих одному и тому же человеку и произошедших в течение указанного периода времени.

См. запрос «task processing» > «clustering task» для получения более подробной информации о создании запроса на задачу Clustering.

См. раздел «Задача Clustering» для более подробной информации об этой задаче.

Задача Reporter

Задача Reporter позволяет получать отчет в формате CSV с расширенной информацией об объектах, объединенных в кластеры.

Можно выбрать столбцы, которые следует добавить в отчет. Также можно получить в ответе изображения, соответствующие каждому из ID в каждом кластере.

См. запрос «task processing» > «reporter task» для более подробной информации о создании запроса на задачу Reporter.

См. раздел «Задача Reporter» для более подробной информации об этой задаче.

Задача Exporter

Задача Exporter позволяет получать данные по событиям и/или лицам и экспортировать их из LP в CSV-файл.

В строках файла содержится информация о запрашиваемых объектах и соответствующие биометрические образцы (если их сохранение указано в запросе).

См. запрос «task processing» > «exporter task» для более подробной информации о создании запроса на выполнение задачи Exporter.

См. раздел «Задача Exporter» для более подробной информации об этой задаче.

Задача Cross-matching

Cross-matching (перекрестное сравнение) означает, что большое количество эталонов можно сравнить с большим количеством кандидатов. Таким образом, каждый эталон сравнивается с каждым кандидатом.

Как эталоны, так и кандидаты задаются с помощью фильтров для лиц и событий.

См. запрос «task processing» > «cross-matching task» в справочном руководстве сервиса API для более подробной информации о создании запроса на выполнение задачи Cross-matching.

См. раздел «Задача Cross-matching» для более подробной информации об этой задаче.

Задача Linker

Задача Linker позволяет:

- осуществлять привязку существующих лиц к спискам;
- создавать лица из существующих событий и привязывать их к спискам.

Лица с привязкой выбираются в соответствии с заданными фильтрами.

См. запрос «task processing» > «linker task» в справочном руководстве сервиса API для более подробной информации о создании запроса на выполнение задачи Linker.

См. раздел «Задача Linker» для более подробной информации об этой задаче.

Задача Estimator

Задача Estimator позволяет выполнять пакетную обработку изображений с использованием указанных политик.

В теле запроса можно указать `handler_id` уже существующего статического или динамического обработчика. Для `handler_id` динамического обработчика доступна возможность задания требуемых политик. Кроме того, в запросе можно создать статический обработчик с указанием политик.

Ресурс может принимать в обработку пять типов источников с изображениями:

- ZIP-архив

- S3-подобное хранилище
- Сетевой диск
- FTP-сервер
- Сетевая файловая система Samba

См. запрос «task processing» > «estimator task» в справочном руководстве сервиса API для более подробной информации о создании запроса на выполнение задачи Estimator.

См. раздел «Задача Estimator» для более подробной информации об этой задаче.

2.13 Backport

В LP 5 сервисы Backport (ретроподдержка) — это механизм, позволяющий принимать запросы в формате предыдущих версий LP, но обрабатывать их в новой версии.

Ретроподдержка в LUNA PLATFORM 3 и LUNA PLATFORM 4 реализуется посредством сервисов Backport 3 и Backport 4 соответственно.

Ретроподдержка позволяет отправлять запросы, аналогичные запросам из LUNA PLATFORM 3 и LUNA PLATFORM 4, и получать отклик в требуемом формате.

При использовании ретроподдержки есть некоторые нюансы.

- Миграция данных в ретроподдержку достаточно сложная.

  Структуры баз данных LUNA PLATFORM 3 и LUNA PLATFORM 4 отличаются от структуры базы данных LUNA PLATFORM 5. Поэтому для переноса данных необходимо выполнить дополнительные шаги, и в процессе могут возникать ошибки.

- Ретроподдержка имеет много ограничений.

  Не вся логика из LP 3 и LP 4 может поддерживаться в ретроподдержке. Смотрите раздел «Ограничения при работе с сервисами Backport» для получения дополнительной информации.

- Ретроподдержка ограничена доступными функциями и не больше разрабатывается.

  Новые функции и ресурсы LUNA PLATFORM 5 не поддерживаются при использовании ретроподдержки и никогда не будут поддерживаться. Таким образом, эти сервисы следует использовать только в том случае, если невозможно выполнить полную миграцию на LUNA PLATFORM 5 и никаких новых функций не требуется.

Пример использования:

К примеру, есть интерфейсное приложение, отправляющее запросы в LUNA PLATFORM 3.

При использовании сервиса Backport 3 запросы в LP 3 принимаются сервисом, форматируются и отправляются в LUNA PLATFORM 5 в формате, соответствующем спецификации API. LP 5 обрабатывает этот запрос и отправляет отклик сервису Backport 3, который приводит все входные результаты к формату LP 3 и отправляет отклик.

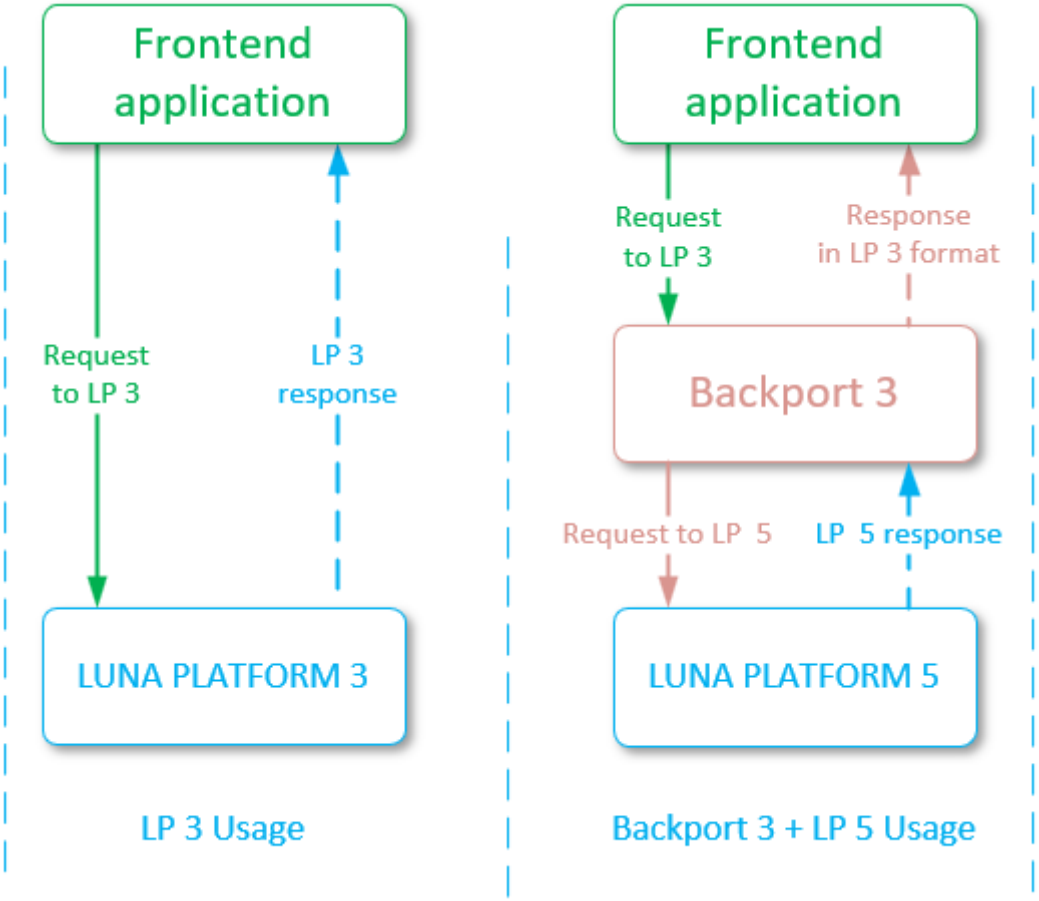


Рис. 13: LP3 vs Backport 3 и LP 5

2.13.1 Ограничения при работе с сервисами Backport

Так как таблицы сохраненных данных и сами данные различны в разных версиях LP, существуют особенности и ограничения при выполнении запросов. Информация об этих особенностях и ограничениях представлена в разделах «Backport 3» и «Backport 4» данного документа.

Сервисы Backport и API в LP 5 можно использовать одновременно с соблюдением следующих ограничений:

- При использовании сервисов Backport рекомендуется использовать разные аккаунты для запросов в сервисы Backport и API в LUNA PLATFORM 5.
- Следует выполнять запросы администратора, которые не используют ID аккаунта, только напрямую в LUNA PLATFORM 5.

2.14 Ресурс sdk

Ресурс sdk позволяет напрямую обращаться к сервису Remote SDK для обнаружения лиц и/или тел и оценки атрибутов входных изображений. После выполнения запроса полученные данные не сохраняются в базе данных и Image Store, и возвращаются только в ответе.

Ресурс sdk сочетает в себе возможности других ресурсов, а также предоставляет дополнительные возможности. Например, с помощью запроса «/sdk» можно выполнять следующие функции:

— оценивать наличие очков на изображении; — оценивать Liveness; — оценивать параметры лиц/тел; — агрегировать параметры лица/тела; — создавать биометрический образец лица и возвращать его биометрический шаблон в формате SDK, закодированном в Base64; — создавать биометрический образец тела и возвращать его биометрический шаблон в формате SDK, закодированном в Base64; — извлекать биометрические шаблоны лица и тела и возвращать их в ответе; — устанавливать порог оценки качества биометрического шаблона для фильтрации изображений, не подходящих для дальнейшей обработки; — фильтровать по наличию маски; — другие.

Биометрический шаблон лица или тела в формате SDK, закодированного в Base64 можно использовать в запросах на сравнение «сырых» биометрических шаблонов.

3 Аккаунты, токены и способы авторизации

3.1 Аккаунт

Аккаунт необходим для разграничения областей видимости объектов для конкретного пользователя. Наличие аккаунта требуется для выполнения большинства запросов. Каждый созданный аккаунт имеет свой собственный уникальный идентификатор «account_id». Все данные аккаунтов сохраняются в БД сервиса Accounts под этим идентификатором.

Аккаунт можно создать с помощью POST запроса «create account» к сервису API, либо с помощью сервиса Admin. При создании аккаунта необходимо указать следующие данные: электронная почта (login), пароль (password) и тип аккаунта (account_type).

3.1.1 Тип аккаунта

Тип аккаунта определяет, какие данные доступны пользователю.

Существует три типа аккаунтов:

- user — тип аккаунта, с помощью которого можно создавать объекты и использовать только данные своего аккаунта.
- advanced_user — тип аккаунта, для которого доступны права, аналогичные «user», а также есть доступ к данным всех аккаунтов. Доступ к данным других аккаунтов означает возможность получать данные (запросы GET), проверять их наличие (запросы HEAD) и выполнять запросы на сравнение по данным других аккаунтов.
- admin — тип аккаунта, для которого доступны права, аналогичные «advanced_user», а также есть доступ к сервису Admin.

В сервисе API можно работать со всеми типами аккаунтов, но создать можно только аккаунты типа «advanced_user» и «user», в то время как в сервисе Admin можно создать все три типа.

По умолчанию в системе существует аккаунт типа «admin» и логином и паролем по умолчанию `root@visionlabs.ai/root`.

С помощью заголовка «Luna-Account-Id» в запросе «create account» можно задать желаемый идентификатор аккаунта. Также его необходимо использовать в случае, если необходимо сохранить возможность работы с данными, созданными в версиях LP до 5.30.0 с помощью указания идентификатора «account_id» в заголовке «Luna-Account-Id». Таким образом, использование данного параметра привяжет старый «account_id» к создаваемому аккаунту (см. подробную информацию о миграции в разделе «Миграция аккаунтов» в руководстве по обновлению LUNA PLATFORM).

В ответе на запрос на создание аккаунта выдается «account_id». После создания аккаунта можно использовать этот идентификатор для авторизации LunaAccountIdAuth или использовать авторизацию авторизацию BasicAuth (авторизация по по логину/паролю).

3.2 Токен

Токен привязывается к существующему аккаунту с любым типом («user», «advanced_user», «admin») и позволяет наложить расширенные ограничения на выполняемые запросы. Например, при создании токена можно дать пользователю разрешение только на создание и изменение всех списков и лиц, или можно запретить использование определенных обработчиков, указав их идентификатор.

Токен создается с помощью запроса «create token» к сервису API.

Для каждого аккаунта можно создать неограниченное количество токенов. Токен и все его ограничения сохраняются в БД и привязываются к аккаунту по параметру «account_id». Нельзя привязать один токен к разным аккаунтам. Для создания токенов с одинаковыми разрешениями в рамках разных аккаунтов рекомендуется сохранить шаблон тела запроса для создания аккаунта и использовать его.

Нет необходимости в использовании и токена и аккаунта. При работе с токенами можно ограничить доступ к ресурсу «/accounts» с помощью внешних средств.

Для токена в любой момент можно выдать дополнительные ограничения с помощью запроса «replace token» или можно отозвать токен с помощью запроса «delete token». В этом случае токен будет удален из БД и больше не может быть использован для авторизации.

При создании токена требуется задать следующие параметры:

— expiration_time – время окончания действия токена в формате RFC 3339. Можно указать бесконечное время действия токена с помощью значения «null» — permissions – действия, которые может выполнять пользователь (см. «Разрешения, задаваемые в токене»)

Для токена также возможно указать видимость токеном данных других аккаунтов с помощью параметра «visibility_area» (см. раздел «Просмотр данных других аккаунтов»).

В ответе на запрос на создание токена выдается «token_id» и закодированный в Base64 JWT токен. После создания токена можно использовать полученный JWT токен для авторизации BearerAuth.

3.2.1 Разрешения, задаваемые в токене

Для создаваемого токена доступны следующие разрешения:

Название разрешения	Описание разрешения	Права
account	права на использование аккаунта	view
face	права на использование лица	creation, view, modification, deletion, matching
list	права на использование списка	creation, view, modification, deletion

Название разрешения	Описание разрешения	Права
event	права на использование события	creation (только запрос «save event»), view, matching
attribute	права на использование атрибута	creation, view, modification, deletion, matching
handler	права на использование обработчика	creation, view, modification, deletion
verifier	права на использование верификатора	creation, view, modification, deletion
task	права на использование задачи и расписания задачи	creation, view, modification, deletion
face_sample	права на использование БО лица	creation, view, deletion
body_sample	права на использование БО тела	creation, view, deletion
object	права на использование объекта	creation, view, deletion
image	права на использование изображения	creation, view, deletion
token	права на использование токена	view, modification, deletion
resources	права на использование ресурсов	«/iso», «/sdk», «/liveness»
emit_events	права на выполнение запросов «generate events»	(см. ниже)
lambdas	права на использование lambda	creation, view, modification, deletion
verify	права на выполнение запросов «perform verification»	(см. ниже)
video_stream	права на использование потоков	creation, view, modification, deletion
video_group	права на использование групп	creation, view, modification, deletion
video_analytic	права на просмотр аналитик	view
пользовательскс	пользовательские права	creation, view, modification, deletion

> Пользовательские права предназначены для проксирования запросов LUNA Streams через LUNA API. В остальных случаях использование пользовательских прав не будет иметь эффекта. См. раздел «Проксирование запросов в LUNA Streams через LUNA API» в руководстве администратора FaceStream.

> Ресурсы «/iso», «/sdk» и «/liveness» по умолчанию не требуют никакой авторизации.

Значение **[]** означает отсутствие всех разрешений.

Разрешения «emit_events» и «verify» позволяют указать можно ли выполнять запросы к ресурсу «generate events» или «perform verification», а также поместить идентификаторы обработчиков или верификаторов в черный или белый списки. Если идентификаторы «handler_id» или «verifier_id» присутствуют в черном списке, то только их использование будет запрещено. Если же идентификаторы присутствуют в белом списке, то только их использование будет разрешено. Максимальное количество идентификаторов в списках — 100. При использовании разрешения «emit_events» у пользователя не должно быть прав «creation» и «modification» на использование обработчика.

> Если выдано разрешение «emit_events», то во время генерации события будут созданы все необходимые объекты независимо от разрешений, задаваемых в токене. Например, разрешение типа «faces» регулирует работу с лицами только в запросах к ресурсам ”faces/*«, но не влияет на создание лица во время генерации события. Таким образом, при использовании обработчика с параметром»store_face”и отсутствии права «creation» для «faces», все равно будет создано лицо.

> Если выдано разрешение «verify», то во время выполнения верификации будут созданы все необходимые объекты независимо от разрешений, задаваемых в токене. Например, разрешение типа «sample» не влияет на создание биометрического образца во время выполнения верификации. Таким образом, при использовании верификатора с параметром «store_sample» и отсутствии права «creation» для «samples», все равно будет создан биометрический образец.

При задании разрешений следует учитывать следующие особенности:

- тип «modification» означает выполнение PATCH и PUT запросов
- прикрепление/откреплление к списку является разрешением типа «modification»
- удаление списка с включенной настройкой «with_faces» (запрос «delete lists») требует разрешения «face» > «deletion»
- прикрепление лица к списку в запросе на создания лица (запрос «create face») требует разрешения «list» > «modification»
- при выполнении сравнения необходимо иметь разрешение «matching» для соответствующих объектов (event, face, attribute)
- запросы на получение статистики по событиям (запрос «get statistics on events») требуют разрешения «event» > «view»
- разрешение «event» > «create» дает право только на создание события с помощью запроса «save event». Для генерации события необходимо воспользоваться разрешением «emit_events» (см. выше)
- разрешение «event» > «create» не распространяется на верификаторы

См. таблицу зависимости разрешений токенов от ресурсов сервиса API в руководстве разработчика.

3.3 Просмотр данных других аккаунтов

Данные другого аккаунта могут быть просмотрены если:

- тип аккаунта установлен как «advanced_user» или «admin»
- при создании токена был указан параметр «visibility_area» = «all».

> Для типа аккаунта «user» нельзя задать «visibility_area» = «all».

Если установлен тип аккаунта «advanced_user»/«admin» и создается токен с установленным параметром «visibility_area» = «account», то при авторизации по токену (BearerAuth) пропадет возможность смотреть данные других аккаунтов, однако при авторизации по логину/паролю (BasicAuth), такая возможность останется.

Если установлен тип аккаунта «advanced_user»/«admin» и создан токен с установленным параметром «visibility_area» = «all», а затем тип аккаунта изменяется на «user» (с помощью запроса «patch account»), то при попытке выполнить запрос на просмотр данных других аккаунтов с помощью токена будет выдана ошибка.

При использовании авторизации LunaAccountIdAuth область видимости регулируется с помощью заголовка «Luna-Account-Id», что означает, что видны только данные аккаунта, связанного с этим идентификатором.

> Для верификаторов недоступна возможность использования области видимости данных всех аккаунтов. При значении параметра «visibility_area» = «all», будет видны данные только своего аккаунта.

3.4 Типы авторизации для доступа к ресурсам

В LUNA PLATFORM доступно три типа авторизации:

- **BasicAuth**. Авторизация по логину и паролю (задаются во время создания аккаунта).
- **BearerAuth**. Авторизация по JWT токену (выдается после создания токена).
- **LunaAccountIdAuth**. Авторизация по заголовку «Luna-Account-Id», в котором указывается сгенерированный после создания аккаунта «account_id». **Данный способ считается устаревшим и не рекомендуется к использованию**.

Авторизация LunaAccountIdAuth может быть включена с помощью настройки «ALLOW_LUNA_ACCOUNT_AUTH_HEADER» в секции «OTHER» настроек сервиса API в Configurator (по умолчанию отключена).

В спецификации OpenAPI заголовок «Luna-Account-Id» помечен словом **Deprecated**.

> Нет необходимости в использовании всех трех типов авторизации при выполнении запросов. Необходимо выбрать предпочтительный способ в зависимости от требуемых задач.

Если в запросе не указан тип авторизации, будет выдаваться ошибка с кодом состояния 403.

3.5 Проверка актуальности учетной записи

С помощью ресурса «verify credentials» можно проверять существующие учетные записи по одному из типов:

- верификация логина/пароля. Если верификация успешна – вернется идентификатор аккаунта и его тип.
- верификация токена. Если верификация успешна, вернется тип аккаунта и все разрешения для токена.
- верификация идентификатора аккаунта. Если верификация успешна – вернется тип аккаунта.

3.6 Логирование информации об аккаунтах

При выполнении любого запроса, в логах сервиса API выводится информация о соответствующем аккаунте. Данный функционал позволяет определить кто именно выполнил тот или иной запрос. Это может потребоваться для информационной безопасности и администраторов систем.

Если запрос был выполнен с авторизацией типа BasicAuth или LunaAccountIdAuth, то в логах будет выдано следующее сообщение:

```
  Request invoked by user (account_id: '270531af-e52e-4538-9181-628d9900a0db')
```

Если запрос был выполнен с авторизацией типа BearerAuth, то в логах будет выдано следующее сообщение:

```
  Request invoked by user (account_id: '270531af-e52e-4538-9181-628d9900a0db'
     token_id: 'd57e16f5-e243-47d2-aa85-8b200c12d86f')
```

Если запрос был выполнен без авторизации, то в логах будет выдано следующее сообщение:

```
  Request invoked by user (account_id: null)
```

В логах сервиса Accounts дополнительно выводится информация о создании токенов конкретными пользователями:

```
  User with account_id: '270531af-e52e-4538-9181-628d9900a0db' create token: '
     d57e16f5-e243-47d2-aa85-8b200c12d86f'
```

Логирование информации о создании токенов позволяет отследить откуда появился токен и какому пользователю он принадлежал, даже после его удаления.

4 Оцениваемые данные

В данном разделе перечислены основные параметры лиц, тел и изображений, оцениваемые LUNA PLATFORM 5, и способы их получения.

Получить параметры можно с помощью различных средств и ресурсов. В основном оценивание параметров выполняется с помощью следующих способов:

1. **Извлечение пола и возраста из изображения лица**. Пол и возраст принадлежат понятию **базовые атрибуты**.

   Для извлечения базовых атрибутов по изображению лица используются ресурсы «/extractor», «/sdk» и политики «extract_policy» ресурсов «/handlers» и «/verifiers».

   Для извлечения этих параметров с помощью ресурса «/extractor» необходимо предварительно создать биометрический образец изображения с помощью ресурса «/detector». В ответ на запрос к ресурсу «/extractor» будут выданы пол и возраст человека. Извлеченные данные имеют TTL (время существования) и удалятся из базы данных по истечении указанного периода.

   > См. подробное описание извлечения базовых атрибутов в разделе «Извлечение биометрических шаблонов и создание атрибутов».

   При оценивании параметров с помощью ресурса «/sdk» нужно отправить исходное изображение в LUNA PLATFORM 5 и указать в параметрах запроса параметр «estimate_basic_attributes». В ответ на запрос будут получены пол и возраст человека. Данные параметры не будут сохранены в базу данных.

   Для извлечения базовых атрибутов лица с помощью запросов «/handlers» и «/verifiers» необходимо использовать параметр «extract_basic_attributes» политики «extract_policy».

   > Оценка пола и возраста человека также доступна по изображению тела. Такой способ проверки не является точным и выполняется с помощью оценки параметров тела (см. ниже «Выполнение оценки параметров тела»).

2. **Выполнение оценки параметров лица и изображения**.

   Для оценивания параметров используются различные ресурсы. В основном используются ресурсы «/detector», «/sdk», «/handlers» и «/verifiers».

   При оценивании параметров с помощью ресурса «/detector» нужно отправить исходное изображение в LUNA PLATFORM 5 и указать в параметрах запроса оценивание необходимых параметров лица или изображения. В ответ на запрос будет создан биометрический образец лица и выданы указанные параметры. Оцененные параметры не будут сохранены в базу данных.

Способ получения параметров с помощью ресурса «/sdk» аналогичен вышеописанному способу, однако биометрический образец не будет создан. Оцененные параметры также не сохранятся в базу данных.

Для оценивания параметров с помощью ресурсов «/handlers» и «/verifiers» необходимо использовать политику «detect_policy» с указанием необходимых параметров.

3. **Выполнение оценки параметров тела**.

   Возможность выполнения оценки параметров тела регулируется особым параметром в лицензионном ключе LUNA PLATFORM 5.

   Для оценивания параметров тела используются ресурсы «/sdk» и «/handlers». Использование данных ресурсов аналогично выполнению оценки параметров лица и изображения (см. выше).

4. **Выполнение проверки параметров лица и изображения** на соответствие стандарту ISO/IEC 19794-5:2011 или по нестандартным условиям.

   Возможность выполнения таких проверок регулируется особым параметром в лицензионном ключе LUNA PLATFORM 5.

   Для выполнения проверки используются ресурсы «/iso», «/detector» (параметр «estimate_face_quality») и группа проверок «face_quality», политики «detect_policy», запросов «/handlers» и «/verifiers».

   В ответах на запросы отображается общий результат прохождения всех проверок («0» или «1»), а также результаты каждой проверки.

   См. подробное описание данной функциональности в разделе «Проверка изображений».

5. **Выполнение оценки Liveness**.

   Возможность выполнения такой оценки регулируется особым параметром в лицензионном ключе LUNA PLATFORM 5.

6. **Выполнение оценки количества людей на изображении**

Все возвращаемые значения и формат ответа зависят от ресурса, где выполняется оценка.

> Обратите внимание, что для получения результатов при отправке запросов к ресурсам «/handlers» или «/verifiers» необходимо сгенерировать событие и выполнить верификацию по заданным обработчикам. См. раздел Объект «Обработчик» для получения более подробной информации о работе с данными ресурсами.

4.1 Пол и возраст по изображению лица

Данная оценка позволяет определить базовые атрибуты (пол и возраст человека) на изображении лица.

> Подробная информация о базовых атрибутах приведена в разделе Объект «Атрибут».

Ресурсы, в которых выполняется оценка:

- «/extractor»

  Название оценки — «extract_basic_attributes».

- «/handlers»

  Название оценки — «policies» > «extract_policy» > «extract_basic_attributes».

- «/verifiers»

  Название оценки — «policies» > «extract_policy» > «extract_basic_attributes».

- «/sdk»

  Название оценки — «estimate_basic_attributes».

4.2 Параметры лиц

4.2.1 Атрибуты глаз

Данная оценка определяет следующие положения для каждого глаза:

- «open» (открытый);
- «closed» (закрытый);
- «occluded» (чем-то перекрыт).

Изображения низкого качества или изображения, на которых перекрыты глаза (например, очки, волосы, перекрытие руками), попадают в категорию «occluded».

Также определяются контрольные точки радужной оболочки. Для каждого глаза выдается массив из 34 контрольных точек.

В ресурсах «/iso», «/detector» (средство проверки изображения) и «detect_policy» > «face_quality» также выполняется сравнение оцененного значения с порогом (в соответствии с ISO или нестандартным порогом).

Ресурсы, в которых выполняется оценка:

- «/detector»

  Название оценки — «estimate_eyes_attributes».

- «/handlers»

  Название оценки — «policies» > «detect_policy» > «estimate_eyes_attributes».

- «/verifiers»

  Название оценки — «policies» > «detect_policy» > «estimate_eyes_attributes».

- «/sdk»

  Название оценки — «estimate_eyes_attributes».

Определение атрибутов глаз с помощью средств для проверки изображения:

- «/iso» и «/detector» (см. разделы «7.2.3 Expression», пункт «a», «7.2.11 Visibility of pupils and irises» и «7.2.13 Eye patches» в стандарте ISO/IEC 19794-5:2011)

  Названия проверок — «left_eye», «right_eye».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Названия проверок — «left_eye», «right_eye».

Допустимые значения прохождения проверок:

Изображение проходит проверку если попадает в допустимое значение соответствующего порога:

Параметр	Допустимое значение
«left_eye» > «threshold»	[«open»]
«right_eye» > «threshold»	[«open»]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.2 Контрольные точки лица

Существует две оценки контрольных точек лица:

- оценка по 5 контрольным точкам лица
- оценка по 68 контрольным точкам лица

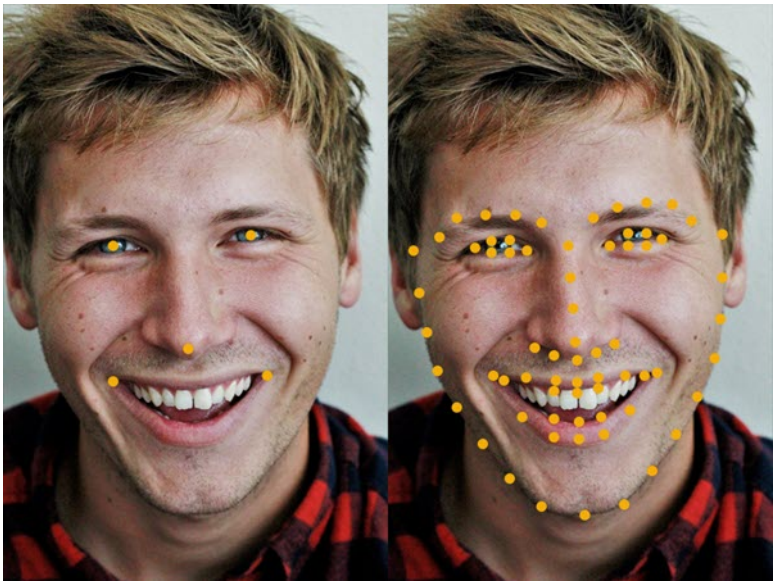


Рис. 14: Оценка по 5 контрольным точкам лица (слева), оценка по 68 контрольным точкам лица (справа)

Контрольные точки используются в различных целях, например, при создании биометрического образца, при извлечении биометрического шаблона и др. См. более подробное описание контрольных точек лица в документации SDK.

- «/detector»

  Название оценки — «detect_landmarks68».

- «/handlers»

  Название оценки — «policies» > «detect_policy» > «detect_landmarks68».

- «/verifiers»

  Название оценки — «policies» > «detect_policy» > «detect_landmarks68».

- «/sdk»

  Название оценки — «estimate_landmarks5».

  Название оценки — «estimate_landmarks68».

4.2.3 Расстояние между центрами глаз

Примечание. Для данной оценки невозможно использовать биометрический образец в качестве входного изображения.

Доступна возможность оценить расстояние между центрами глаз в пикселях. Также выполняется сравнение оцененного значения с порогом (в соответствии с ISO или нестандартным порогом).

Такую оценку можно выполнить только с помощью средств для проверки изображения:

- «/iso» и «/detector» (см. раздел «5.6.5 Eye and nostril centre Landmark Points» в стандарте ISO/IEC 19794-5:2011)

  Название проверки — «eye_distance».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Название проверки — «eye_distance».

Допустимый диапазон прохождения проверки:

Изображение проходит проверку если попадает в допустимый диапазон соответствующего порога:

Параметр	Допустимый диапазон
«eye_distance» > «threshold»	[90…inf]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.4 Эффект «красных глаз»

Данная оценка определяет наличие эффекта «красных глаз», где:

- «0» — на изображении лица нет эффекта «красных глаз»;
- «1» — на изображении лица присутствует эффект «красных глаз».

Также выполняется сравнение оцененного значения с порогом (в соответствии с ISO или нестандартным порогом).

Требования к изображению:

Для корректных результатов проверки должны быть выполнены нижеописанные требования.

В таблице ниже приведены требования к параметрам качества:

Параметр	Требуемый диапазон
«illumination»	[0.61…1]
«specularity»	[0.57…1]
«blurriness»	[0.5…1]
«dark»	[0.1…1]
«light»	[0.1…1]

В таблице ниже приведено требование к естественности освещения:

Параметр	Требуемый диапазон
«natural_light»	[0.5…1]

Ресурсы, в которых выполняется оценка:

Оценка эффекта «красных глаз» доступна только с помощью средств для проверки изображения:

- «/iso» и «/detector» (см. раздел «7.3.4 Unnatural colour» в стандарте ISO/IEC 19794-5:2011)

  Название проверки — «red_eyes».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Название проверки — «red_eyes».

Допустимое значение прохождения проверки:

Изображение проходит проверку если попадает в допустимое значение соответствующего порога:

Параметр	Допустимое значение
«red_eyes» > «threshold»	«1»

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.5 Направление взгляда

Данная оценка определяет направление взгляда. Направление взгляда определяется следующими параметрами:

- угол наклона взгляда вверх/вниз (pitch);
- угол поворота взгляда вправо/влево (yaw);

Положительное значение угла pitch означает направление взгляда вверх, а отрицательное значение — направление взгляда вниз.

Положительное значение угла yaw означает направление взгляда вправо, а отрицательное значение — направление взгляда влево.

Нулевое положение соответствует направлению взгляда, перпендикулярному плоскости лица, где ось симметрии параллельна вертикальной оси камеры.

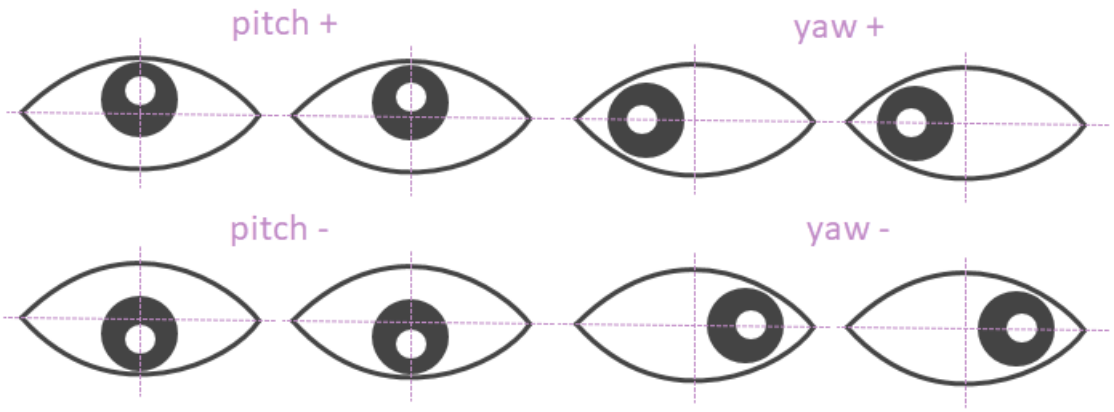


Рис. 15: Направление взгляда

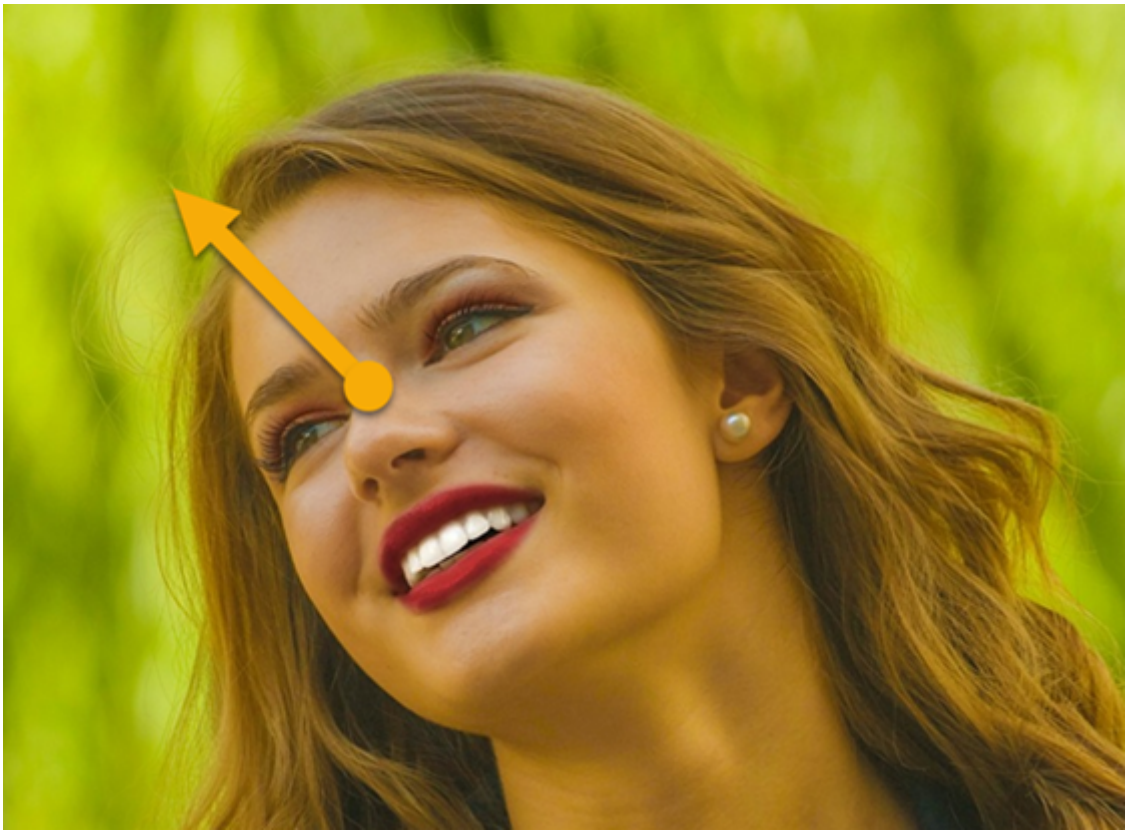


Рис. 16: Направление взгляда: pitch = 31.5, yaw = 31.17

В ресурсах «/iso» и «detect_policy» > «face_quality» также выполняется сравнение оцененного значения с порогом (в соответствии с ISO или нестандартным порогом).

Ресурсы, в которых выполняется оценка:

- «/detector»

Название оценки — «estimate_gaze».

- «/handlers»

  Название оценки — «policies» > «detect_policy» > «estimate_gaze».

- «/verifiers»

  Название оценки — «policies» > «detect_policy» > «estimate_gaze».

- «/sdk»

  Название оценки — «estimate_gaze».

Определение направления взгляда с помощью средств для проверки изображения:

- «/iso» и «/detector» (см. раздел «7.2.3 Expression» пункт «e» в стандарте ISO/IEC 19794-5:2011)

  Названия проверок — «gaze_yaw», «gaze_pitch».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Названия проверок — «gaze_yaw», «gaze_pitch».

Допустимые диапазоны прохождения проверок:

Изображение проходит проверку если попадает в допустимый диапазон соответствующего порога:

Параметр	Допустимые диапазоны
«gaze_yaw» > «threshold»	[-5…5]
«gaze_pitch» > «threshold»	[-5…-5]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.6 Очки

Данная оценка определяет наиболее вероятное состояние очков из следующих состояний:

- «sun_glasses» (солнечные очки);
- «glasses» (обычные очки);
- «no_glasses» (очков нет).

В ресурсах «/iso» и «detect_policy» > «face_quality» также выполняется сравнение оцененного значения с порогом (в соответствии с ISO или нестандартным порогом).

Ресурсы, в которых выполняется оценка:

- «/sdk»

  Название оценки — «estimate_glasses».

Определение состояния очков с помощью средств для проверки изображения:

- «/iso» и «/detector» (см. раздел «7.2.9 Eye glasses» в стандарте ISO/IEC 19794-5:2011)

  Название проверки — «glasses».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Название проверки — «glasses».

Допустимые значения прохождения проверок:

Изображение проходит проверку если попадает в допустимое значение соответствующего порога:

Параметр	Допустимые значения
«glasses» > «threshold»	[«no_glasses», «eyeglasses»]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.7 Брови

Данная оценка определяет наиболее вероятное состояние бровей из следующих состояний:

- «neutral» — брови находятся в обычном положении;
- «raised» — брови подняты;
- «squinting» — глаза прищурены, брови опущены;
- «frowning» — брови нахмурены.

Также выполняется сравнение оцененного значения с порогом (в соответствии с ISO или нестандартным порогом).

Доступна возможность указывать несколько состояний бровей в качестве допустимых.

Рис. 17: Слева направо — «neutral», «raised», «squinting», «frowning»

Требования к изображению:

Для корректных результатов проверки должны быть выполнены нижеописанные требования.

В таблице ниже приведены требования к положению головы:

Параметр	Требуемый диапазон
«pitch»	[-20…20]
«roll»	[-20…20]
«yaw»	[-20…20]

В таблице ниже приведено требование к ширине лица:

Параметр	Требуемый диапазон
«face_width»	> 80

Ресурсы, в которых выполняется оценка:

Определение состояния бровей доступно только с помощью средств для проверки изображения:

- «/iso» и «/detector» (см. раздел «7.2.3 Expression», пункты «d», «f» и «g» в стандарте ISO/IEC 19794-5:2011)

  Название проверки — «eyebrows_state».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Название проверки — «eyebrows_state».

Допустимое значение прохождения проверки:

Изображение проходит проверку если попадает в допустимое значение соответствующего порога:

Параметр	Допустимое значение
«eyebrows_state» > «threshold»	[«neutral»]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.8 Атрибуты рта

Данная оценка определяет вероятностную оценку для каждого нижеперечисленного параметра в диапазоне [0..1]:

- «opened» (рот открыт);
- «smile» (улыбка);
- «occluded» (рот чем-то перекрыт).

Кроме того, определяется степень достоверности обнаружения рта.

В ресурсах «/iso» и «detect_policy» > «face_quality» также выполняется сравнение оцененного значения с порогом (в соответствии с ISO или нестандартным порогом).

Ресурсы, в которых выполняется оценка:

- «/detector»

  Название оценки — «estimate_mouth_attributes».

- «/handlers»

  Название оценки — «policies» > «detect_policy» > «estimate_mouth_attributes».

- «/verifiers»

  Название оценки — «policies» > «detect_policy» > «estimate_mouth_attributes».

- «/sdk»

  Название оценки — «estimate_mouth_attributes».

Определение атрибутов рта с помощью средств для проверки изображения:

- «/iso» и «/detector» (см. раздел «7.2.3 Expression» пункты «a», «b» и «c» в стандарте ISO/IEC 19794-5:2011)

  Названия проверок — «mouth_smiling», «mouth_occluded», «mouth_open».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Названия проверок — «mouth_smiling», «mouth_occluded», «mouth_open».

Допустимые диапазоны прохождения проверок:

Изображение проходит проверку если попадает в допустимый диапазон соответствующего порога:

Параметр	Допустимые диапазоны
«mouth_occluded» > «threshold»	[0…0.5]
«mouth_smiling» > «threshold»	[0…0.5]
«mouth_opened» > «threshold»	[0…0.5]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.8.1 Состояние улыбки

Данная оценка определяет наиболее вероятное состояние улыбки из следующих состояний:

- «none» — улыбки не найдено, поэтому дополнительные параметры не определяются
- «smile_lips» — обычная улыбка со сомкнутыми губами
- «smile_teeth» — улыбка с открытыми зубами

Также выполняется сравнение оцененного значения с порогом (в соответствии с ISO или нестандартным порогом).

При необходимости можно указать несколько состояний улыбки в качестве допустимых.

Требования к изображению:

Для корректных результатов проверки должны быть выполнены нижеописанные требования.

В таблице ниже приведены требования к положению головы:

Параметр	Требуемый диапазон
«pitch»	[-20…20]
«roll»	[-10…10]
«yaw»	[-25…25]

В таблице ниже приведено требование к ширине лица:

Параметр	Требуемый диапазон
«face_width»	> 80

Ресурсы, в которых выполняется оценка:

Определение состояния улыбки доступно только с помощью средств для проверки изображения:

- «/iso» и «/detector» (см. раздел «7.2.3 Expression» пункты «a», «b» и «c» в стандарте ISO/IEC 19794-5:2011)

  Название проверки — «smile_properties».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Название проверки — «smile_properties».

Допустимое значение прохождения проверки:

Изображение проходит проверку если попадает в допустимое значение соответствующего порога:

Параметр	Допустимое значение
«smile_properties» > «threshold»	[«none»]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.9 Качество изображения

Данная оценка определяет вероятностную оценку для каждого нижеперечисленного параметра в диапазоне [0..1], где 0 соответствует низкому качеству, а 1 – высокому качеству:

- «dark» — степень того, что фото не затемнено;
- «light» — степень того, что фото не засвечено;
- «blurriness» — степень размытости;
- «illumination» — степень равномерности освещения. Чем меньше разница между светлыми и темными зонами лица, тем выше расчетное значение. Если освещение равномерно распределено по всему лицу, значение близится к «1».
- «specularity» — степень отсутствия бликов. Чем выше оценочное значение, тем меньше бликов и лучше качество изображения. Если оценочное значение низкое, значит на лице яркие блики.

В ресурсах «/iso» и «detect_policy» > «face_quality» также выполняется сравнение оцененного значения с порогом (в соответствии с ISO или нестандартным порогом).

> Качество изображения определяется с помощью специально обученной нейронной сети VisionLabs. При необходимости можно определить освещенность лица на изображении с помощью алгоритма, выполняющего оценку в соответствии со стандартом ICAO (см. раздел "Равномерность освещения по стандарту ICAO).

Эти данные не сохраняются в базе данных и на основе этих данных не выполняется фильтрация изображений.

Примеры представлены на изображениях ниже. Справа показаны изображения хорошего качества.

Рис. 18: Размытое изображение (слева), не размытое изображение (справа)

Рис. 19: Слишком темное изображение (слева), изображение хорошего качества (справа)

Рис. 20: Слишком светлое (слева), изображение хорошего качества (справа)

Рис. 21: Изображение с неравномерным освещением (слева), изображение с равномерным освещением (справа)

Рис. 22: Изображение с зеркальным отражением и содержит блики на лице (слева), изображение хорошего качества (справа)

Наиболее важными параметрами качества изображения для распознавания лиц являются темнота, свет и размытие.

Параметры освещенности и зеркальности позволяют выбирать изображения лучшего визуально-

го качества. Эти два параметра не оказывают значимого влияния на работу алгоритмов распознавания лиц.

Ресурсы, в которых выполняется оценка:

- «/detector»

  Название оценки — «estimate_quality».

- «/handlers»

  Название оценки — «policies» > «detect_policy» > «estimate_quality».

- «/verifiers»

  Название оценки — «policies» > «detect_policy» > «estimate_quality».

- «/sdk»

  Название оценки — «estimate_quality».

Определение качества изображения с помощью средств для проверки изображения:

- «/iso» и «/detector» (см. разделы «7.2.7 Subject and scene lighting», «7.3.2 Contrast and saturation», «7.3.3 Focus and depth of field», «7.2.8 Hot spots and specular reflections», «7.2.12 Lighting artefacts», «7.2.7 Subject and scene lighting» и «7.2.12 Lighting artefacts» в стандарте ISO/IEC 19794-5:2011)

  Названия проверок — «illumination_quality», «specularity_quality», «blurriness_quality», «dark_quality», «light_quality».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Названия проверок — «illumination_quality», «specularity_quality», «blurriness_quality», «dark_quality», «light_quality».

Допустимые диапазоны прохождения проверок:

Изображение проходит проверку если попадает в допустимый диапазон соответствующего порога:

Параметр	Допустимые диапазоны
«illumination_quality» > «threshold»	[0…0.3]
«specularity_quality» > «threshold»	[0…0.3]
«blurriness_quality» > «threshold»	[0.61…1]
«dark_quality» > «threshold»	[0.5…1]
«light_quality» > «threshold»	[0.57…1]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.9.1 Равномерность освещения по стандарту ICAO

Примечание. Для данной оценки невозможно использовать биометрический образец в качестве входного изображения.

Доступна возможность оценки равномерности освещения по требованиям, указанным в стандарте ICAO.

Также выполняется сравнение оцененного значения с порогом.

> В соответствии со стандартом рекомендуется использовать цветные изображения. При использовании черно-белых изображений результаты могут быть неожиданными.

Определение равномерности освещения по стандарту ICAO доступно только с помощью средства для проверки изображения — группы проверок «/handlers» > «detect_policy» > «face_quality» ресурсов «/handlers» и «/verifiers».

Название проверки — «illumination_uniformity».

Допустимое значение прохождения проверки:

Изображение проходит проверку если попадает в допустимое значение соответствующего порога:

Параметр	Допустимое значение
«illumination_uniformity» > «threshold»	[0.5…1]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.10 Фон изображения

4.2.10.1 Яркость фона

Примечание. Для данной оценки невозможно использовать биометрический образец в качестве входного изображения.

Данная оценка определяет степень яркости фона, где:

- [0…0.1] — черный фон
- [0.1…0.3] — темный фон
- [0.3…0.97] — светлый фон

- [0.97…1] — белый фон

Рис. 23: Фон темный, background_lightness = 0.13 (слева), фон светлый background_lightness = 0.94 (справа)

Ресурсы, в которых выполняется оценка:

Определение степени яркости фона доступно только с помощью средств для проверки изображения:

- «/iso» и «/detector» (см. раздел «B.2.9 Backgrounds» в стандарте ISO/IEC 19794-5:2011)

  Название проверки — «background_lightness».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Название проверки — «background_lightness».

Допустимый диапазон прохождения проверки:

Изображение проходит проверку если попадает в допустимый диапазон соответствующего порога:

Параметр	Допустимый диапазон
«background_lightness» > «threshold»	[0.2…1]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.10.2 Однородность фона

Примечание. Для данной оценки невозможно использовать биометрический образец в качестве входного изображения.

Данная оценка позволяет определить степень однородности фона, где:

- «0» — фон неоднородный;
- «1» — фон однородный.

Рис. 24: Фон неоднородный, background_uniformity = 0.004 (слева), фон однородный, background_uniformity = 0.7 (справа)

Ресурсы, в которых выполняется оценка:

Определение однородности фона доступно только с помощью средств для проверки изображения:

- «/iso» и «/detector» (см. раздел «B.2.9 Backgrounds» в стандарте ISO/IEC 19794-5:2011)

  Название проверки — «background_uniformity».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Название проверки — «background_uniformity».

Допустимый диапазон прохождения проверки:

Изображение проходит проверку если попадает в допустимый диапазон соответствующего порога:

Параметр	Допустимый диапазон
«background_uniformity» > «threshold»	[0.5…1]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.11 Динамический диапазон по стандарту ICAO

Примечание. Для данной оценки невозможно использовать биометрический образец в качестве входного изображения.

Данная оценка представляет собой определение отношения яркости самых светлых и самых тёмных участков лица по требованиям, указанным в стандарте ICAO.

Также выполняется сравнение оцененного значения с порогом.

Определение динамического диапазона изображения доступно только с помощью средств для проверки изображения:

- «/detector»

  Название проверки — «dynamic_range».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Название проверки — «dynamic_range».

Допустимый диапазон прохождения проверки:

Изображение проходит проверку если попадает в допустимый диапазон соответствующего порога:

Параметр	Допустимый диапазон
«dynamic_range» > «threshold»	[0.5…1]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.12 Естественность освещения

Данная оценка определяет естественное ли освещение на лице, где:

- «0» — освещение неестественное;
- «1» — освещение естественное.

Рис. 25: Освещение неестественное (слева), освещение естественное (справа)

Также выполняется сравнение оцененного значения с порогом (в соответствии с ISO или нестандартным порогом).

Требования к изображению:

Для корректных результатов проверки должны быть выполнены нижеописанные требования.

В таблице ниже приведено требование к маске:

Параметр	Требуемое значение
«predominant_mask»	«missing»

В таблице ниже приведено требование к параметру качества изображения:

Параметр	Требуемый диапазон
«blurriness»	[0.5…1]

В таблице ниже приведено требование к очкам:

Параметр	Требуемые значения
«glasses»	«no_glasses» или «eyeglasses»

Ресурсы, в которых выполняется оценка:

Определение естественности освещения доступно только с помощью средств для проверки изображения:

- «/iso» и «/detector» (см. раздел «7.3.4 Unnatural colour» в стандарте ISO/IEC 19794-5:2011)

  Название проверки — «natural_light».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Название проверки — «natural_light».

Допустимое значение прохождения проверки:

Изображение проходит проверку если попадает в допустимое значение соответствующего порога:

Параметр	Допустимое значение
«natural_light» > «threshold»	«1»

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.13 Тип цвета изображения на основе лица

Данная оценка определяет наиболее вероятный тип цвета, определяемого по лицу, из следующих:

- «color» — изображение цветное;
- «grayscale» — изображение черно-белое;
- «infrared» — изображение находится в ближнем инфракрасном диапазоне (near infrared).

Также выполняется сравнение оцененного значения с порогом (в соответствии с ISO или нестандартным порогом).

Ресурсы, в которых выполняется оценка:

Определение типа света доступно только с помощью средств для проверки изображения:

- «/iso» и «/detector» (см. раздел «7.4.4 Use of near infra-red cameras» в стандарте ISO/IEC 19794-5:2011)

Название проверки — «face_color_type».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Название проверки — «face_color_type».

Допустимое значение прохождения проверки:

Изображение проходит проверку если попадает в допустимое значение соответствующего порога:

Параметр	Допустимое значение
«face_color_type» > «threshold»	[«color»]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.14 Положение головы

Данная оценка определяет положение головы. Положение головы определяется тремя параметрами:

- угол наклона головы вверх/вниз (pitch);
- угол отклонения головы вправо/влево (roll);
- угол поворота головы вправо/влево (yaw).

Значения углов определяются в диапазоне от «-180» до «180».

Положительное значение угла pitch означает направление наклона головы вверх, а отрицательное значение — направление наклона головы вниз.

Положительное значение угла roll означает направление отклонения головы вправо, а отрицательное значение — направление отклонения головы влево.

Положительное значение угла yaw означает направление поворота головы вправо, а отрицательное значение — направление поворота головы влево.

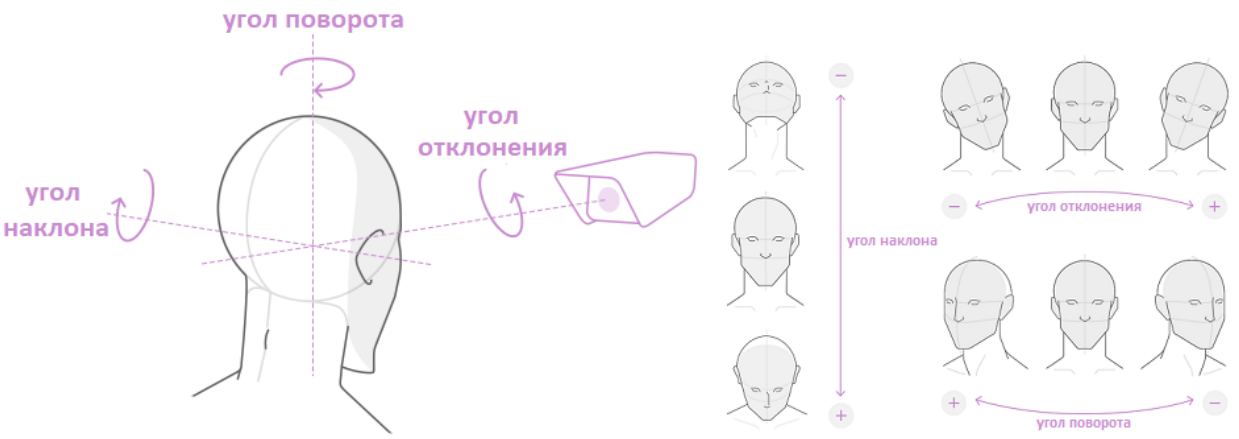


Рис. 26: Положение головы

В ресурсах «/iso» и «detect_policy» > «face_quality» также выполняется сравнение оцененного значения с порогом (в соответствии с ISO или нестандартным порогом).

Во всех перечисленных ниже ресурсах, за исключением «/iso», доступна возможность отфильтровать изображения по положению головы.

В ресурсах «/detector», «/handlers», «/verifiers» и «/sdk» для порога задается значение от «0» до «180». Значение по умолчанию равно «180», что означает, что голова на изображении может быть повёрнута на любой угол от «-180» до «180». При установке любого другого значения (например, «30») все детекции с углом головы, который меньше или равен «-30» и больше или равен «30» будут отфильтрованы.

Для поля «face_quality» ресурсов «/handlers» и «/verifiers» задается минимальный и максимальный порог в отдельных полях.

Ресурсы, в которых выполняется оценка:

- «/detector»

  Название оценки — «estimate_head_pose».

- «/handlers»

  Название оценки — «policies» > «detect_policy» > «estimate_head_pose».

- «/verifiers»

  Название оценки — «policies» > «detect_policy» > «estimate_head_pose».

- «/sdk»

  Название оценки — «estimate_head_pose».

Ниже представлены рекомендуемые пороги для проведения оценки в ресурсах «/detector», «/handlers», «/verifiers» and «/sdk».

Рекомендуемые максимальные пороги:

В таблице ниже приведены рекомендуемые максимальные пороги положения головы для проведения оценки **в кооперативном режиме**:

Параметр	Рекомендуемые максимальные пороги
«roll_threshold»	30
«pitch_threshold»	15
«yaw_threshold»	15

В таблице ниже приведены рекомендуемые максимальные пороги положения головы для проведения оценки **в некооперативном режиме**:

Параметр	Рекомендуемые максимальные пороги
«roll_threshold»	30
«pitch_threshold»	30
«yaw_threshold»	30

Определение положения головы с помощью средств для проверки изображения:

- «/iso» и «/detector» (см. раздел «7.2.2 Pose» в стандарте ISO/IEC 19794-5:2011)

  Названия проверок — «head_roll», «head_pitch», «head_yaw».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Названия проверок — «head_roll», «head_pitch», «head_yaw».

Допустимые диапазоны прохождения проверок:

Изображение проходит проверку если попадает в допустимый диапазон соответствующего порога:

Параметр	Допустимые диапазоны
«head_yaw» > «threshold»	[-5…5]
«head_pitch» > «threshold»	[-5…-5]
«head_roll» > «threshold»	[-8…-8]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.15 Положение лица по вертикали и горизонтали

Примечание. Для данных оценок невозможно использовать биометрический образец в качестве входного изображения.

Данные оценки определяют положение центральной точки по вертикали и горизонтали относительно изображения.

Также выполняется сравнение оцененных значений с порогами (в соответствии с ISO или нестандартными порогами).

Ресурсы, в которых выполняется оценка:

Определение положения лица по вертикали и горизонтали доступно только с помощью средств для проверки изображения:

- «/iso» и «/detector» (см. разделы «8.3.2 Horizontally centred face» и «8.3.3 Vertical position of the face» в стандарте ISO/IEC 19794-5:2011)

  Названия проверок — «head_horizontal_center», «head_vertical_center».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Названия проверок — «head_horizontal_center», «head_vertical_center».

Допустимые диапазоны прохождения проверок:

Изображение проходит проверку если попадает в допустимый диапазон соответствующего порога:

Параметр	Допустимые диапазоны
«head_horizontal_center» > «threshold»	[0.45…0.55]
«head_vertical_center» > «threshold»	[0.3…0.5]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.16 Ширина и высота головы (вертикальный и горизонтальный размеры)

Примечание. Для данных оценок невозможно использовать биометрический образец в качестве входного изображения.

Данные оценки определяют вертикальный и горизонтальный размер головы относительно размера изображения. Также выполняется сравнение оцененных значений с порогами (в соответствии с ISO или нестандартными порогами).

Ресурсы, в которых выполняется оценка:

Определение размеров головы по горизонтали и вертикали доступно только с помощью средств для проверки изображения:

- «/iso» и «/detector» (см. разделы «8.3.4 Width of head» и «8.3.5 Length of head» в стандарте ISO/IEC 19794-5:2011)

  Названия проверок — «head_width», «head_height».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Названия проверок — «head_width», «head_height».

Допустимые диапазоны прохождения проверок:

Изображение проходит проверку если попадает в допустимый диапазон соответствующего порога:

Параметр	Допустимые диапазоны
«head_width» > «threshold»	[0.5…0.75]
«head_height» > «threshold»	[0.6…0.9]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.17 Ширина и высота лица

Примечание. Для данных оценок невозможно использовать биометрический образец в качестве входного изображения.

Данные оценки определяют ширину и высоту лица в пикселях. Также выполняется сравнение оцененных значений с заданными порогами.

Ресурсы, в которых выполняется оценка:

Определение ширины или высоты лица на изображении доступно только с помощью средств для проверки изображения:

- «/detector»

  Названия проверок — «face_width», «face_height».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Названия проверок — «face_width», «face_height».

Допустимые диапазоны прохождения проверок:

Изображение проходит проверку если попадает в допустимый диапазон соответствующего порога:

Параметр	Допустимые диапазоны
«face_width» > «threshold»	[180…1920]
«face_height» > «threshold»	[180…inf]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.18 Отступы от краёв изображения

Примечание. Для данных оценок невозможно использовать биометрический образец в качестве входного изображения.

Оценки отступов от краёв изображения определяются как отступы от границы изображения (левой, правой, верхней, нижней) до границы лица (левой, правой, верхней, нижней) в пикселях. Также выполняется сравнение оцененных значений с заданными порогами.

Ресурсы, в которых выполняется оценка:

Определение отступов от краёв изображения доступно только с помощью средств для проверки изображения:

- «/detector»

  Названия проверок — «indent_upper», «indent_lower», «indent_right», «indent_left».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Названия проверок — «indent_upper», «indent_lower», «indent_right», «indent_left».

Допустимые диапазоны прохождения проверок:

Изображение проходит проверку если попадает в допустимый диапазон соответствующего порога:

Параметр	Допустимые диапазоны
«indent_upper» > «threshold»	[20…inf]

Параметр	Допустимые диапазоны
«indent_lower» > «threshold»	[20…inf]
«indent_right» > «threshold»	[20…inf]
«indent_left» > «threshold»	[20…inf]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.19 Маска

Данная оценка определяет вероятностную оценку для каждого нижеперечисленного параметра в диапазоне [0..1]:

- «medical_mask» (надета медицинская маска);
- «missing» (не надета медицинская маска);
- «occluded» (лицо закрыто другим предметом, помимо медицинской маски).

Также определяется наиболее вероятное состояние маски.

Помимо трёх основных состояний определяются следующие дополнительные состояния:

- «correct» — маска на лице, рот и нос закрыты маской
- «mouth» — маска закрывает только рот
- «clear» — на лице нет маски
- «chin» — маска находится под подбородком и не перекрывает зону от глаз до рта
- «partially» — лицо частично перекрыто, но не медицинской маской и не маской с полным перекрытием лица
- «full» — на лице присутствует маска, при которой полностью закрыто лицо, например, балаклава/лыжная маска

Каждому основному состоянию маски соответствует одно из двух дополнительных свойства. Наиболее вероятное дополнительное свойство возвращается в поле «predominant_occlusion»:

- состоянию «medical_mask» соответствует свойство «correct» или «mouth»
- состоянию «missing» соответствует свойство «clear» или «chin»
- состоянию «occluded» соответствует свойство «partially» или «full»

Для каждого из свойств возвращается вероятностная оценка в диапазоне [0..1].

Дополнительные свойства маски не записываются в БД и по ним не выполняется фильтрация.

Рис. 27: Слева направо — «correct», «clear», «partially»

Ресурсы, в которых выполняется оценка:

- «/detector»

  Название оценки — «estimate_mask».

- «/handlers»

  Название оценки — «policies» > «detect_policy» > «estimate_mask».

- «/verifiers»

  Название оценки — «policies» > «detect_policy» > «estimate_mask».

- «/sdk»

  Название оценки — «estimate_mask».

4.2.20 Эмоции

Данная оценка определяет вероятностную оценку для каждого нижеперечисленного параметра в диапазоне [0..1]:

- «anger» (злость);
- «disgust» (отвращение);
- «fear» (страх);
- «happiness» (счастье);
- «neutral» (нейтральность);
- «sadness» (грусть);
- «surprise» (удивление).

Также определяется наиболее вероятная эмоция.

Эмоции можно сохранить в объекте события при создании события.

Ресурсы, в которых выполняется оценка:

- «/detector»

  Название оценки — «estimate_emotions».

- «/handlers»

  Название оценки — «policies» > «detect_policy» > «estimate_emotions».

- «/verifiers»

  Название оценки — «policies» > «detect_policy» > «estimate_emotions».

- «/sdk»

  Название оценки — «estimate_emotions».

4.2.21 Положение плеч

Данная оценка определяет наиболее вероятное состояние положения плеч из следующих состояний:

- «non-parallel» (плечи не параллельны)
- «parallel» (плечи параллельны)
- «hidden» (плечи скрыты)

Ресурсы, в которых выполняется оценка:

Определение головного убора доступно только с помощью средств для проверки изображения:

- «/iso» и «/detector» (см. раздел «7.2.5 Shoulders» в стандарте ISO/IEC 19794-5:2011)

  Название проверки — «shoulders_position».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Название проверки — «shoulders_position».

Допустимое значение прохождения проверки:

Изображение проходит проверку если попадает в допустимое значение соответствующего порога:

Параметр	Допустимый диапазон
«shoulders_position» > «threshold»	[«parallel»]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.22 Головной убор

Данная оценка определяет наиболее вероятный тип головного убора из следующих:

- «none» (нет головного убора)
- «baseball_cap» (кепка/бейсболка);
- «beanie» (шапка);
- «peaked_cap» (фуражка);
- «shawl» (платок);
- «hat_with_ear_flaps» (ушанка);
- «helmet» (шлем/каска);
- «hood» (капюшон);
- «hat» (шляпа);
- «other» (прочее).

Также выполняется сравнение оцененного значения с порогом (в соответствии с ISO или нестандартным порогом).

Доступна возможность указывать несколько типов головного убора в качестве допустимых.

Требования к изображению:

Для корректных результатов проверки должны быть выполнены нижеперечисленные требования.

В таблице ниже приведены требования к положению головы:

Параметр	Требуемый диапазон
«pitch»	[-20…20]
«roll»	[-10…10]
«yaw»	[-25…25]

В таблице ниже приведено требование к ширине лица:

Параметр	Требуемый диапазон
face_width	> 80

Ресурсы, в которых выполняется оценка:

Определение головного убора доступно только с помощью средств для проверки изображения:

- «/iso» и «/detector» (см. раздел «B.2.7 Head coverings» в стандарте ISO/IEC 19794-5:2011)

  Название проверки — «headwear_type».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Название проверки — «headwear_type».

Допустимое значение прохождения проверки:

Изображение проходит проверку если попадает в допустимое значение соответствующего порога:

Параметр	Допустимое значение
«headwear_type» > «threshold»	[«none»]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.23 Бочкообразная дисторсия (эффект «Fisheye»)

Данная оценка определяет наличие эффекта «Fisheye», где:

- «0» — на изображении не присутствует эффект «Fisheye»;
- «1» — на изображении присутствует эффект «Fisheye».

Также выполняется сравнение оцененного значения с порогом (в соответствии с ISO или нестандартным порогом).

Требования к изображению:

Для корректных результатов проверки должны быть выполнены нижеперечисленные требования.

В таблице ниже приведены требования к положению головы:

Параметр	Требуемый диапазон
«pitch»	[-20…20]
«roll»	[-10…10]
«yaw»	[-25…25]

В таблице ниже приведено требование к ширине лица:

Параметр	Требуемый диапазон
«face_width»	> 80

Ресурсы, в которых выполняется оценка:

Определение эффекта «Fisheye» доступна только с помощью средств для проверки изображения:

- «/iso» и «/detector» (см. раздел «7.3.6 Radial distortion of the camera lens» в стандарте ISO/IEC 19794-5:2011)

  Название проверки — «radial_distortion».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Название проверки — «radial_distortion».

Допустимое значение прохождения проверки:

Изображение проходит проверку если попадает в допустимое значение соответствующего порога:

Параметр	Допустимое значение
«radial_distortion» > «threshold»	«1»

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.2.24 Перекрытие лица

Эстиматор позволяет определить наличие перекрытия всего лица или его отдельных зон.

Эстиматор возвращает информацию о перекрытии:

- Всего лица
- Зоны лба
- Зоны глаз
- Зоны носа
- Зоны рта
- Нижней части лица

Пороги перекрытия

Возможно задать допустимый процент перекрытия всего лица или каждой отдельной части лица на уровне всей системы в Configurator. Кроме того, пороги можно переопределить при создании обработчика.

Возможно задать порог для перекрытия волосами. Порог указывает допустимый процент перекрытия волосами, при котором волосы не учитываются в качестве перекрытия. Всё перекрытие, которое превышает указанный порог, учитывается в общем перекрытии лица и перекрытии каждой из зон лица.

- Если порог перекрытия волосами задан равным 0, то волосы любой длины будут считаться перекрытием.
- Если порог перекрытия волосами задан равным 1, то перекрытие волосами не учитывается.

Пороги могут быть изменены в соответствии с бизнес задачами.

Примечание. Зона усов и бороды никогда не считаются перекрытием лица.

Фильтрация

Доступна фильтрация. Имеется возможность указать список зон, которые не должны быть перекрыты.

Детекция фильтруется, если перекрытие любой из указанных зон превышает порог. Дальнейшая обработка детекции не выполняется и в ответе возвращается причина фильтрации.

Ресурсы, в которых выполняется оценка:

- «/handlers»

  Название оценки — «estimate_face_occlusion».

  Проверки в рамках `face_quality`:

  - `face_occlusion` - перекрытие зоны лица;
  - `lower_face_occlusion` - перекрытие нижней части лица;
  - `forehead_occlusion` - перекрытие зоны лба;
  - `nose_occlusion` - перекрытие зоны носа.

- «/verifiers»

  Название оценки — «estimate_face_occlusion».

  Проверки в рамках `face_quality`.

- «/sdk»

  Название оценки — «estimate_face_occlusion».

4.3 Параметры изображений

4.3.1 Формат изображения

Данная оценка определяет формат входящего изображения («JPEG», «JPEG2000» или «PNG»). Также выполняется сравнение оцененного значения с порогом (в соответствии с ISO или нестандартным порогом).

Ресурсы, в которых выполняется оценка:

Определение формата изображения доступно только с помощью средств для проверки изображения:

- «/iso» и «/detector» (см. раздел «7.5 Format requirements for the Frontal Image Type» в стандарте ISO/IEC 19794-5:2011)

  Название проверки — «image_format».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Название проверки — «image_format».

Допустимые значения прохождения проверки:

Изображение проходит проверку если попадает в допустимое значение соответствующего порога:

Параметр	Допустимые значения
«image_format» > «threshold»	[«JPEG», «JPEG2000», «PNG»]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.3.2 Размер изображения

Данная оценка определяет размер изображения в байтах. Также выполняется сравнение оцененного значения с заданным порогом.

Ресурсы, в которых выполняется оценка:

Определение размера изображения доступно только с помощью средств для проверки изображения:

- «/detector»

  Название проверки — «image_size».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

Название проверки — «image_size».

Допустимый диапазон прохождения проверки:

Изображение проходит проверку если попадает в допустимый диапазон соответствующего порога:

Параметр	Допустимый диапазон
«image_size» > «threshold»	[5120…2097152]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.3.3 Ширина и высота изображения

Данные оценки определяют ширину и высоту изображения в пикселях. Также выполняется сравнение оцененных значений с порогами (в соответствии с ISO или нестандартными порогами).

Ресурсы, в которых выполняется оценка:

Определение ширины и высоты изображения доступно только с помощью средств для проверки изображения:

- «/iso» и «/detector» (см. разделы «5.7.4 Width» и «5.7.5 Height» в стандарте ISO/IEC 19794-5:2011)

  Названия проверок — «image_height», «image_width».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Названия проверок — «image_height», «image_width».

Допустимые диапазоны прохождения проверок:

Изображение проходит проверку если попадает в допустимый диапазон соответствующего порога:

Параметр	Допустимые диапазоны
«image_height» > «threshold»	[180…1920]
«image_width» > «threshold»	[180…1080]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.3.4 Соотношение сторон изображения

Данная оценка определяет пропорциональное отношение ширины изображения к высоте. Также выполняется сравнение оцененного значения с заданным порогом.

Ресурсы, в которых выполняется оценка:

Определение соотношения сторон изображения доступно только с помощью средств для проверки изображения:

- «/detector»

  Название проверки — «aspect_ratio».

- «detect_policy» > «face_quality» в ресурсах «/handlers» и «/verifiers»

  Название проверки — «aspect_ratio».

Допустимый диапазон прохождения проверки:

Изображение проходит проверку если попадает в допустимый диапазон соответствующего порога:

Параметр	Допустимый диапазон
«aspect_ratio» > «threshold»	[0.74…0.8]

> Для средства проверки изображения «face_quality» допустимое значение может быть задано вручную, однако это будет означать отклонение от стандарта.

4.3.5 Метаданные EXIF

При включённой оценке EXIF все теги изображения анализируются, после чего выводятся их названия и значения. См. спецификацию JEITA CP-3451 EXIF для получения подробной информации. Возвращаются следующие данные:

- make;
- model;
- orientation;
- latitude;
- longitude;
- artist;
- software;
- dateTime;
- digitalZoomRatio;
- flash;

- uid.

Ресурсы, в которых выполняется оценка:

- «/detector»

  Название оценки — «extract_exif».

- «/handlers»

  Название оценки — «policies» > «detect_policy» > «extract_exif».

- «/verifiers»

  Название оценки — «policies» > «detect_policy» > «extract_exif».

- «/sdk»

  Название оценки — «use_exif_info».

4.4 Параметры тел

4.4.1 Пол и возраст по изображению тела

Данная оценка позволяет определить пол и возраст человека по изображению тела.

> Определение пола и возраста по изображению тела является менее точным, чем по лицу.

Ресурсы, в которых выполняется оценка:

- «/handlers»

  Название оценки — «policies» > «detect_policy» > «body_attributes» > «estimate_basic_-attributes».

- «/sdk»

  Название оценки — «estimate_body_basic_attributes».

4.4.2 Верхняя часть тела

Данная оценка определяет параметры следующих элементов одежды на верхней части тела:

- «headwear» — головной убор (нет, есть, неизвестно), цвет головного убора (белый, черный, прочий, неизвестно);
- «sleeve» — рукава (длинные рукава, короткие рукава, неизвестно);
- «upper_clothing» — цвет верхней одежды (бежевый, чёрный, синий, коричневый, зеленый, серый, хаки, разноцветный, оранжевый, розовый, фиолетовый, красный, белый, желтый, неизвестно)

Ресурсы, в которых выполняется оценка:

- «/handlers»

  Название оценки — «policies» > «detect_policy» > «body_attributes» > «estimate_upper_body».

- «/sdk»

  Название оценки — «estimate_upper_body».

4.4.3 Нижняя часть тела

Данная оценка определяет параметры следующих элементов одежды на нижней части тела:

- «lower_garment» — нижняя одежда (брюки, шорты, юбка, неизвестно), цвет нижней одежды (бежевый, чёрный, синий, коричневый, зеленый, серый, хаки, разноцветный, оранжевый, розовый, фиолетовый, красный, белый, желтый, неизвестно);
- «shoes» — цвет обуви (черный, белый, прочий, неизвестный);

Ресурсы, в которых выполняется оценка:

- «/handlers»

  Название оценки — «policies» > «detect_policy» > «body_attributes» > «estimate_lower_body».

- «/sdk»

  Название оценки — «estimate_lower_body».

4.4.4 Наличие рюкзака

Данная оценка определяет наличие рюкзака на теле:

- «0» — на изображении тела нет рюкзака;
- «1» — на изображении тела есть рюкзак.

Ресурсы, в которых выполняется оценка:

- «/handlers»

  Название оценки — «policies» > «detect_policy» > «body_attributes» > «estimate_accessories».

- «/sdk»

  Название оценки — «estimate_accessories».

4.5 Liveness

Примечание. Возможность выполнения такой оценки регулируется особым параметром в лицензионном ключе LUNA PLATFORM 5.

Технология Liveness позволяет обнаруживать атаки на биометрическое предъявление. Для оценки Liveness в LUNA PLATFORM, используется эстиматор LUNA SDK OneShotLiveness.

В результате оценки Liveness может вернуться один из следующих результатов:

- «0» — человек не является реальным;
- «1» — человек является реальным;
- «2» — результат проверки неизвестен.

Ресурсы, в которых выполняется оценка:

- «/handlers»

  Название оценки — «policies» > «detect_policy» > «estimate_liveness».

- «/verifiers»

  Название оценки — «policies» > «detect_policy» > «estimate_liveness».

- «/sdk»

  Название оценки — «estimate_liveness».

- «/liveness»

См. дополнительную информацию про Liveness в разделе «Описание OneShotLiveness».

4.6 Deepfake

Примечание. Возможность выполнения такой оценки регулируется особым параметром в лицензионном ключе LUNA PLATFORM 5.

Примечание. Для данной оценки невозможно использовать биометрический образец в качестве входного изображения.

Данная оценка позволяет обнаруживать подмену лиц с помощью технологии DeepFake на фотоизображениях.

В результате оценки Deepfake могут вернуться следующие результаты:

- «prediction» = «fake» — человек не является реальным;
- «prediction» = «real» — человек является реальным;
- «score» = [0…1] — степень достоверности выполнения оценки.

При необходимости можно настроить обработчик так, чтобы фильтровать события по предполагаемому результату оценки Deepfake («fake» или «real»). Для этого в теле запроса обработчика надо указать параметр «deepfake_states» со значением «0» (фильтровать по значению «fake») или «1» (фильтровать по значению «real»). Например, если параметр «deepfake_states» равен «1» (фильтровать по «real»), а эстиматор определил, что результат «fake», то в событии вернется пустое поле «events», а результаты проверки попадут в поле «filtered_detections».

В запросах «create handler» и «create verifier» доступна возможность задать порог «real_threshold» и режим работы «mode». В запросе «sdk» будут использованы значения данных параметров по умолчанию (см. ниже) без возможности явного указания.

Порог

С помощью порога «real_threshold» можно задать значение в диапазоне [0…1], ниже которого система будет считать, что человек не является реальным.

Например, если значение порога «real_threshold» = «0.5», а степень достоверности выполнения оценки «score» = «0.4», то в теле ответа будет выдан результат «prediction» = «fake». Если же значение порога «real_threshold» = «0.6», а степень достоверности выполнения оценки «score» = «0.7», то в теле ответа будет выдан результат «prediction» = «real».

Значение по умолчанию — «0.5».

Режимы работы

Режимы работы описаны в таблице ниже.

Режим работы	Описание
«mode» = «1»	Упрощенный режим работы.
«mode» = «2» (по умолчанию)	Режим работы с использованием дополнительной модели нейронной сети. При использовании данного режима будет дополнительно выполняться предварительная оценка лица на исходном изображении. Если результат предварительной проверки определил, что лицо является поддельным, то в теле ответа будет возвращен результат «score» = «0» и «prediction» = «fake».

Требования к изображению

Для корректных результатов проверки должны быть выполнены нижеописанные требования.

В таблице ниже приведены требования к положению головы:

Параметр	Требуемый диапазон
«pitch»	[-20…20]
«yaw»	[-30…30]

В таблице ниже приведено требование к ширине лица:

Параметр	Требуемый диапазон
«face_width»	> 150

Ресурсы, в которых выполняется оценка:

- «/handlers»

  Название оценки — «policies» > «detect_policy» > «estimate_deepfake».

- «/verifiers»

  Название оценки — «policies» > «detect_policy» > «estimate_deepfake».

- «/sdk»

  Название оценки — «estimate_deepfake».

4.7 Количество людей

Примечание. Возможность выполнения такой оценки регулируется особым параметром в лицензионном ключе LUNA PLATFORM 5.

Данная оценка определяет количество людей на изображении.

Рис. 28: Оценка количества людей

Ресурсы, в которых выполняется оценка:

- «/handlers»

  Название оценки — «policies» > «detect_policy» > «estimate_people_count».

- «/sdk»

  Название оценки — «estimate_people_count».

При необходимости совместно с оценкой количества людей можно получить X и Y координаты людей с помощью параметра «people_count_coordinates» в ресурсе «/handlers» и параметра «people_-coordinates» в ресурсе «/sdk».

5 Взаимодействие сервисов LP

На схеме ниже для каждого сервиса указаны отдельные базы данных. Они приведены для наглядности. Не нужно запускать отдельный экземпляр БД для каждого сервиса. Можно запустить один экземпляр (например, PostgreSQL) и использовать его для хранения данных каждого из сервисов LP. У каждого сервиса будет своя таблица в этой базе данных.

См. раздел «Общие сведения» для подробной информации о базах данных, используемых сервисами LP.

Сервис API обеспечивает RESTful интерфейс для выполнения детекции лиц, извлечения и сравнения биометрических шаблонов.

Запросы отправляются в LP с помощью протокола HTTP. Стандартный механизм работы: внешний сервис отправляет запросы в LP, получает результаты и обрабатывает их в соответствии с заданными в запросе параметрами.

> Всю информацию об основных запросах в API можно найти в спецификации OpenAPI.

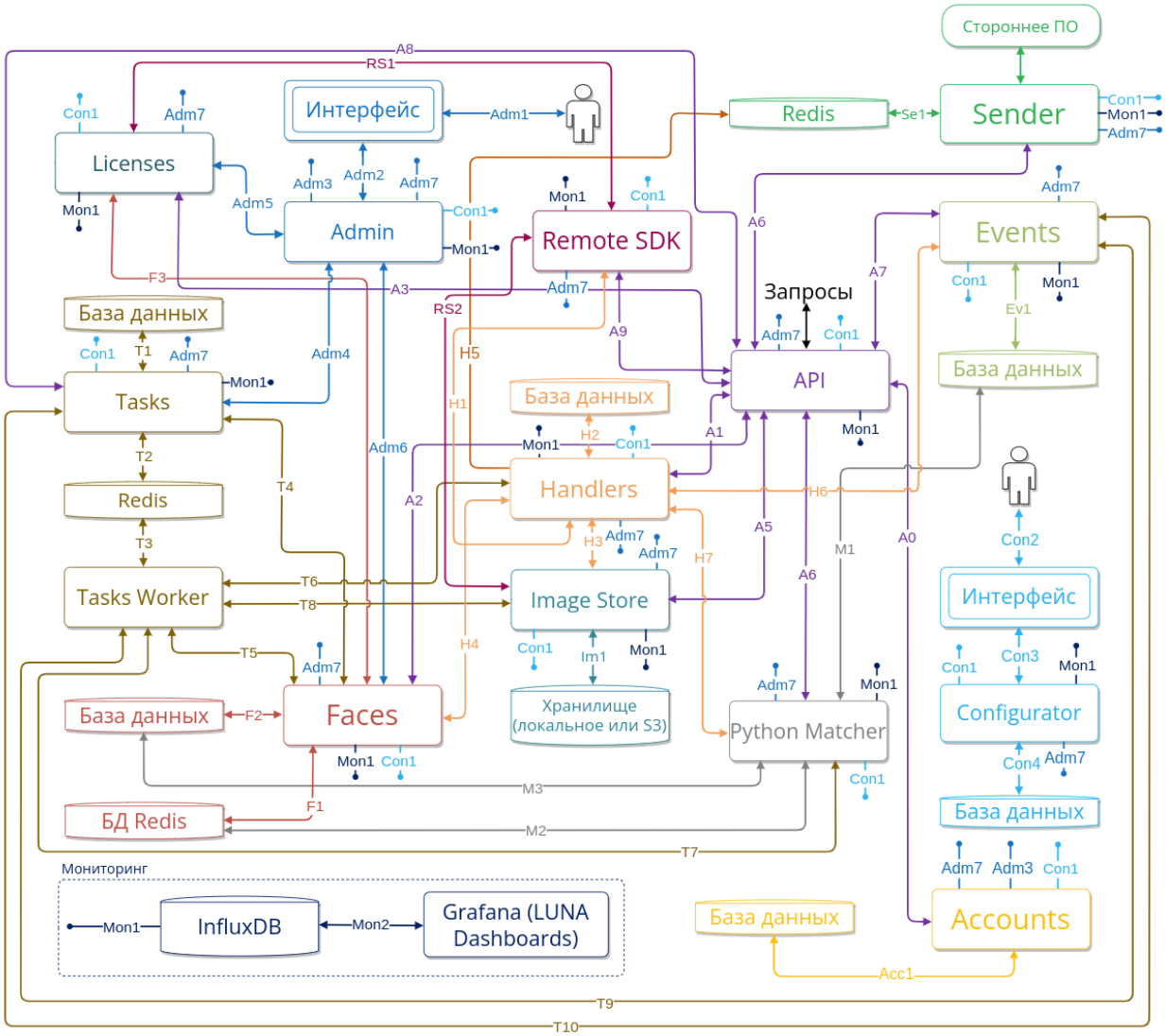


Рис. 29: Взаимодействие сервисов

> Некоторые сервисы можно отключить (см. раздел «Общие сведения»).

API получает запросы, обрабатывает их и проверяет авторизацию пользователя с помощью запроса к сервису **Accounts (A0)**. Сервис Accounts проверяет наличие аутентификационных данных пользователя в базе данных Accounts **(Acc1)**. Если данные пользователя не найдены в базе данных, то выдается ошибка авторизации. Если данные найдены, то соответствующая информация отправляется в сервис API, где тот передает исходные запросы в другие сервисы.

Основные операции (детекция, эстимация, извлечение, сравнение) выполняются нижеописанным образом.

Подход «Последовательное выполнение запросов»:

- запрос «detect faces», в котором выполняется обнаружение лица, оценка параметров лица

и создание биометрических образцов, отправляется из сервиса API в сервис Handlers **(A1)**, а затем перенаправляются в сервис Remote SDK **(H1)**;
- запрос «extract attributes», в котором выполняется извлечение временных атрибутов, отправляется из сервиса API в сервис Handlers **(A1)**, а затем перенаправляется в сервис Remote SDK **(H1)**;
- запрос «matching faces» отправляется из сервиса API в сервис Python Matcher **(A6)**.

Подход «Параллельное выполнение запросов»:

- запрос «create handler» с правилами детекции, эстимации, извлечения и матчинга лиц и тел отправляется из сервиса API в сервис Handlers **(A1)**;
- запрос «generate events» отправляется из сервиса API в сервис Handlers **(A1)**, а затем перенаправляется в сервис Remote SDK **(H1)**

Сервис API отправляет запросы в сервис Faces для создания/изменения новых лиц (запросы из секции «faces»), а также создания/изменения списков (запросы из секции «lists») **(A2)**.

Сервис API получает информацию о текущих лицензионных условиях из сервиса Licenses **(A3)**.

Сервис API отправляет биометрические образцы от внешних сервисов в сервис Image Store (запрос «save face/body sample») **(A5)**.

Handlers создает новые обработчики и хранит их в базе данных Handlers **(H2)**.

Также сервис перенаправляет запросы на детекцию и эстимацию в сервис Remote SDK **(H1)**.

> Сервис Handlers включается/отключается в настройке «ADDITIONAL_SERVICE_USAGE».

Remote SDK обрабатывает запросы на обнаружение лица/тела и создание атрибутов, оценивает параметры лиц/тел.

Сервис Remote SDK обрабатывает запрос на обнаружение лиц/тел и оценку параметров из сервиса Handlers **(H1)**, отправляет результат обратно в Handlers **(H1)**, который отправляет полученные биометрические образцы в сервис Image Store **(H3)**.

Сервис Remote SDK обрабатывает запрос на создание атрибутов из сервиса Handlers **(H1)**, запрашивая у сервиса Handlers существующие биометрические образцы, который тот запрашивает у сервиса Image Store **(H3)**. Далее сервис Handlers отправляет созданные временные атрибуты в сервис Faces **(H4)**, который хранит их в базе данных Redis **(F1)**.

Запросы «/iso», «/sdk», «/liveness» выполняются напрямую к сервису Remote SDK **(A9)**. В таком случае выполняется проверка лицензии с помощью взаимодействия с сервисом Licenses **(RS1)**. Для возможности использования набора биометрических образцов, сервис Remote SDK взаимодействует с сервисом Image Store **(RS2)**.

Image Store получает биометрические образцы из сервиса Remote SDK и хранит их на локальном накопителе или на S3-подобном облачном хранилище и предоставляет доступ к ним **(Im1)**.

> Сервис Image Store включается/отключается в настройке «ADDITIONAL_SERVICE_USAGE».

Faces отвечает за взаимодействие с базой данных Faces, которая хранит: лица, БШ лиц и списки **(F2)**. Он дает доступ к сохраненным данным для сервисов API, Python Matcher и Tasks.

Faces также хранит созданные временные атрибуты в базе данных Redis **(F1)**.

Каждый раз при создании нового лица, сервис Faces отправляет информацию о количестве созданных лиц с прикрепленными атрибутами в сервис Licenses (**F3**) (см. раздел «Максимальное количество лиц»).

Events хранит и предоставляет информацию о событиях. После создания события сервис Events получает созданное событие из сервиса Handlers **(H6)** и хранит его в базе данных Events **(Ev1)**.

> Использование сервиса Events можно включать и отключать в файле конфигурации сервиса API.

Python Matcher используется для сравнения биометрических шаблонов. Запрос на сравнение может быть отправлен напрямую сервисом API **(A6)**. Если в обработчике указана политика сравнения, то запрос на сравнение БШ поступит из сервиса Handlers **(H7)**. Python Matcher отправляет запросы на сравнение в базу данных Faces **(M3)** или Events **(M1)**. Эти базы данных сравнивают биометрические шаблоны и отправляют результаты сравнения.

Python Matcher также может получать временные атрибуты, требуемые для сравнения, из базы данных Redis **(M2)**.

> Доступна возможность дополнительно использовать сервис Python Matcher Proxy, который может перенаправлять запросы либо сервису Python Matcher, либо плагинам сравнения. Плагины сравнения позволяют значительно ускорить выполнение запросов на сравнение (см. раздел «Плагины сравнения»). Данный сервис не отображен на схеме, но имеет те же линии связи, что и сервис Python Matcher.

Admin. Пользователь может управлять аккаунтами и выполнять другие действия через пользовательский интерфейс сервиса Admin **(Adm1)**. Соответствующие запросы отправляются в сервис Admin **(Adm2)**.

Сервис Admin получает информацию об аккаунтах из сервиса Accounts **(Adm3)**.

Сервис Admin отправляет запросы в сервис Tasks **(Adm4)**. Эти задачи выполняются для всех данных БД, а не только для отдельного ID аккаунта.

Сервис Admin получает информацию о текущих лицензионных условиях из сервиса Licenses **(Adm5)**.

Сервис Admin получает количество лиц с привязанными атрибутами из базы данных Faces **(Adm6)** и подсчитывает процент заполненности базы данных.

Сервис Admin выполняет запросы ко всем сервисам LP, получая системную информацию (см. запрос «/luna_sys_info» в спецификации OpenAPI сервиса Admin) **(Adm7)**.

Sender. Все созданные события передаются сервису Sender через канал БД Redis **(H5)**. Внешние сторонние приложения подписываются на получение событий в соответствии с заданными фильтрами **(Se2)**. Сервис Sender проверяет канал Redis, получает требуемую информацию и отправляет ее стороннему ПО **(Se1)**.

> Сервис Sender включается/отключается в настройке «ADDITIONAL_SERVICE_USAGE».

Configurator. Сервисы LP получают конфигурационные параметры из сервиса Configurator **(Con1)**. Пользователь может управлять конфигурациями в сервисе Configurator посредством пользовательского интерфейса **(Con2)**. Изменения отправляются в Configurator **(Con3)**. Все изменения в конфигурационных файлах сохраняются в базе данных Configurator **(Con4)**.

> Сервис Configurator используется по умолчанию для каждого сервиса LUNA PLATFORM.

Tasks. Сервис Tasks получает запросы из сервиса API **(A8)**. Далее Tasks создает и сохраняет задачу в базе данных Tasks **(T1)**.

Далее сервис Tasks взаимодействует со своим «рабочим процессом» через Redis **(T2, T3)**, создавая подзадачи и объединяя результаты. См. подробную информацию в разделе «Диаграммы задач».

Сервис Tasks получает данные из сервиса Faces для задач Clustering, Linker и Additional extraction **(T4)**.

Сервис Tasks получает данные из сервиса Events для задач Clustering и Linker **(T9, T10)**.

Сервис Tasks получает данные из сервиса Handlers для задачи Estimator **(T6)**.

> Сервис Tasks включается/отключается в настройке «ADDITIONAL_SERVICE_USAGE».

Взаимодействие «рабочих процессов» Tasks с другими сервисами зависит от типа задачи.

«Рабочие процессы» Tasks получают данные из сервиса Faces для каждой обрабатываемой задачи **(T5)**.

«Рабочие процессы» Tasks отправляют запрос в Python Matcher **(T7)** для задач Clustering, Cross-matching и ROC-curve calculation.

«Рабочие процессы» Tasks хранят все отчеты в хранилище Image Store **(T8)**. «Рабочие процессы» Tasks также хранят и получают кластеры из Image Store.

«Рабочие процессы» Tasks отправляют запрос для выполнения повторного извлечения биометрических шаблонов в сервис Handlers **(T6)**.

Мониторинг. Система мониторинга получает запросы и ошибки от каждого сервиса **(Mon1)**. Эти данные хранятся в базе данных Influx **(Mon2)**. Затем данные мониторинга отправляются в Grafana для визуализации данных мониторинга **(Mon3)**. Более подробную информацию о мониторинге можно найти в разделе «Мониторинг».

> У сервисов Grafana и Influx DB есть собственные интерфейсы (см. раздел «Пользовательские интерфейсы»).

> Данная схема не содержит архитектуру сервисов Backport 3, Backport 4, Lambda и сервисов видеоаналитики (Video Agent и Video Manager). См. подробную архитектуру сервиса Backport 3 в разделе «Архитектура Backport 3» и сервиса Backport 4 в разделе «Архитектура Backport 4». См. диаграмму взаимодействия сервиса Lambda в разделе «Диаграммы lambda».

6 Описание сервисов

В этом разделе представлена более подробная информация о функциях сервисов LP.

Базы данных могут быть не указаны на поясняющих рисунках.

См. таблицу с потреблением ресурсов каждым из перечисленных ниже сервисов в разделе «Потребление ресурсов сервисами».

6.1 Общая информация о сервисах

6.1.1 «Рабочие процессы»

Для сервисов LUNA PLATFORM можно задавать количество «рабочих процессов» для использования дополнительных ресурсов и памяти системы для обработки запросов к сервису. Сервис автоматически запускает несколько процессов и распределяет запросы между процессами.

При запуске сервиса в Docker-контейнере, количество «рабочих процессов» задается с помощью параметра `WORKER_COUNT`.

Например, если выставить значение `WORKER_COUNT=2` для сервиса Faces, то сервис будет потреблять в 2 раза больше ресурсов и памяти.

> Обратите внимание на количество доступных основных компонентов на вашем сервере при использовании этой функции.

Использование «рабочих процессов» – это альтернативный способ линейного масштабирования сервисов. При увеличении количества экземпляров сервисов на одном сервере рекомендуется использовать дополнительные «рабочие процессы».

> Не рекомендуется использовать дополнительные «рабочие процессы» для сервиса Remote SDK, если он использует графический процессор. Проблемы могут возникнуть, если возникнет нехватка памяти графического процессора, и «рабочие процессы» будут мешать друг другу.

6.1.2 Автоматическая перезагрузка конфигураций

Сервисы LP поддерживают автоматическую перезагрузку конфигураций. При изменении параметра, он автоматически обновляется для всех экземпляров соответствующих сервисов. Если эта функция включена, нет необходимости перезапускать сервисы вручную.

Эта функция доступна для всех настроек, предусмотренных для каждого сервиса Python. Необходимо включать эту функцию вручную при каждом запуске сервиса. См. раздел «Включение автоматической перезагрузки конфигурации».

Начиная с версии 5.5.0, перезагрузка конфигурации для сервисов Faces и Python Matcher выполняется в основном путем перезапуска соответствующих процессов.

6.1.2.1 Ограничения

Сервис может работать некорректно при применении новых настроек. Настоятельно рекомендуется не отправлять запросы в сервис при изменении важных настроек (настройки БД, список рабочих плагинов и др.).

Применение новых настроек может привести к перезапуску сервиса и сбросу кэшей (например, кеш сервиса Python Matcher). Например, при изменении версии биометрического шаблона по умолчанию будет перезапущена платформа LP. Изменение уровня ведения журнала не вызывает перезапуска сервиса (если было указано допустимое значение параметра).

6.1.2.2 Включение автоматической перезагрузки конфигурации

Можно включить эту функцию, указав функцию `--config-reload` в командной строке. В Docker конейнерах эта функция включается с помощью параметра «RELOAD_CONFIG».

Можно указать период проверки конфигурации в аргументе командной строки `--pulling-time`. По умолчанию установлено значение 10 секунд. В Docker конейнерах эта функция включается с помощью параметра «RELOAD_CONFIG_INTERVAL».

6.1.2.3 Процесс обновления конфигураций

Сервисы LP периодически получают настройки из сервиса Configurator или файлов конфигурации. Это зависит от способа получения конфигураций для конкретного сервиса.

Каждый сервис сравнивает свои имеющиеся настройки с полученными:

- При изменении настроек сервиса они будут запрошены и применены.
  - Если запрос конфигураций не удался, сервис продолжит работу без внесения каких-либо изменений в существующие конфигурации;
  - Если проверка соединений с новыми настройками не удалась, сервис повторит попытку запроса новых конфигураций через 5 секунд. Сервис отключится после 5 неудачных попыток;
- Если текущие настройки и новые запрошенные настройки совпадают, сервис Configurator не будет выполнять никаких действий.

6.1.3 Выполнение переноса базы данных

Необходимо выполнить скрипт переноса, чтобы обновить структуру базы данных при обновлении до новых сборок LP. По умолчанию перенос автоматически применяется при запуске скрипта db_create.

> Этот метод может быть полезен при необходимости выполнить откат к предыдущей сборке LUNA PLATFORM или обновить структуру базы данных без изменения сохраненных данных. В любом случае, перед применением любых изменений рекомендуется создать резервную копию вашей базы данных.

Можно запускать перенос из контейнера или использовать одиночную команду.

6.1.3.1 Одиночная команда

Ниже приведен пример для сервиса Tasks.

```
docker run \
-v /etc/localtime:/etc/localtime:ro \
-v /tmp/logs/tasks:/srv/logs \
--rm \
--network=host \
dockerhub.visionlabs.ru/luna/luna-tasks:v.3.24.8 \
alembic -x luna-config=http://127.0.0.1:5070/1 upgrade head
```

6.1.3.2 Запуск из контейнера

Чтобы выполнить перенос из контейнера, необходимо выполнить следующие действия (пример приведен для сервиса Configurator):

- Зайти в Docker-контейнер сервиса. См. «Вход в контейнер» в руководстве по установке LP 5.
- Запустить перенос.

Для большинства сервисов параметры конфигурации должны быть получены от сервиса Configurator, а команда имеет следующий вид:

```
alembic -x luna-config=http://127.0.0.1:5070/1 upgrade head
```

`-x luna-config = http://127.0.0.1:5070/1` – указывает, что параметры конфигурации для переноса должны быть получены от сервиса Configurator.

Для сервиса Configurator параметры берутся из файла «srv/luna_configurator/configs/config.conf».

Для сервиса Configurator следует использовать следующую команду:

```
alembic upgrade head
```

- Выход из контейнера. Контейнер будет удален после выхода.

```
exit
```

6.2 Сервис API

LUNA API – это веб-сервис распознавания лиц. Он предоставляет интерфейс RESTful для взаимодействия с другими сервисами LUNA PLATFORM.

С помощью сервиса API можно отправлять запросы другим сервисам LP и выполнять следующие задачи:

- обработка и анализ изображений:
  - распознавание лиц/тел на фотографиях;
  - оценка атрибутов лица (возраст, пол, этническая принадлежность) и параметров лица (поза головы, эмоции, направление взгляда, атрибуты глаз, атрибуты рта);
  - оценка параметров тела (возраст, пол, аксессуары, головной убор, цвета верхней и нижней одежды, тип рукавов);
- поиск похожих лиц/тел в базе данных;
- хранение полученных атрибутов лиц в базах данных;
- создание списков для поиска;
- сбор статистики;
- гибкое управление запросами для удовлетворения требований обработки пользовательских данных.

6.3 Сервис Remote SDK

Сервис Remote SDK используется для:

- обнаружения лиц и оценки параметров лиц,
- обнаружения тел и оценки параметров тел,
- проверки изображений по указанным порогам и оценки параметров изображений,
- создания нормализованных изображений,
- извлечения базовых атрибутов и биометрического шаблона, в т.ч. агрегированных,
- обработки изображений на основе политик обработчиков и верификаторов.

> Обнаружение лиц, тел, извлечение БШ, оценка параметров и атрибутов выполняются с помощью нейронных сетей. Алгоритм со временем совершенствуется и появляются новые нейронные сети. Они могут отличаться друг от друга по производительности и точности. Выбирать нейронную сеть следует исходя из бизнес требований к работе системы.

6.3.1 Сервис Remote SDK с графическим процессором

Сервис Remote SDK может использовать GPU для вычислений вместо CPU. Для каждого экземпляра сервиса Remote SDK используется один GPU.

Извлечение атрибутов на графическом процессоре предусмотрено для получения максимальной производительности. Для входящих изображений выполняется пакетная обработка. При этом снижаются затраты на вычисления для изображения, но не обеспечивается минимальная задержка для каждого изображения.

Ускорение графического процессора разработано для приложений с высокой нагрузкой, где выполняются тысячи запросов в секунду. Нецелесообразно использовать ускорение графического процессора в сценариях с небольшой нагрузкой, если задержка имеет значение.

6.3.2 Агрегирование

На основе всех переданных в одном запросе изображений может быть получен единый набор базовых атрибутов и агрегированный биометрический шаблон. Кроме того, во время создания события выполняется агрегация полученных значений Liveness, эмоций, наличия медицинской маски для лиц и верхней и нижней частей тела, пола, возраста и аксессуаров для тел.

Для агрегированного БШ результаты сравнения являются более точными. Агрегирование рекомендуется использовать при получении с одной камеры нескольких изображений. Не гарантируется, что агрегированные БШ обеспечат улучшения в других случаях.

Считается, что каждый параметр агрегирован из БО. Для активации агрегирования атрибутов необходимо использовать параметр «aggregate_attributes» в запросах «extract attributes» (только для лиц) и «sdk». Агрегация значений Liveness, эмоций и масок для лиц и верхней части тела, пола,

возраста и аксессуаров для тел доступна с помощью параметра «aggregate_attributes» в запросе «generate events», при условии, что ранее в обработчике была произведена оценка этих параметров, а также в запросе «sdk».

> В ответе выдается массив «sample_id», даже если в запросе использовался только один биометрический образец. В этом случае в массив включается один идентификатор БО.

6.3.3 Форматы биометрических шаблонов

В LUNA PLATFORM доступна работа со следующими форматами биометрических шаблонов:

Формат БШ	Содержание файлов	Размер
SDK +	Набор байтов (сам БШ). ——————————————+ Набор байтов, указывающих версию. ——————————————+ Набор байтов сигнатуры.	Размер зависит от версии нейронной сети (см. раздел «Нейросети») ——————————————+ Размер равен 4 байтам. ——————————————+ Размер равен 4 байтам.
Raw	Набор байтов (сам БШ), закодированный в Base64	Размер зависит от версии нейронной сети (см. раздел «Нейросети»)
XPK-файлы	Файлы, которые хранят БШ в формате SDK	Размер зависит от количества БШ внутри файла

Примечание. Форматы Raw и XPK-файлы являются устаревшими. Рекомендуется работать с форматом SDK.

> SDK и Raw форматы могут быть напрямую привязаны к лицу или сохранены во временный атрибут (см. «Создание объектов с использованием внешних данных»).

В большинстве запросов на извлечение, биометрический шаблон записывается в базу данных как набор байтов, не выдаваясь в теле ответа.

Существует несколько запросов, с помощью которых можно получить биометрический шаблон в формате SDK:

- запрос «sdk»;
- запрос "get temporary attributes и "get temporary attribute.

> С помощью LUNA PLATFORM невозможно получить биометрические шаблоны в формате Raw и SDK. Можно использовать другое программное обеспечение VisionLabs для получения данных форматов (например, LUNA SDK). Биометрические шаблоны, полученные с помощью вы-

> шеописанных ресурсов или с помощью программного обеспечения VisionLabs называются «сырыми» биометрическими шаблонами.

Использование «сырых» биометрических шаблонов для сравнения

Вышеописанные форматы биометрических шаблонов могут быть использованы в запросах на использование «сырых» биометрических шаблонов.

Внешний БШ можно использовать как **эталон** в следующих ресурсах:

- «/matcher/faces»,
- «/matcher/bodies»,
- «/matcher/raw»,
- «/handlers/{handler_id}/events», когда установлена схема тела запроса «multipart/form-data»,
- «/verifiers/{verifier_id}/verifications», когда установлена схема тела запроса «multipart/form-data»,
- «/verifiers/{verifier_id}/raw».

Внешний БШ можно использовать как **кандидат** в следующих ресурсах:

- «/matcher/raw»,
- «/verifiers/{verifier_id}/raw».

6.3.4 Создание объектов с использованием внешних данных

Можно создать временный атрибут или лицо, отправив внешние базовые атрибуты и БШ в LUNA PLATFORM. Таким образом, вы можете хранить эти данные во внешнем хранилище и отправлять их в LP только для обработки запросов.

Можно создать атрибут или лицо с помощью:

- базовых атрибутов и их БО;
- биометрических шаблонов (БШ в чистом виде в Base64 или БШ SDK в Base64);
- базовых атрибутов и биометрических шаблонов с соответствующими данными.

> Биометрические образцы не являются обязательными и могут не использоваться при создании атрибута или лица.

Более подробная информация приведена в справочном руководстве сервиса API, в разделах «create temporary attribute» и «create face».

6.3.5 Проверка изображений на соответствие стандартам

Сервис Remote SDK позволяет проверить изображения по стандарту ISO/IEC 19794-5:2011 или указанным пользователем порогам с помощью трех способов:

- запрос «iso»

- параметр «estimate_face_quality» запроса «detect faces»
- группа параметров «face_quality» политики «detect_policy» запроса «generate events»

Например, необходимо, выполнить проверку является ли изображение подходящего формата, указав в качестве удовлетворительного условия форматы «JPEG» и «JPEG2000». Если изображение подходит под данное условие, система вернет значение «1», если же формат обрабатываемого изображение отличен от заданного условия, то система вернет значение «0». Если условия не заданы — система вернет оцененное значение формата изображения.

Подробная информация и перечень выполняемых оценок и проверок описан в разделе «Проверка изображений».

> Возможность выполнения проверки и оценки параметров изображений регулируется специальным параметром в файле лицензии.

6.3.6 Включение/отключение некоторых эстиматоров и детекторов

По умолчанию сервис Remote SDK запускается со всеми включенными эстиматорами и детекторами. При необходимости можно отключить использование некоторых эстиматоров или детекторов при запуске контейнера Remote SDK. Отключение ненужных эстиматоров позволяет экономить оперативную память или память GPU, поскольку при старте сервиса Remote SDK выполняется проверка возможности выполнения указанных оценок и загрузка нейронных сетей в память.

При отключении эстиматора или детектора можно также удалить его нейронную сеть из контейнера Remote SDK.

Отключение эстиматоров или детекторов возможно с помощью передачи аргументов с названиями эстиматоров в команду запуска сервиса Remote SDK. Аргументы передаются в контейнер с помощью переменной «EXTEND_CMD».

Список доступных эстиматоров:

Аргумент	Описание
--enable-all-estimators-by-default	включить все эстиматоры по умолчанию
--enable-human-detector	одновременный детектор тел и тел
--enable-face-detector	детектор лиц
--enable-body-detector	детектор тел
--enable-people-count-estimator	эстиматор количества людей
--enable-face-landmarks5-estimator	эстиматор 5 контрольных точек лица
--enable-face-landmarks68-estimator	эстиматор 68 контрольных точек лица
--enable-head-pose-estimator	эстиматор положения головы

Аргумент	Описание
--enable-liveness-estimator	эстиматор Liveness
--enable-deepfake-estimator	эстиматор Deepfake
--enable-fisheye-estimator	эстиматор бочкообразной дисторсии (эффекта FishEye)
--enable-face-detection-background-estimator	эстиматор фона изображения
--enable-face-warp-estimator	эстиматор биометрического образца лица
--enable-body-warp-estimator	эстиматор биометрического образца тела
--enable-quality-estimator	эстиматор качества изображения
--enable-image-color-type-estimator	эстиматор типа цвета по лицу
--enable-face-natural-light-estimator	эстиматор естественности освещения
--enable-eyes-estimator	эстиматор глаз
--enable-gaze-estimator	эстиматор направления взгляда
--enable-mouth-attributes-estimator	эстиматор атрибутов рта
--enable-emotions-estimator	эстиматор эмоций
--enable-mask-estimator	эстиматор маски
--enable-glasses-estimator	эстиматор очков
--enable-eyebrow-expression-estimator	эстиматор бровей
--enable-red-eyes-estimator	эстиматор красных глаз
--enable-headwear-estimator	эстиматор головного убора
--enable-basic-attributes-estimator	эстиматор базовых атрибутов
--enable-face-descriptor-estimator	эстиматор извлечения биометрического шаблона лица
--enable-body-descriptor-estimator	эстиматор извлечения биометрического шаблона тела
--enable-body-attributes-estimator	эстиматор атрибутов тел

Можно явно указать какие эстиматоры и детекторы включены или выключены с помощью передачи соответствующих аргументов в переменную «EXTEND_CMD», или же включить (по умолчанию) или выключить их все с помощью аргумента `enable-all-estimators-by-default`. Можно вы-

ключить использование всех эстиматоров и детекторов, а затем включить определенные эстиматоры с помощью передачи соответствующих аргументов.

Пример команды запуска сервиса Remote SDK с использованием только детектора лиц и эстиматоров биометрического образца лица и эмоций.

```
docker run \
...
--env=EXTEND_CMD="--enable-all-estimators-by-default=0 --enable-face-
    detector=1 --enable-face-warp-estimator=1 --enable-emotions-estimator=1"
    \
...
```

6.4 Сервис Handlers

Сервис Handlers используется для создания и хранения обработчиков и верификаторов.

Данные обработчиков и верификаторов хранятся в базе данных Handlers.

6.4.1 Отправка событий в сторонний сервис

В LUNA PLATFORM доступна возможность отправки уведомлений через веб-сокеты или веб-хуки (HTTP). Для этого предназначена политика «callbacks».

Отправка уведомлений через веб-сокеты

Политика «callbacks» с параметром `luna-ws-notification` предоставляет механизм уведомлений, основанный на принципах веб-сокетов. Этот тип callback'а позволяет получать события через веб-сокеты от сервиса Sender, который взаимодействует с сервисом Handlers через механизм pub/sub по каналу Redis. Это обеспечивает прямое, мгновенное обновление данных, используя двусторонний канал связи между клиентом и сервером.

Преимущества:

- Прямое, мгновенное обновление данных по веб-сокетам.
- Эффективное использование открытого двустороннего канала.
- Низкая задержка в передаче уведомлений.

> В предыдущих версиях LUNA PLATFORM использовалась политика «notification_policy». В настоящий момент она считается устаревшей и не рекомендуется к использованию. Основным преимуществом механизма callback'ов перед устаревшей «notification_policy» является возможность указания нескольких callback'ов с разными фильтрами в запросе на создание обработчика, в результате чего будет отправлено только одно событие.

См. подробную информацию в разделе «Сервис Sender».

Отправка уведомлений через веб-хуки

Политика «callbacks» с параметром `http` предоставляет механизм уведомлений, основанный на принципах веб-хуков для HTTP. Они обеспечивают асинхронное взаимодействие между системами, позволяя внешним сервисам реагировать на появление событий. В рамках этой политики можно задать конкретные параметры, такие как тип протокола, адрес внешней системы, параметры и данные авторизации.

Преимущества:

- Более гибкий механизм настройки уведомлений.
- Простая интеграция с различными внешними системами.
- Использует привычные HTTP-протоколы и конфигурации.

6.5 Сервис Image Store

Сервис Image Store хранит следующие данные:

- Биометрические образцы лица и тела. БО сохраняются в Image Store сервисом Remote SDK или с помощью запросов «samples» > «detect faces» и «samples» > «save face/body sample».
- Отчеты о задачах. Отчеты сохраняются «рабочими процессами» сервиса Tasks.
- Любые объекты, загружаемые с помощью запроса «create objects».
- Информацию о кластеризации.

Сервис Image Store имеет возможность сохранять данные либо на локальном накопителе, либо в S3-подобном облачном хранилище (например, Amazon S3 и др.).

6.5.1 Описание бакетов

Данные хранятся в специальных директориях, называемых бакетами. У каждого бакета есть уникальное имя. Имена бакетов должны быть заданы строчными буквами.

В LP используются следующие бакеты:

- «visionlabs-samples» — хранит БО лиц.
- «visionlabs-body-samples» — хранит БО тел.
- «visionlabs-image-origin» — хранит исходные изображения.
- «visionlabs-objects» — хранит объекты.
- «task-result» — хранит результаты, полученные после обработки задач с помощью сервиса Tasks.
- «portraits» — необходим для использования сервиса Backport 3. В бакете хранятся портреты (см. описание портретов в документации LUNA PLATFORM 3).

> Процедура создания бакетов описана в руководстве по установке LP 5 в разделе «Создание бакетов».

После запуска контейнера Image Store и команд для создания контейнеров, бакеты сохраняются в локальное хранилище или S3.

По умолчанию локальные файлы хранятся в каталоге «/var/lib/luna/current/example-docker/image_-store» на сервере. Они сохраняются в каталоге «/srv/local_storage/» в контейнере Image Store.

Бакет содержит каталоги с биометрическими образцами или другими данными. Названия каталогов соответствуют первым четырем буквам идентификатора БО. Все образцы распределяются по этим каталогам в соответствии с первыми четырьмя символами их идентификаторов.

Рядом с объектом бакета расположен файл «*.meta.json», содержащий «account_id», используемый при выполнении запроса. Если объект бакета не является биометрическим образцом (например, объект бакета — JSON-файл в бакете «task-result»), то в данном файле также будет указан «Content-Type».

Пример структуры директорий бакетов «visionlabs-samples», «task-result» и «visionlabs-bodies-samples» приведен ниже.

```
./local_storage/visionlabs-samples/8f4f/
            8f4f0070-c464-460b-sf78-fac234df32e9.jpg
            8f4f0070-c464-460b-sf78-fac234df32e9.meta.json
            8f4f1253-d542-621b-9rf7-ha52111hm5s0.jpg
            8f4f1253-d542-621b-9rf7-ha52111hm5s0.meta.json
./local_storage/task-result/1b03/
            1b0359af-ecd8-4712-8fc0-08401612d39b
            1b0359af-ecd8-4712-8fc0-08401612d39b.meta.json
./local_storage/visionlabs-bodies-samples/6e98/
            6e987e9c-1c9c-4139-9ef4-4a78b8ab6eb6.jpg
            6e987e9c-1c9c-4139-9ef4-4a78b8ab6eb6.meta.json
```

> При хранении большого количества биометрических образцов может потребоваться значительный объем памяти. Один биометрический образец занимает около 30 Кбайт дискового пространства.

Также рекомендуется создавать резервные копии БО. БО используются при изменении версии нейросети или при необходимости восстановить базу данных лиц.

6.5.2 Использование S3-подобного хранилища

Для включения использования S3-подобного хранилища необходимо выполнить следующие действия:

- убедиться, что ключ доступа имеет достаточные полномочия для доступа к бакетам S3-подобного хранилища;
- запустить сервис Image Store (см. раздел «Image Store» в руководстве по установке);
- задать значение «S3» для настройки «storage_type» настроек сервиса Image Store;
- заполнить настройки для соединения с S3-подобным хранилищем (адрес, ключи Access Key и Secret Key и др.) в группе параметров «S3» настроек сервиса Image Store;
- запустить скрипт по созданию бакетов `lis_bucket_create.py` (см. раздел «Создание бакетов» в руководстве по установке)

При необходимости можно отключить проверку SSL-сертификата с помощью настройки «verify_ssl» в группе параметров «S3» настроек сервиса Image Store. Это позволяет использовать самоподписанный SSL-сертификат.

6.5.3 TTL объектов

Можно задать время жизни (TTL) для объектов в бакетах (как локальных, так и S3). Под объектами понимаются:

- биометрические образцы лиц или тел
- изображения или объекты, создаваемые в ресурсах «/images» или «/objects»
- исходные изображения
- результаты задач

TTL для объектов рассчитывается относительно формата времени GMT.

TTL для объектов задается в днях следующими способами:

- во время создания бакета для всех объектов сразу (основная политика TTL бакета)
- после создания бакета для конкретных объектов с помощью запросов к соответствующим ресурсам

Количество дней выбирается из списка в соответствующих запросах (см. ниже).

Кроме количества дней параметр может принимать значение «-1», означающее, что нужно хранить объекты бессрочно.

6.5.3.1 Настройка основной политики TTL бакета

Основную политику TTL бакета можно настроить следующими способами:

- с помощью флага `--bucket-ttl` для скрипта `lis_bucket_create.py`. Например, `python3 ./base_scripts/lis_bucket_create.py -ii --bucket-ttl=2`.
- с помощью запроса к сервису Image Store. Например, `curl -X POST http://127.0.0.1:5020/1/buckets?bucket=visionlabs-samples?ttl=2`.

6.5.3.2 Настройка TTL для конкретных объектов

TTL для конкретных объектов можно настроить с помощью параметра «ttl» в следующих местах:

- в политиках «storage_policy» > «face_sample_policy», «body_sample_policy» и «image_origin_policy» обработчика
- в запросах «create object», «create images» и «save sample»
- в запросах на создание задач или расписания в поле «result_storage_policy»

Если параметр «ttl» не задан, то будет применена основная политика бакета, в котором находится объект (см. выше).

6.5.3.3 Добавление TTL к существующим объектам

Добавить TTL для существующего конкретного объекта можно с помощью PUT-запросов к ресурсам `/objects`, `/images`, `/samples/{sample_type}` сервиса Image Store. Добавить TTL результа-

тов задач к уже созданным и выполненным задачам невозможно. Добавить TTL результатов задач к уже созданному расписанию можно с помощью запроса «replace tasks schedule». Для созданных или запущенных на момент запроса задач TTL результатов задач применен не будет.

Добавить TTL к основной политике бакета (ко всем существующим объектам в бакете) можно с помощью PUT-запроса к ресурсу `/buckets` сервиса Image Store.

Для бакета, расположенного в S3, необходимо выполнить миграцию с помощью скрипта «base_-scripts/migrate_ttl_settings» сервиса Image Store. Это связано с тем, что для TTL объектов в S3 применяется через фильтры, связанные с тегами. Команда выполнения миграции бакетов в S3 приведена в руководстве по установке. См. подробную информацию о миграции бакетов S3 в разделе «Миграция для добавления TTL к объектам в S3».

6.5.3.4 Поддерживаемые облачные провайдеры

Поддерживаются облачные провайдеры Amazon S3, облачное хранилище Яндекса и MinIO.

6.5.3.5 Миграция для добавления TTL к объектам в S3

Настройка жизненного цикла для объектов в S3 применяется через фильтры, связанные с тегами (см. официальную документацию). Это предполагает, что объекты имеют тег с ограниченным набором значений, а бакеты имеют набор правил, основанных на значении этого тега.

Для добавления тегов и правил необходимо выполнить миграцию. Миграция строго необходима для полного применения настройки жизненного цикла по следующим причинам:

- бакеты без правил не будут удалять объекты, даже если пользователь указывает время жизни для конкретного объекта,
- объекты без тегов никогда не будут удалены, даже если пользователь указывает время жизни бакета.

Необходимо добавить следующие теги и правила:

- для поддержки TTL для бакетов необходимо добавить тег `vl-expire` со значением по умолчанию для всех существующих объектов.
- для поддержки TTL для конкретных объектов необходимо добавить набор или связанные с TTL правила жизненного цикла для существующих сегментов:

```
{
    "ID": "vl-expire-<ttl>",
    "Expiration": {
        "Days": <ttl>,
    },
    "Filter": {"Tag": {"Key": "vl-expire", "Value": <ttl>}},
    "Status": "Enabled",
```

```
    }
```

Поддерживается набор определенных значений тегов, связанных с TTL объекта: 1, 2, 3, 4, 5, 6, 7, 14, 30, 60, 90, 180, 365.

Процесс выполнения миграции состоит из двух этапов:

- настройка жизненного цикла бакета, расширенная набором правил жизненного цикла, связанных с TTL.
- присвоение каждому объекту в бакете тега `vl-expire`, если он еще не обладает таковым.

Присвоение тега для каждого объекта при необходимости можно пропустить с помощью аргумента `-update-tags=0`.

См. примеры команд для выполнения миграции в руководстве по обновлению.

Проблемы с разрешениями

По умолчанию все ресурсы S3, включая бакеты, объекты и настройку TTL, являются приватными. При необходимости правила и теги по умолчанию могут быть созданы владельцем ресурса вручную одним из применимых методов. Подробности см. в официальной документации S3.

Полезные ссылки на официальную документацию:

- Creating a lifecycle configuration
- Put lifecycle configuration
- Managing object tags
- Expiring objects

6.5.3.6 Окончание TTL

Когда TTL объекта подходит к концу, то он помечается для удаления. Для локальных бакетов выполняется задача очистки 1 раз в день (в 01:00 утра). Для бакетов S3 используются внутренние правила конфигурации TTL. Чтобы предотвратить конфликты или дублирование задач очистки при задействовании нескольких экземпляров или рабочих процессов, реализован механизм блокировки. Это гарантирует, что за выполнение процесса очистки локального хранилища отвечает только один экземпляр или рабочий процесс.

> Между датой истечения срока действия и датой фактического удаления объекта может возникнуть задержка. Как для локального хранилища, так и для S3.

6.5.3.7 Поиск истекающих объектов

Чтобы узнать, когда срок действия объекта истекает, можно использовать запросы с методами HEAD к ресурсам `/objects` и `/images`. Эти запросы возвращают заголовки ответов `X-Luna-Expiry-Date`, в которых указана дата, когда объект больше не подлежит сохранению.

6.5.4 Внешние биометрические образцы

В Image Store можно отправить внешний биометрический образец. Внешний биометрический образец можно получить с помощью стороннего программного обеспечения или программного обеспечения VisionLabs (например, FaceStream).

См. запрос POST на ресурсе «/samples/{sample_type}» в «APIReferenceManual.html» для получения дополнительной информации.

Внешний БО должен соответствовать определенным стандартам, чтобы LP могла его обработать.

БО, полученные с помощью программного обеспечения VisionLabs, удовлетворяют этому требованию.

При использовании стороннего программного обеспечения не гарантируется, что результат обработки внешнего БО будет таким же, как и для образца VisionLabs. БО может быть низкого качества (слишком темный, размытый и т.д.). В результате низкого качества можно получить неверные результаты обработки изображения.

> Рекомендуется проконсультироваться с представителями VisionLabs перед использованием внешних БО.

6.6 Сервис Accounts

Сервис Accounts предназначен для:

- Создания, управления и хранения аккаунтов
- Создания, управления и хранения токенов и их разрешений
- Верификации аккаунтов и токенов

См. раздел «Аккаунты, токены и способы авторизации» для более подробной информации о системе авторизации в LUNA PLATFORM 5.

Все создаваемые аккаунты, токены и их разрешения сохраняются БД сервиса Accounts.

6.6.1 Алгоритмы JWT-токенов

Механизм аутентификации JWT (JSON Web Tokens) поддерживает различные алгоритмы для подписи токенов. В этом разделе описывается используемый по умолчанию алгоритм и необходимые шаги для использования альтернативного алгоритма.

6.6.1.1 Алгоритм по умолчанию

По умолчанию сервис использует алгоритм HS256 для подписи JWT-токенов. При необходимости можно использования асимметричного криптографического шифрования, вы можете использовать алгоритм ES256.

6.6.1.2 Использование алгоритма ES256

Чтобы использовать алгоритм ES256, следуйте этим шагам:

1. Сгенерируйте закрытый ключ ECDSA

   Сначала нужно сгенерировать закрытый ключ ECDSA, используя кривую `prime256v1`. Это можно сделать с помощью инструментов командной строки, таких как OpenSSL.

   Пример команды:

```
openssl ecparam -genkey -name prime256v1 -out ec_private.pem
```

   Вы также можете сгенерировать ключ, защищенный паролем, например:

```
openssl ecparam -genkey -name prime256v1 | openssl ec -aes256 -out
   ec_private_enc.pem
```

2. Закодируйте закрытый ключ в формат Base64

После генерации закрытого ключа закодируйте его в формат Base64. Это можно сделать с помощью инструментов, доступных в большинстве операционных систем.

Пример команды:

```
base64 -w0 ecdsa_private.pem > ecdsa_private_base64
```

3. Установите переменную окружения

   Закодированный закрытый ключ должен быть указан в переменной окружения `ACCOUNTS_JWT_ECDSA_K` при старте контейнера. Это позволяет сервису использовать ключ для подписи JWT-токенов алгоритмом ES256.

   Кроме того, если ваш закрытый ключ защищен паролем, вы можете указать пароль в переменной среды `ACCOUNTS_JWT_ECDSA_KEY_PASSWORD`.

   Пример команды запуска контейнера с передачей переменных окружения:

```
docker run \
--env=CONFIGURATOR_HOST=127.0.0.1 \
--env=ACCOUNTS_JWT_ECDSA_KEY=jwt_ecdsa_key \
--env=ACCOUNTS_JWT_ECDSA_KEY_PASSWORD=ecdsa_key_password \
...
```

Следуя этим шагам, сервис сможет подписывать JWT-токены с использованием алгоритма ES256, обеспечивая повышенную безопасность с помощью асимметричной криптографии.

6.6.1.3 Влияние изменения типа алгоритма

Переключение алгоритма подписи с HS256 на ES256 (или наоборот) оказывает значительное влияние на проверку токенов. **Все существующие токены, подписанные предыдущим алгоритмом, станут недействительными** после внесения изменений. Это происходит потому, что механизм проверки подписи токена ожидает, что структура и криптографическая база токена будут соответствовать новоуказанному алгоритму.

6.7 Сервис Faces

Сервис Faces предназначен для:

- Создания временных атрибутов;
- Создания лиц;
- Создания списков;
- Прикрепления лиц к спискам;
- Управления общей базой данных, в которой хранятся лица с прикрепленными данными и списки;
- Получения информации о существующих лицах и списках.

6.8 Сервисы сравнения

Python Matcher позволяет осуществлять:

- Сравнение в соответствии с заданными фильтрами. Такое сравнение выполняется непосредственно в базе данных лиц или событий. Сравнение по БД целесообразно, когда установлено несколько фильтров.
- Сравнение по спискам. В этом случае рекомендуется сохранять биометрические шаблоны в кеше Python Matcher.

Python Matcher Proxy используется для маршрутизации запросов к сервисам Python Matcher и плагинам сравнения.

6.8.1 Python Matcher

Python Matcher использует базу данных Faces для фильтрации и сравнения, когда лица заданы как кандидаты для сравнения и для них указаны фильтры. Эта функция всегда включена для Python Matcher.

Python Matcher использует базу данных Events для фильтрации и сравнения, когда события заданы как кандидаты для сравнения и для них указаны фильтры. Сравнение с использованием базы данных Events является необязательным и не используется, если не используется сервис Events.

Для сравнения по БД требуется функция сравнения VLMatch. Она должна быть зарегистрирована для БД Faces и БД Events. Функция использует библиотеку, которая должна быть скомпилирована для вашей текущей версии БД. Информацию об этом можно найти в руководстве по установке в разделах «Компиляция библиотеки VLMatch», «Создание функции VLMatch для БД Faces» и «Создание функции VLMatch для БД Events».

Сервис Python Matcher дополнительно использует «рабочие процессы», обрабатывающие запросы.

6.8.2 Python Matcher Proxy

Сервис API отправляет запросы на прокси-сервер Python Matcher, если его использование включено в настройках сервиса API. Затем сервис Python Matcher Proxy перенаправляет запросы к сервису Python Matcher или на плагины сравнения (если они используются).

Если плагины сравнения не используются, то сервис перенаправляет запросы только к сервису Python Matcher. Таким образом, не нужно использовать Python Matcher Proxy если не собираетесь использовать плагины сравнения. См. описание работы плагинов сравнения в разделе «Плагины сравнения».

6.8.3 Кеширование списков

Когда лица заданы как кандидаты для сравнения и идентификаторы списков для них указаны в качестве фильтров, Python Matcher выполняет сравнение по спискам.

По умолчанию при запуске сервиса Python Matcher все биометрические шаблоны во всех списках кешируются в его память.

> Управление кешированием осуществляется за счет группы параметров «DESCRIPTORS_CACHE».

> Сервис Python Matcher не будет запущен, пока не загрузит в кеш все доступные биометрические шаблоны.

При выполнении запроса на сравнение по спискам, сервис Python Matcher автоматически добавляет его в очередь, откуда его забирает «рабочий процесс» и направляет в сущность Cached Matcher для выполнения сравнения по кешированным данным.

После выполнения сравнения, «рабочий процесс» забирает результаты и возвращает их сервису Python Matcher и пользователю.

Такое кеширование позволяет значительно увеличить производительность сравнения.

При необходимости можно обрабатывать только конкретные списки с помощью настройки «cached_data > faces_lists > include» или исключать списки с помощью настройки «cached_data > faces_lists > exclude». Последняя особенно полезна при работе с модулем LUNA Index Module для реализации логики обработки части списков с помощью Python Matcher, а части с помощью LIM Indexed Matcher.

> См. подробную информацию о LIM в разделе «Сравнение большого набора биометрических шаблонов».

6.8.3.1 Кеш «рабочих процессов»

Когда для сервиса Python Matcher запускается несколько «рабочих процессов», каждый из «рабочих процессов» использует один и тот же кеш биометрических шаблонов.

Его изменение может как ускорить, так и замедлить работу сервиса. Если нужно убедиться, что кеш хранится в каждом из процессов Python Matcher, необходимо запустить каждый из экземпляров сервера отдельно.

6.9 Сервис Events

Сервис Events предназначен для:

- Хранения всех созданных событий в базе данных Events.
- Выдачи всех событий в соответствии с фильтрами.
- Сбора статистики по всем существующим событиям в соответствии с заданной агрегацией и частотой/периодичностью.
- Хранения биометрических шаблонов, созданных для событий.

> Поскольку событие представляет собой отчет, нельзя изменить уже существующие события.

Сервис Events должен быть активирован в файле конфигурации сервиса API. В противном случае события не будут сохраняться в базе данных.

6.9.1 База данных для сервиса Events

В качестве базы данных для сервиса Events используется PostgreSQL.

На скорость обработки запросов в первую очередь влияют:

- количество событий в базе данных
- отсутствие индексов для PostgreSQL

PostgreSQL показывает допустимую скорость обработки запросов, если количество событий составляет от 1 000 000 до 10 000 000. Если количество событий превышает 10 000 000, запрос к PostgreSQL может завершиться неудачей.

> Скорость обработки запросов статистики к СУБД PostgreSQL можно увеличить, настроив базу данных и создав индексы.

6.9.2 Географическое положение

При создании события можно добавить географическое положение.

Географического положение представлено в виде JSON с GPS-координатами географической точки:

- долгота – географическая долгота в градусах
- широта – географическая широта в градусах

Географическое положение указывается в параметре «location» тела запроса на создание события. См. запрос «create new events» в спецификации OpenAPI сервиса Events.

Можно использовать фильтр географического положения, чтобы получить все события, которые произошли в соответствующей области.

6.9.2.1 Фильтр географического положения

Фильтр географического положения – это ограничивающий прямоугольник, заданный координатами его центра (начало) и некоторой дельтой.

Он задается с использованием следующих параметров:

- origin_longitude
- origin_latitude
- longitude_delta
- latitude_delta

Фильтр можно использовать при получении событий, получении статистики по событиям и выполнении сравнения событий.

Фильтр географического положения считается правильно заданным, если:

- заданы и *origin_longitude*, и *origin_latitude*.
- не заданы ни *origin_longitude*, ни *origin_latitude*, ни *longitude_delta* или *latitude_delta*.

Если заданы параметры *origin_longitude* и *origin_latitude*, а параметры *longitude_delta* не заданы – применяется значение по умолчанию (см. значение по умолчанию в документации OpenAPI).

> Прочтите следующие рекомендации перед использованием фильтров географического положения.

Общие рекомендации и ограничения для фильтров географического положения:

- Не нужно создавать фильтры с общей точкой или границей на Международной линии перемены дат (IDL), Северном или Южном полюсе. Они не полностью поддерживаются из-за особенностей пространственного индекса базы данных. Результат фильтрации может быть **непредсказуемым**;
- Фильтры географического положения с границами длиной более 180 градусов не допускаются;
- Настоятельно рекомендуется использовать фильтр географического положения только в масштабах города. Если указана большая область, результаты фильтрации на границах области могут быть неожиданными из-за пространственных особенностей.
- Не нужно создавать фильтр слишком протяжённый по долготе или широте. Рекомендуется устанавливать значения дельт близкими друг к другу.

Последние две рекомендации приводятся из-за пространственных особенностей фильтра. В соответствии с этими функциями, если установлены большие значения одной или двух дельт, результат может отличаться от предполагаемого, несмотря на то, что он будет правильным. См. подробную информацию в разделе «Особенности фильтра».

6.9.2.2 Эффективность фильтра

Эффективность фильтра географического положения зависит от типа пространственных данных, используемых для сохранения географического положения события в базе данных.

Поддерживаются два типа пространственных данных:

- GEOMETRY (геометрический): пространственный объект с координатами, выраженными в виде пар (долгота, широта), определенных в декартовой плоскости. При всех расчетах используются декартовы координаты.
- GEOGRAPHY (географический): пространственный объект с координатами, выраженными в виде пар (долгота, широта), определенных как на поверхности идеальной сферы, или пространственный объект в системе координат WGS84.

Подробное описание см. в разделе геометрия в сравнении с географией.

Фильтр географического положения на основе функции PostGIS ST_Covers, поддерживается как для геометрического, так и для географического типа.

6.9.2.3 Особенности фильтра

Фильтр географического положения имеет некоторые особенности, обусловленные PostGIS.

Если при использовании географического типа фильтр географического положения покрывает большую часть поверхности планеты, результат фильтрации может быть неожиданным, но географически правильным из-за некоторых пространственных особенностей.

Например, в базу данных добавлено событие со следующей географической позицией:

```
{
    "longitude": 16.79,
    "latitude": 64.92,
}
```

Далее применяется фильтр географического положения и производится попытка найти нужную точку на карте. В результате фильтр получается слишком большим по долготе:

```
{
    "origin_longitude": 16.79,
    "origin_latitude": 64.92,
    "longitude_delta": 2,
    "latitude_delta": 0.01,
}
```

Таким образом, этот фильтр **не выдает** ожидаемое событие. Событие фильтруется по пространственным особенностям. В данном примере показан случай, когда точка находится за пределами

фильтра.

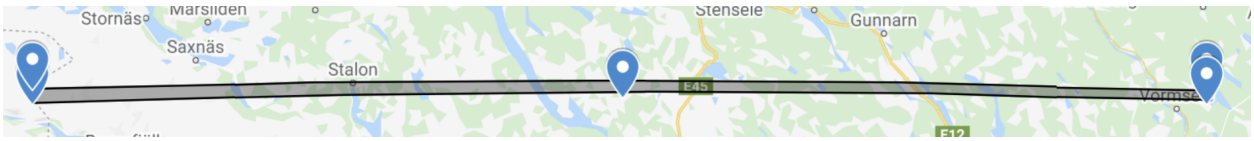


Рис. 30: Пример слишком широкой зоны

Необходимо учитывать эту особенность при создании фильтра.

Подробная информация приведена в разделе Geography сайта Postgis.

6.9.3 Создание событий

События создаются с помощью обработчиков. Обработчики хранятся в базе данных Handlers. Необходимо указать требуемый идентификатор обработчика в запросе на создание события. Все данные, хранящиеся в событии, будут получены в соответствии с параметрами обработчика.

Необходимо выполнить два отдельных запроса на создание события.

Первый запрос создает обработчик. Обработчик содержит политики, которые определяют, как обрабатывается изображение, и, следовательно, определяют сервисы LP, используемые для обработки.

Второй запрос создает новые события с использованием существующего обработчика. Событие создается для каждого обработанного изображения.

Можно указать следующие дополнительные данные для каждого запроса на создание события:

- внешний идентификатор (для созданных лиц),
- пользовательские данные (для созданных лиц),
- источник (для созданных событий),
- теги (для созданных событий).

Обработчик обрабатывает политику за политикой. Все данные из запроса обрабатываются политикой перед переходом к следующей политике. Сначала политика «detect» выполняется для всех изображений из запроса, затем применяется политика «multiface», затем выполняется политика «extract» для всех полученных биометрических образцов и т.д. Более подробную информацию об обработчиках см. в разделе «Описание обработчиков».

6.9.4 Обобщенные события

В случае, если необходимо сохранить событие с какой-либо пользовательской структурой, сгенерированной видеоаналитикой, плагином или каким-либо клиентским приложением, следует

использовать механизм **обобщенных событий**. Размер обобщенного события не должен превышать лимит в 2 МБ.

Поскольку обобщенные события имеют свободную структуру, они должны быть разделимы по типу (параметр «event_type»), который следует указать при сохранении события.

При сохранении обобщенного события также следует предоставить некоторую общую информацию (например, время создания события, время окончания события, аккаунт, которому принадлежит событие).

Несмотря на то, что структура обобщенного события свободна, она может содержать определенный набор полей, специфичных для LUNA PLATFORM (например, местоположение, идентификатор потока, идентификатор трека), поиск по которым может выполняться более эффективно за счет оптимизации хранения этих данных в базе данных Events.

Пользователю предоставляется:

- Возможность сохранять события свободной структуры.
- Возможность получать сохраненные события.
- Возможность фильтровать по общей информации обобщенного события (например, `event_type`, `event_create_time`, `event_end_time`, `account_id`).
- Возможность фильтровать по необязательным полям обобщенного события, присущим LUNA PLATFORM (например, `location`, `stream_id`, `track_id`).
- Возможность фильтровать по полям содержимого пользовательской структуры.

Пользователь обязуется:

- Поддерживать данные в соответствии с заданными схемами. В случае несоответствия PostgreSQL не позволит вставить строку с типом, который не может быть добавлен в существующий индекс (если таковой имеется).
- Мигрировать данные при необходимости.
- Строить частичные индексы в соответствии с типами событий.
- Указывать тип данных при выполнении запроса (по умолчанию все значения предполагаются строками).
- Обращать внимание на имена полей пользовательской структуры. Поля, по которым будет выполняться фильтрация, не должны содержать зарезервированных ключевых слов, таких как `:int`, двойных подчеркиваний, специальных символов и т. д.

См. подробную информацию о построениии индексов и решению различных проблем с обобщенными событиями в руководстве разработчика сервиса Events.

6.9.5 Способ сохранения обобщенных событий

Сохранить обощенное событие можно с помощью запроса «create new general events» к сервису API.

Соответственно, если пользователь пишет собственный плагин или приложение, то он должен отправлять данные в вышеописанный эндпоинт.

6.9.6 Поиск обобщенных событий

Для поиска событий предназначены следующие запросы:

- «get general events»
- «get general event»

В данные запросы можно передать фильтры. Существует несколько соглашений для указания фильтров:

- Для навигации по вложенным объектам используйте точку (`.`), например, `event.user_info.temperature`.
- Для указания оператора сравнения используйте суффикс с двойным подчеркиванием (`__eq`, `__like`, `__in`, `__gt`, `__lt` и т. д.), например, `event.user_info.temperature__gte`.
- Для указания типа данных используйте суффикс с двоеточием (`:string`, `:integer`, `:numeric`), например, `event.user_info.temperature__gte:numeric`.

Также можно получить статистику по обобщенным событиям с помощью запроса «get statistics on general events».

6.9.7 Метаинформация события

В случае, если вместе с событием необходимо сохранить какие-либо дополнительные данные, следует использовать поле «meta». Поле «meta» хранит в себе данные формата JSON. Общий размер данных, хранимых в поле «meta» для одного события не может превышать 2 Мб. Предполагается, что с помощью данного функционала пользователь создаст свою модель данных (структуру события) и будет использовать её для хранения необходимых данных.

> Обратите внимание, что в поле «meta» нельзя указывать имена полей с пробелами.

Данные в поле «meta» можно записывать следующими способами:

- в теле запроса «generate events» с типом содержимого `application/json` или `multipart/form-data`
- в теле запроса «save event»
- с помощью пользовательского плагина или клиентского приложения.

В теле запроса «generate events» доступна возможность задать поле «meta» как для конкретных изображений, так и для всех изображений сразу (взаимная метаинформация). Для запросов с включенной агрегацией для агрегированного события будет использоваться только взаимная метаинформация, а метаинформация для конкретных изображений будет игнорироваться. См. подробную информацию в теле запроса «generate events» в спецификации OpenAPI.

Пример записи поля «meta»:

```
{
    "meta": {
        "user_info": {
            "temperature": 36.6
        }
    }
}
```

Для того, чтобы хранить несколько структур, необходимо явно разделять их, чтобы избежать пересечения полей. Например, следующим образом:

```
{
    "struct1": {
        ...
    },
    "struct2": {
        ...
    }
}
```

6.9.7.1 Поиск по полю «meta»

Содержимое поля «meta» можно получить с помощью соответствующего фильтра в запросе «get events».

Фильтр нужно вводить с помощью определенного синтаксиса — `meta.<path.to.field>__<operator>:<type>`, где:

- `meta.` — указание, что идет обращение к полю «meta» базы данных Events;
- `<path.to.field>` — путь до объекта. Для навигации по вложенным объектам используется точка (`.`). Например, в строке `{"user_info":{"temperature":"36.6"}}` для обращения к объекту `temperature` нужно использовать следующий фильтр `meta.user_info.temperature`
- `__<operator>` — один из следующих операторов — `eq` (по умолчанию), `neq`, `like`, `nlike`, `in`, `nin`, `gt`, `gte`, `lt`, `lte`. Например, `meta.user_info.temperature__gte`;
- `:type` — один из следующих типов данных — `string`, `integer`, `numeric`. Например, `meta.user_info.temperature__gte:numeric`.

> Для каждого оператора доступно использование определенных типов данных. См. таблицу зависимости операторов от типов данных в спецификации OpenAPI.

При необходимости можно построить индекс для улучшения поиска. См. подробную информацию о построении индекса в руководстве разработчика сервиса Events.

6.9.7.2 Важные замечания

При работе с полем «meta» необходимо помнить следующее:

- нужно сохранять данные в соответствии с заданными схемами; в случае несоответствия, PostgreSQL не позволит вставить строку с типом, который не может быть добавлен в существующий индекс (если таковой имеется);
- при необходимости можно мигрировать данные;
- при необходимости можно построить индекс;
- нужно указывать тип данных при выполнении запроса (по умолчанию предполагается, что все значения являются строками);
- нужно обращать внимание на названия полей; поля, по которым производится фильтрация, не должны содержать зарезервированных ключевых слов, таких как `:int`, двойные подчеркивания, специальные символы и так далее.

6.9.8 Сохранение удаленных событий

Доступа возможность сохранения удаленных событий (обычных и обобщенных) в базу данных Events.

Удаленные события хранятся в БД Events в таблицах `deleted_event` и `deleted_general_event`. См. раздел «Описание БД Events».

Для включения этой функциональности необходимо установить переменную окружения `EVENT_DELETION_LOG=1` в команде запуска сервиса Events. Например:

```
docker run \
--env=CONFIGURATOR_HOST=127.0.0.1 \
--env=CONFIGURATOR_PORT=5070 \
--env=PORT=5040 \
--env=EVENT_DELETION_LOG=1 \
--name=luna-events \
...
```

После запуска сервиса идентификаторы удаленных событий будут записаны в соответствующие таблицы БД.

Получить удаленные события можно с помощью следующих запросов:

- «get deleted events»
- «get deleted general events»

6.10 Сервис Sender

Сервис Sender – это дополнительный сервис, который используется для отправки событий через веб-сокеты. Данный сервис коммуницирует с сервисом Handlers (в котором создаются события) через механизм pub/sub по каналу БД Redis.

> При необходимости можно отправлять уведомления по HTTP-протоколу. См. раздел «Отправка событий в сторонний сервис» для более подробной информации.

События создаются на основе обработчиков. Для получения уведомлений необходимо наличие включенной политики «callbacks» с параметром «luna-ws-notification». У данной политики есть фильтры, позволяющие отправлять уведомления только при определенных условиях, например, отправлять только при большом сходстве кандидата с эталоном (параметр ”similarity__lte”).

> В предыдущих версиях LUNA PLATFORM использовалась политика «notification_policy». В настоящий момент она считается устаревшей и не рекомендуется к использованию. Основным преимуществом механизма callback’ов перед устаревшей «notification_policy» является возможность указания нескольких callback’ов с разными фильтрами в запросе на создание обработчика, в результате чего будет отправлено только одно событие.

Необходимо настроить подключение к веб-сокетам по специальному запросу. Рекомендуется создавать соединение через веб-сокеты, используя ресурс «/ws» сервиса API. В запросе можно указать фильтры (параметры запроса), т.е. можно настроить сервис Sender так, чтобы получать только определенные события. См. спецификацию OpenAPI для получения подробной информации о конфигурации создания подключения к веб-сокету.

Также можно настроить веб-сокеты напрямую через сервис Sender (ресурс «/ws» сервиса Sender). Этот способ можно использовать для снижения нагрузки на сервис API.

После создания события, оно может:

- сохраниться в базе данных сервиса Events. Для сохранения события необходимо включить сервис Events;
- быть выдано в ответ без сохранения в базе данных.

В обоих случаях событие отправляется через канал БД Redis в сервис Sender.

> БД Redis в данном случае выступает в качестве связи сервисов Sender и Handlers и не хранит передаваемые события.

> Сервис Sender не зависит от сервиса Events. События могут отправляться в сервис Sender, даже если сервис Events отключен.

Общий процесс работы выглядит следующим образом:

Создание обработчиков и указание фильтров для отправки уведомлений

1. Пользователь отправляет запрос «create handler» в сервис API, где включает политику «callbacks» и задает фильтры, в соответствии с которыми будет выполняться отправка событий в сервис Sender;
2. Сервис API отправляет запрос в сервис Handlers;
3. Сервис Handlers отправляет ответ в сервис API;
4. Сервис API отправляет «handler_id» пользователю.

Пользователь сохраняет идентификатор «handler_id», который необходим для создания событий.

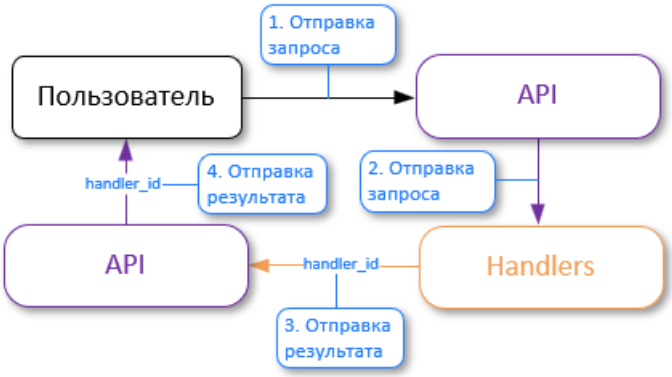


Рис. 31: Создание обработчиков и указание фильтров для отправки уведомлений

Активация подписки на события и фильтрация их отправки

1. Пользователь или приложение отправляет запрос «ws handshake» в сервис API и задает фильтры, благодаря которым можно будет фильтровать полученные данные от сервиса Handlers;
2. Сервис API отправляет запрос в сервис Sender;
3. Сервис Sender устанавливает постоянное соединение через веб-сокеты с пользовательским приложением.
4. Теперь при генерации события, оно будет автоматически перенаправляться в сервис Sender (см. ниже) в соответствии с указанными фильтрами.

Теперь при генерации события, оно будет автоматически перенаправляться в сервис Sender (см. ниже) в соответствии с указанными фильтрами.

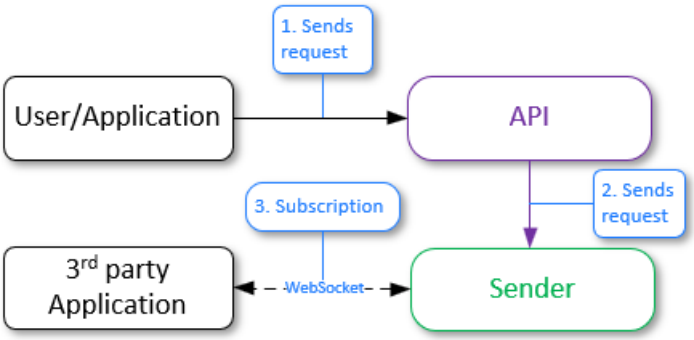


Рис. 32: Активация подписки на события и фильтрация их отправки

Генерация событий и отправка в Sender

1. Пользователь или приложение отправляет запрос «generate events» в сервис API;
2. Сервис API отправляет запрос в сервис Handlers;
3. Сервис Handlers отправляет запрос в соответствующие сервисы LP;
4. Сервисы LP обрабатывают запросы и отправляют результаты. Создаются новые события;
5. Сервис Handlers отправляет событие в базу данных Redis, используя модель pub/sub. В Redis есть канал, на который подписан сервис Sender, и он ожидает получение сообщений из этого канала;
6. Redis отправляет полученные события в сервис Sender по каналу;
7. Для получения событий сторонние приложения должны быть подписаны на сервис Sender через веб-сокеты. Если есть подписанное стороннее приложение, Sender отправляет ему события в соответствии с указанными фильтрами.

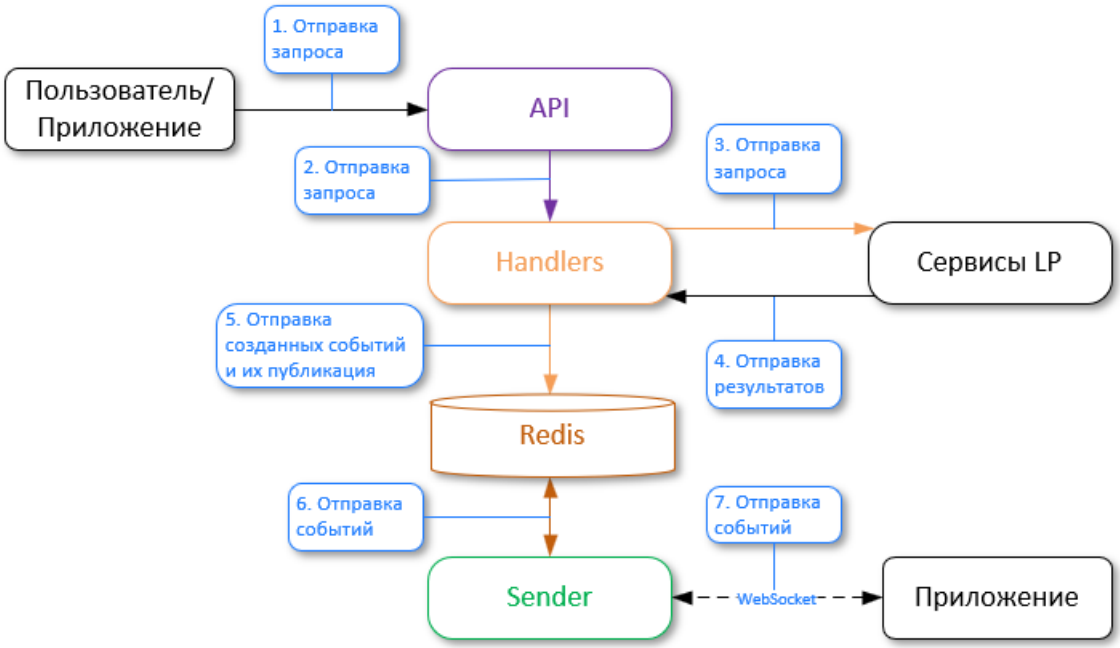


Рис. 33: Генерация событий и отправка в Sender

См. документацию OpenAPI для получения информации о структуре JSON, выдаваемой сервисом Sender.

6.11 Сервис Tasks

Сервис Tasks предназначен для выполнения длительных задач.

6.11.1 Общая информация о задачах

Обработка задач занимает значительное время, поэтому после создания задачи в ответе возвращается идентификатор задачи.

После завершения обработки задачи можно получить результаты задачи с помощью запроса «task» > «get task result». Необходимо указать идентификатор задачи, чтобы получить ее результаты.

Примеры результатов обработки задач можно найти в разделе ответа на запрос «task» > «get task result». Необходимо выбрать тип задачи в разделе **Response samples** документации.

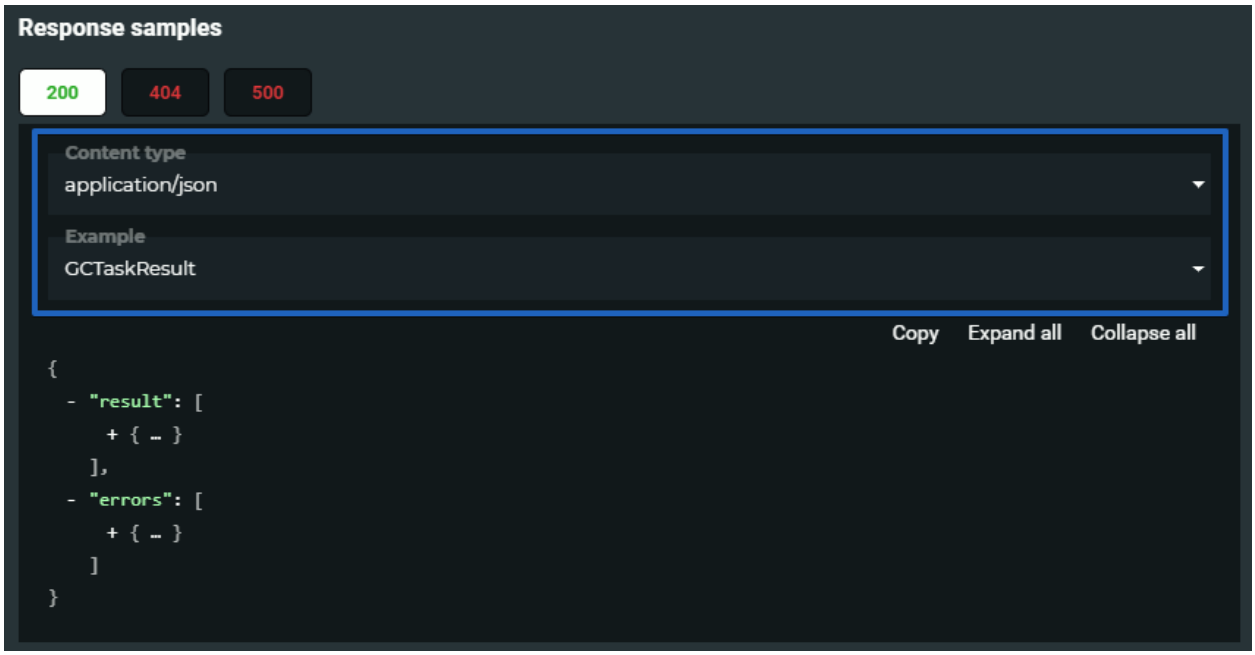


Рис. 34: Выбор необходимого примера

Перед отправкой запроса на получение результата необходимо убедиться в том, что задача выполнена:

- Можно проверить статус задачи, указав идентификатор задачи в запросе «tasks» > «get task». Существуют следующие статусы задач:

статус задач	значение
в ожидании (pending)	0

статус задач	значение
обрабатывается (in progress)	1
отменена (cancelled)	2
выполнена с ошибкой (failed)	3
сбор результатов (collect results)	4
выполнена (done)	5

- Можно получить информацию обо всех задачах с помощью запроса «tasks» > «get tasks». Можно установить фильтр, чтобы получать информацию только по интересующим задачам.

> Запросы на выполнение задач доступны для различных сервисов. В данном документе рассмотрены примеры создания задач с помощью сервиса API, Admin и пользовательского интерфейса Admin. Для подробной информации о создании задач с помощью прямых запросов к сервису Tasks см. в спецификации сервиса Tasks.

6.11.2 Задача Clustering

В результате выполнения задачи создается кластер с объектами, выбранными в соответствии с заданными фильтрами для лиц или событий. Объекты, соответствующие всем фильтрам, добавляются в кластер. Имеющиеся фильтры зависят от типа объекта: события или лица.

> Для получения статуса задачи или результатов её выполнения используются специальные запросы (см. «Общая информация о задачах»).

Можно использовать задачу создания отчета для получения отчета об объектах, добавленных в кластеры.

Кластеризация выполняется в несколько этапов:

- объекты с биометрическими шаблонами собираются в соответствии с предоставленными фильтрами
- каждый объект сопоставляется со всеми остальными объектами
- создаются кластеры в виде групп «связанных компонентов» из графа схожести.

  Здесь «связанные» означает, что схожесть превышает указанный порог или значение по умолчанию «DEFAULT_CLUSTERING_THRESHOLD» из настроек.
- при необходимости загружаются существующие изображения, соответствующие каждому объекту: аватар для лица, первый биометрический образец для события.

В результате выполнения задачи выдается массив кластеров. Кластер содержит идентификаторы объектов (лиц или событий), схожесть которых превышает заданный порог. Можно использовать информацию для дальнейшего анализа данных.


```
{
    "errors": [],
    "result": {
        "clusters": [
            [
                "6c721b90-f5a0-409a-ab70-bc339a70184c"
            ],
            [
                "8bc6e8df-410b-4065-b592-abc5f0432a1c"
            ],
            [
                "e4e3fc66-53b4-448c-9c88-f430c00cb7ea"
            ],
            [
                "02a3a1c4-93d7-4b69-99ec-21d5ef23852e",
                "144244cb-e10e-478c-bdac-18cd2eb27ee6",
                "1f4cdbcb-7b1e-40cc-873b-3ff7fa6a6cf0"
            ]
        ],
        "total_objects": 6,
        "total_clusters": 4
    }
}
```


> Результат задачи Clustering может также содержать информацию об ошибках, возникших при обработке объектов.

> Для такой задачи можно создать расписание.

6.11.3 Задача Reporter

В результате выполнения этой задачи создается отчет по задаче Clustering. Можно выбрать данные, которые необходимо добавить в отчет. Отчет предоставляется в формате CSV.

> Для получения статуса задачи или результатов её выполнения используются специальные запросы (см. «Общая информация о задачах»).

Можно указать идентификатор задачи Clustering и столбцы, которые необходимо добавить в отчет. Выбранные столбцы соответствуют основным полям событий и лиц.

> Необходимо убедиться в том, что выбранные столбцы соответствуют объектам, выбранным при выполнении задачи Clustering.

Также можно получить изображения для всех объектов в кластерах при их наличии.

6.11.4 Задача Exporter

С помощью этой задачи можно собирать данные о событиях и/или лицах и экспортировать их из LP в CSV-файл. В строке файла представлены запрошенные объекты и соответствующие биометрические образцы (если они были запрошены).

> При сборе данных с помощью этой задачи используется память. Поэтому, вполне возможно, что «рабочий процесс» Task будет завершён OOM (Out-Of-Memory) killer при запросе большого количества данных.

Экспортировать данные о событиях или лицах можно с помощью запроса «/tasks/exporter». Необходимо указать, какой тип объекта требуется, установив параметр `objects_type` при создании запроса. Также можно сузить количество данных для запроса, задав фильтры для лиц и событий. См. запрос «exporter task» в справочном руководстве сервиса API.

В результате выполнения задачи возвращается ZIP-архив, содержащий CSV-файл.

> Для получения статуса задачи или результатов её выполнения используются специальные запросы (см. «Общая информация о задачах»).

> При выполнении задачи Exporter с большим количеством лиц базе данных Faces (например, 90 000 000 лиц), время выполнения запросов к сервису Faces может быть значительно увеличено. Для ускорения выполнения запросов можно задать для настройки PostgreSQL «parallel_setup_cost» значение `500`. Однако следует учитывать, что изменение данной настройки может повлечь за собой другие последствия, поэтому следует внимательно отнестись к изменению настройки.

> Для такой задачи можно создать расписание.

6.11.5 Задача Cross-matching

При выполнении задачи выполняется сравнение всех эталонов со всем кандидатами. Кандидаты и эталоны задаются на основе фильтров для лиц и событий.

> Сравнение выполняется только для объектов, содержащих извлеченные биометрические шаблоны.

В поле `limit` можно указать максимальное количество кандидатов сравнения, выдаваемых для каждого совпадения.

Можно установить `threshold`, чтобы указать минимально допустимое значение схожести. Если схожесть двух БШ ниже указанного значения, результат сравнения будет проигнорирован и не будет выдаваться в ответе. Эталоны без совпадений с кандидатами также будут проигнорированы.

Сравнение выполняется в несколько этапов:

- На основе указанных фильтров подбираются объекты с БШ.
- Каждый объект-эталон сравнивается с каждым объектом-кандидатом.
- Результаты сравнения сортируются (лексикографически) и фильтруются (применяются `limit` и `threshold`).

> Можно получить информацию о статусе задачи или результатах с помощью дополнительных запросов (см. «Общая информация о задачах»).

В результате выполнения задачи возвращается массив. Каждый элемент массива содержит эталон и наиболее похожие кандидаты для него. Информация об ошибках, возникших при выполнении задачи, также выдается в ответе.


```
{
"result": [
    {
        "reference_id": "e99d42df-6859-4ab7-98d4-dafd18f47f30",
        "candidates": [
            {
                "candidate_id": "93de0ea1-0d21-4b67-8f3f-d871c159b740",
                "similarity": 0.548252
            },
            {
                "candidate_id": "54860fc6-c726-4521-9c7f-3fa354983e02",
                "similarity": 0.62344
            }
        ]
    },
    {
        "reference_id": "345af6e3-625b-4f09-a54c-3be4c834780d",
        "candidates": [
            {
                "candidate_id": "6ade1494-1138-49ac-bfd3-29e9f5027240",
                "similarity": 0.7123213
            },
            {
                "candidate_id": "e0e3c474-9099-4fad-ac61-d892cd6688bf",
                "similarity": 0.9543
            }
        ]
```

```
        }
    ],
    "errors": [
        {
            "error_id": 10,
            "task_id": 123,
            "subtask_id": 5,
            "error_code": 0,
            "description": "Faces not found",
            "detail": "One or more faces not found, including face with id '8
                f4f0070-c464-460b-bf78-fac225df72e9'",
            "additional_info": "8f4f0070-c464-460b-bf78-fac225df72e9",
            "error_time": "2018-08-11T09:11:41.674Z"
        }
    ]
    }
```


> Для такой задачи можно создать расписание.

6.11.6 Задача Linker

С помощью данной задачи можно прикреплять лица к спискам в соответствии с заданными фильтрами.

В запросе можно указать создание нового списка для привязки к нему или задать уже существующий список.

Для выполнения задачи можно задать фильтры для лиц или событий. Если для привязки к списку задано событие, новое лицо создается на основе этого события.

Если не указан фильтр `create_time_lt`, будет установлено текущее время.

В результате выполнения задачи выдаются идентификаторы лиц, привязанных к списку.

> Для получения статуса задачи или результатов её выполнения используются специальные запросы (см. «Общая информация о задачах»).

Процесс выполнения задачи Linker для лиц:

- Проверяется наличие списка с указанным `list_id` или создаётся новый список (если установлен параметр `create_list`, равный 1).
- Определяются границы идентификатора лица. Затем формируется одна или несколько подзадач примерно по 1000 идентификаторов лиц в каждой – в зависимости от распространения идентификаторов лиц.
- Для каждой подзадачи:

  - Определяются идентификаторы лиц, указанные для текущей подзадачи в соответствии с фильтрами в подзадаче.
  - Выполняется запрос в сервис Faces на привязку указанных лиц к указанному списку.
  - Результат каждой подзадачи сохраняется в сервисе Image Store.
- После завершения последней подзадачи, «рабочий процесс» собирает результаты всех подзадач, объединяет и помещает их в сервис Image Store (в виде результатов задачи).

Процесс выполнения задачи Linker для событий:

- Проверяется наличие списка с указанным `list_id` или создаётся новый список (если установлен параметр `create_list`, равный 1).
- Получение номера страниц с событиями. Затем формируется одна или несколько подзадач.
- Для каждой подзадачи:
  - С сервиса Events передается событие с его БШ.
  - В сервисе Faces создается лицо, к нему прикрепляются атрибуты и биометрические образцы.
  - Выполняется запрос в сервис Faces на привязку указанных лиц к указанному списку.
  - Результат каждой подзадачи сохраняется в сервисе Image Store.
- После завершения последней подзадачи, «рабочий процесс» собирает результаты всех подзадач, объединяет и помещает их в сервис Image Store (в виде результатов задачи).

> Для такой задачи можно создать расписание.

6.11.7 Задача Garbage collection

При обработке задачи могут быть удалены лица, события, обобщенные события или биометрические шаблоны.

- когда БШ заданы в качестве `target` для удаления, необходимо указать версию БШ для удаления. Все БШ указанной версии будут удалены.
- если события заданы в качестве `target` для удаления, необходимо указать один или несколько следующих параметров:
  - идентификатор учетной записи;
  - верхнее исключенное пороговое значение времени создания события;
  - верхнее исключенное пороговое значение появления события в видеопотоке;
  - идентификатор обработчика, использованного для создания события.
- если лица заданы в качестве `target` для удаления, необходимо указать один или несколько следующих параметров:
  - верхнее исключенное пороговое значение времени создания лица;
  - нижнее включенное пороговое значение времени создания лица;

- пользовательские данные.
- идентификатор списка;

> При необходимости можно удалить биометрические образцы, связанные с удаляемыми лицами или событиями. Для событий также можно удалить исходные изображения.

С помощью запроса «tasks» > «garbage collection task» сервиса API можно указать события (значение `events`), обобщенного события (значение `general_events`) или лица (значение `faces`) в качестве значений для поля `target`, тогда как в сервисах Admin или Tasks можно задавать в качестве `target` лица, события, обобщенные события и БШ (значение `descriptors`). В таком случае указанные объекты будут удалены для всех существующих аккаунтов.

> Для такой задачи можно создать расписание.

> Для получения статуса задачи или результатов её выполнения используются специальные запросы (см. «Общая информация о задачах»).

6.11.8 Задача Additional extraction

Задача Additional extraction повторно извлекает биометрические шаблоны, извлеченные с помощью предыдущей модели нейронной сети, с использованием новой версии нейронной сети. Это позволяет сохранить используемые ранее биометрические шаблоны при обновлении модели нейронной сети. Если нет необходимости в использовании старых БШ, то можно не выполнять данную задачу и просто обновить модель нейронной сети в настройках Configurator.

> В данном разделе описывается работа с задачей Additional extraction. См. подробную информацию о нейросетях, процессе обновления нейросети на новую модель и соответствующие примеры в разделе «Нейросети».

Повторное извлечение можно выполнить для объектов лиц и событий. Можно повторно извлечь БШ лиц, БШ тел (для событий) или базовые атрибуты, если они не были извлечены ранее.

Для повторного извлечения биометрических шаблонов с помощью новой нейросети необходимы биометрические образцы. БШ новой версии не будут извлечены для лиц и событий, у которых отсутствуют биометрические образцы.

> Крайне рекомендуется не выполнять никаких запросов, изменяющих состояние баз данных в процессе обновления версии БШ. Это может привести к потере данных.

> Создайте резервную копию баз данных LP и хранилища Image Store перед запуском задачи Additional extraction.

При обработке задачи извлекается БШ новой нейросети для каждого объекта (лица или события), чья версия биометрического шаблона совпадает с версией, указанной в настройках «DEFAULT_FACE_DESCRIPTOR_VERSION» (для лиц) или «DEFAULT_HUMAN_DESCRIPTOR_VERSION» (для тел).

Биометрические шаблоны, чья версия не совпадает с версией, указанной в данных настройках, повторно не извлекаются. Их можно удалить с помощью задачи Garbage collection.

Запрос к сервису Admin:

Необходимо выполнить запрос к ресурсу `/additional_extract`, указав следующие параметры в теле запроса:

- content > extraction_target – цель извлечения: БШ лица, БШ тела или базовые атрибуты
- content > options > descriptor_version – новая версия нейронной сети (не применимо для базовых атрибутов)
- content > filters > object_type – тип объекта: лица или события

> При необходимости можно дополнительно отфильтровать тип объекта по «account_id», ”face_id__lt”и пр.

> Для дополнительной информации см. запрос «create additional extract task» в спецификации OpenAPI сервиса Admin.

> Для получения статуса задачи или результата её выполнения используются специальные запросы (см. «Общая информация о задачах»).

Пользовательский интерфейс сервиса Admin:

Необходимо выполнить следующие действия:

- Открыть пользовательский интерфейс сервиса Admin: `http://<admin_server_ip>:5010/tasks`;
- Запустить задачу Additional extraction, нажав на соответствующую кнопку;
- В появившемся окне задать тип объекта (лица или события), цель извлечения (БШ лица, БШ тела или базовые атрибуты), новую модель нейросети (неприменимо для базовых атрибутов) и нажать «Start», подтвердив запуск задачи.

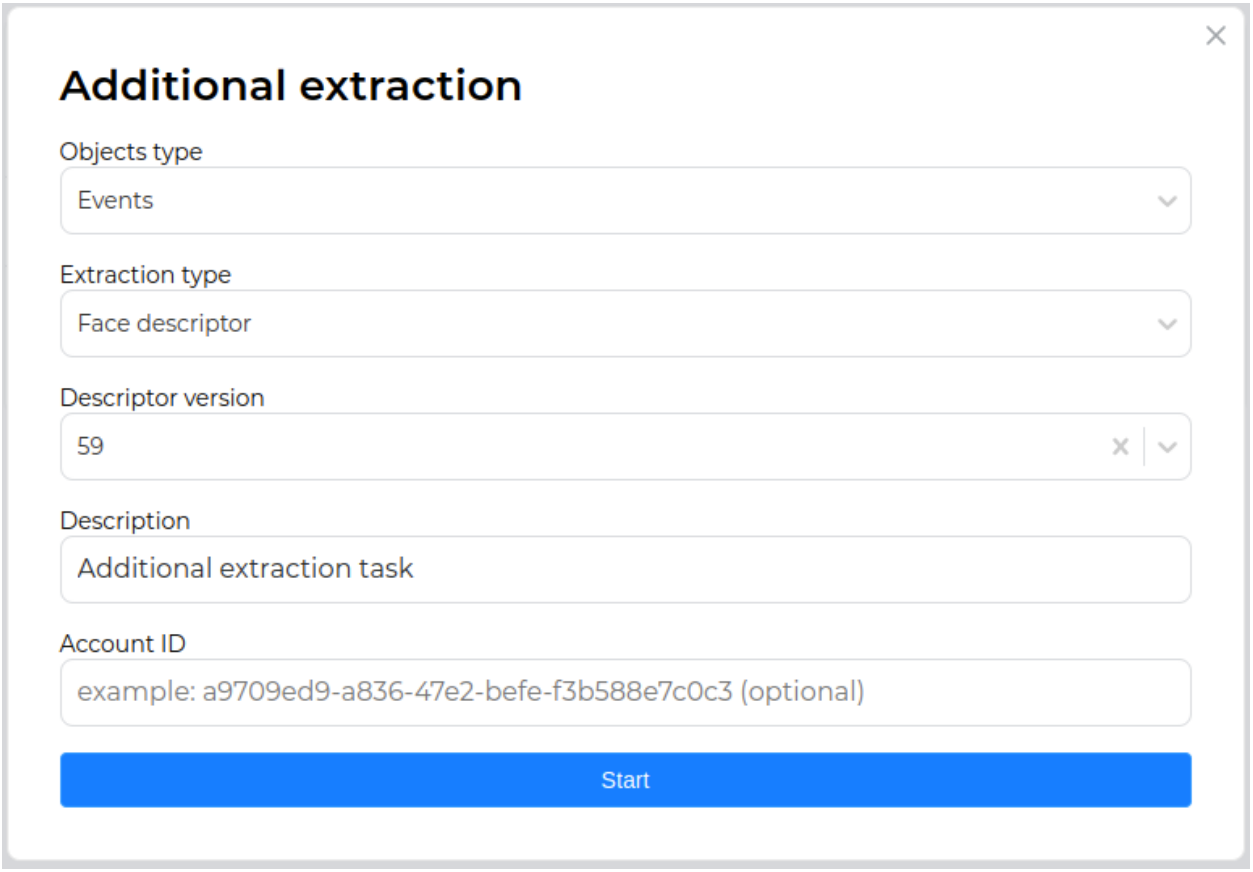


Рис. 35: Задание необходимых настроек

> При необходимости можно дополнительно отфильтровать тип объекта по «account_id».

> См. подробную информацию о пользовательском интерфейсе Admin в разделе «Пользовательский интерфейс сервиса Admin».

> Для такой задачи можно создать расписание.

6.11.9 Задача ROC-curve calculating

В результате выполнения этой задачи создается кривая рабочих характеристик приема с TPR (истинно положительной частотой) против FPR (ложно положительной частотой).

См. дополнительную информацию о построении ROC кривой в документе «TasksDevelopmentManual».

Задача расчета ROC

ROC (Receiver Operating Characteristic) – это измерение производительности для задач классификации при различных настройках пороговых значений. ROC-кривая строится в виде отношения TPR (True Positive Rate) к FPR (False Positive Rate). TPR – это количество пар истинно положительных

совпадений, деленное на общее предполагаемое количество пар положительных совпадений, а FPR – количество пар ложных положительных совпадений, деленное на общее предполагаемое количество пар отрицательных совпадений. Каждая точка (FPR, TPR) ROC-кривой соответствует определенному порогу схожести. См. дополнительную информацию в Wikipedia.

При использовании ROC производительность модели определяется следующим образом:

- область под ROC-кривой (или AUC);
- частота ошибок типа I и типа II равна точке, то есть точке пересечения ROC-кривой и вторичной главной диагонали.

Производительность модели также определяется попаданием в топ-N вероятности, то есть вероятность попадания положительной пары сравнения в топ-N для любой группы результатов сравнения, отсортированной по схожести.

Для выполнения задачи ROC требуется предварительно созданная разметка. При желании можно указать *threshold_hit_top* (по умолчанию 0) для расчета попадания в топ-N вероятность, сравнить *limit* (по умолчанию 5), *key_FPRs* – список ключевых значений FPR для расчета ключевых точек ROC-кривой и фильтры с *account_id*. Также для создания задачи нужен *account_id*.

> Для получения статуса задачи или результатов её выполнения используются специальные запросы (см. «Общая информация о задачах»).

Разметка

Разметка предполагается в следующем формате:

```
[{'face_id': <face_id>, 'label': <label>}]
```

Метка (или идентификатор группы) может быть числом или любой строкой.

Пример:

```
[{'face_id': '94ae2c69-277a-4e46-817d-543f7d3446e2', 'label': 0},
 {'face_id': 'cd6b52be-cdc1-40a8-938b-a97a1f77d196', 'label': 1},
 {'face_id': 'cb9bda07-8e95-4d71-98ee-5905a36ec74a', 'label': 2},
 {'face_id': '4e5e32bb-113d-4c22-ac7f-8f6b48736378', 'label': 3},
 {'face_id': 'c43c0c0f-1368-41c0-b51c-f78a96672900', 'label': 2}]
```

> Для такой задачи можно создать расписание.

6.11.10 Задача Estimator

Задача Estimator позволяет выполнять пакетную обработку изображений с использованием указанных политик.

В результате выполнения задачи возвращается JSON с данными для каждого из обработанных изображений и информацией о произошедших ошибках.

В теле запроса можно указать `handler_id` уже существующего статического или динамического обработчика. Для `handler_id` динамического обработчика доступна возможность задания требуемых политик. Кроме того, в запросе можно создать статический обработчик с указанием политик.

Ресурс может принимать в обработку пять типов источников с изображениями:

- ZIP-архив
- S3-подобное хранилище
- Сетевой диск
- FTP-сервер
- Сетевая файловая система Samba

Для получения корректных результатов обработки изображений с помощью задачи Estimator все обрабатываемые изображения должны быть либо в исходном формате, либо в формате биометрических образцов. Тип передаваемых изображений для всех источников указывается в теле запроса в параметре «image_type».

> Для такой задачи можно создать расписание. При создании расписания невозможно указать ZIP-архив в качестве источника изображений.

ZIP-архив как источник изображений для задачи Estimator

Размер архива задаётся с помощью параметра «ARCHIVE_MAX_SIZE» в конфигурационном файле «config.py» сервиса Tasks. По умолчанию размер равен 100 Гб. В качестве ссылки на архив можно использовать внешний URL-адрес или URL-адрес архива, сохраненного в Image Store. Во втором случае архив сначала следует сохранить в LP с помощью запроса POST к ресурсу «/objects».

При использовании внешнего URL-адреса, ZIP-архив сначала скачивается в хранилище контейнера Tasks Worker, где происходит распаковка и обработка изображений. После окончания работы задачи архив вместе с распакованными изображениями удаляются из хранилища.

> Необходимо учитывать наличие свободного места для вышеописанных действий.

Передаваемый архив может быть защищён паролем. Пароль можно передать в запросе с помощью параметра «authorization» > «password».

S3-подобное хранилище как источник изображений для задачи Estimator

Для такого типа источника доступно задание следующих параметров:

- Параметр «bucket_name» — имя бакета/Access Point ARN/Outpost ARN (обязательное действие);
- Параметр «endpoint» — endpoint (только при указании имени бакета);
- Параметр «region» — bucket region (только при указании имени бакета);

- Параметр «prefix» — префикс ключа файла. Также может использоваться для загрузки изображений из определенной папки, например, «2022/January».

Для настройки авторизации предназначены следующие параметры:

- Public access key (обязательное действие);
- Secret access key (обязательное действие);
- Signature version («s3v2»/«s3v4»).

Также доступна возможность рекурсивного выкачивания изображений из вложенных папок бакета (параметр «recursive») и сохранения исходных изображений (параметр «save_origin»).

Дополнительную информацию о работе с S3-подобными хранилищами см. в руководстве пользователя AWS.

Сетевой диск как источник изображений для задачи Estimator

Для такого типа источника доступно задание следующих параметров:

- Параметр «path» — абсолютный путь к директории с изображениями в контейнере (обязательное поле);
- Параметр «follow_links» — включение/выключение обработки символических ссылок (symlink);
- Параметр «prefix» — префикс ключа файла. Может использоваться для загрузки изображений из определенной директории;
- Параметр «postfix» — постфикс ключа файла. Может использоваться для загрузки изображений с определенным расширением.

См. пример использования префиксов и постфиксов в описании ресурса «/tasks/estimator».

При использовании такого типа источника и запуске сервисов Tasks и Tasks Worker через Docker-контейнеры необходимо смонтировать директорию с изображениями с сетевого диска в локальную директорию и синхронизировать её с указанной директорией в контейнере. Смонтировать директорию с сетевого диска можно любым удобным способом. После этого можно синхронизировать смонтированную директорию с директорией в контейнере с помощью следующей команды при запуске сервисов Tasks и Tasks Worker:

```
docker run \
...
-v /var/lib/luna/current/images:/srv/images
...
```

/var/lib/luna/current/images — путь к предварительно смонтированной директории с изображениями с сетевого диска.

/srv/images — путь к директории с изображениями в контейнере, куда они будут перенесены с сетевого диска. Этот путь должен быть указан в теле запроса задачи Estimator (параметр «path»).

Как и в задаче Estimator с использованием S3-подобного хранилища в качестве источника изображений, доступна возможность рекурсивного получения изображений из вложенных директорий бакета и выбора типа передаваемых изображений.

FTP-сервер как источник изображений для задачи Estimator

Для такого типа источника в теле запроса доступно задание следующих параметров для соединения с FTP-сервером:

- Параметр «host» — IP-адрес или имя хоста FTP-сервера (обязательное поле);
- Параметр «port» — порт FTP-сервера;
- Параметр «max_sessions» — максимальное количество разрешенных сеансов на FTP-сервере;
- Параметры «user» и «password» — параметры авторизации (обязательные поля).

Как и в задачах Estimator с использованием S3-подобного хранилища или сетевого диска в качестве источников изображений, доступна возможность задания пути до директории с изображениями, рекурсивного получения изображений из вложенных директорий, выбора типа передаваемых изображений, а также указания префикса и постфикса.

См. пример использования префиксов и постфиксов в описании ресурса «/tasks/estimator».

Samba как источник изображений для задачи Estimator

Для такого типа источника параметры аналогичны параметрам FTP-сервера, за исключением параметра «max_sessions». Также если не указываются данные авторизации, подключение к Samba будет осуществляться как гостевое.

6.11.11 Обработка задачи

Сервис Tasks включает в себя «рабочие процессы» Tasks. Сервис Tasks получает запросы, создает задачи в БД и отправляет подзадачи «рабочим процессам» Tasks. «Рабочие процессы» сервиса Tasks реализованы в виде отдельного контейнера Tasks Worker. «Рабочие процессы» Tasks получают подзадачи и выполняют все необходимые запросы к другим сервисам для решения подзадач.

Ниже представлен общий подход к работе с задачами.

- Сервис API получает запрос на создание новой задачи;
- Сервис Tasks создает новую задачу и отправляет подзадачи «рабочим процессам»;
- «Рабочие процессы» Tasks обрабатывают подзадачи и создают отчеты;
- Если несколько «рабочих процессов» обработали подзадачи и создали несколько отчетов, «рабочий процесс», выполнивший последнюю подзадачу, собирает все отчеты и создает единый отчет;
- После завершения выполнения задачи последний «рабочий процесс» обновляет свой статус в базе данных Tasks;

- Пользователь может отправлять запросы на получение информации о задачах и подзадачах и количестве активных задач. Пользователь может отменять или удалять задачи;
- Пользователь может получать информацию об ошибках, возникших при выполнении задач;
- После завершения выполнения задачи пользователь может отправить запрос на получение результатов задачи.

См. раздел «Диаграммы задач» для получения подробной информации об обработке задач.

6.11.12 Запуск задач по расписанию

В LUNA PLATFORM доступна возможность задать расписание выполнения задач Garbage collection, Clusterization, Exporter, Linker, Estimator, Additional extract, Cross-matching и Roc-curve calculating. С помощью расписания можно гибко управлять временем начала выполнения задач. Например, можно настроить регулярное расписание для очистки событий старше одного месяца каждую пятницу в ночное время.

> Для использования фильтра относительно текущего времени («now-time»), текущее время будет отсчитываться не от создания расписания, а от создания задачи расписанием в соответствии с cron-выражением. См. подробную информацию в разделе «Фильтры now-time».

Расписание создается с помощью запроса «create tasks schedule» к сервису API, в котором указывается содержимое создаваемой задачи и временной интервал её запуска. Для указания временного интервала используются Cron-выражения.

> Cron-выражения используются для определения расписания выполнения задач. Они состоят из пяти полей, разделенных пробелами. Каждое поле определяет определенный временной интервал, в котором задача должна быть выполнена. Номер недели начинается с воскресенья.

> Тем задачам, которые можно выполнить только через сервис Admin (например, задача удаления некоторых объектов через GC), можно назначить расписание только в сервисе Admin.

В ответ на запрос выдается уникальный идентификатор «schedule_id», по которому можно получить информацию о статусе задачи, время выполнения следующей задачи и пр. (запросы ”get tasks schedule и ”get tasks schedules). Идентификатор и вся дополнительная информация сохраняются в таблицу «schedule» БД Tasks.

При необходимости можно создать отложенное расписание, а затем активировать его с помощью параметра «action» = «start» запроса «patch tasks schedule». Аналогично можно остановить работу задачи по расписанию с помощью «action» = «stop». Для удаления расписания можно воспользоваться запросом «delete tasks schedule».

Права на работу с расписаниями задаются в токене разрешением «task». Это означает, что если у пользователя есть разрешение на работу с задачами, то он также сможет воспользоваться расписанием.

> Возможность создания расписания также доступна для задач Lambda.

6.11.12.1 Примеры Cron-выражений

В данном разделе описаны различные примеры Cron-выражений.

1. Запускать задачу каждый день в 3 часа ночи:

```
0 3 * * *
```

2. Запускать задачу каждую пятницу в 18:30:

```
30 18 * * 5
```

3. Запускать задачу каждый первый день месяца в полдень:

```
0 12 1 * *
```

4. Запускать задачу каждые 15 минут:

```
*/15 * * * *
```

5. Запускать задачу каждое утро в 8:00, кроме выходных (суббота и воскресенье):

```
0 8 * * 1-5
```

6. Запускать задачу в 9:00 утра в первый и 15-й день каждого месяца, но только если это понедельник:

```
0 9 1,15 * 1
```

6.11.13 Отправка уведомлений об изменении статуса задач и подзадач

При необходимости можно отправлять уведомления об изменении статуса задач и подзадач с помощью механизма callback'ов. Callback'и позволяют отправлять данные в стороннюю систему по указанному URL или в Telegram. Для настройки уведомлений необходимо настроить политику «notification_policy» в параметрах запроса соответствущей задачи.

Также можно настроить отправку уведомлений для задач и подзадач в настройках расписания.

При необходимости можно получить информацию о текущем состоянии политики уведомления или изменить какие-то данные политики с помощью запросов «get task notification policy» и «replace task notification policy».

6.11.14 Дополнительная защита паролей и токенов

Пароли и токены, передаваемые в задаче Estimator и в политике «notification_policy», могут быть дополнительно зашифрованы. Для этого необходимо при старте контейнера сервиса Tasks передать пользовательские значения переменным окружения FERNET_PASSPHRASE и SALT.

FERNET_PASSPHRASE — это пароль или ключ, используемый для шифрования данных с помощью алгоритма Fernet.

SALT — это случайная строка, добавляемая к паролю перед его хешированием.

> Fernet — это симметричный алгоритм шифрования, который обеспечивает аутентификацию и целостность данных, а также конфиденциальность. При использовании этого алгоритма один и тот же ключ используется для шифрования и дешифрования данных.

> Salt добавляется для усложнения процесса взлома пароля методом подбора. Каждый раз, когда пароль хешируется, используется уникальная строка, что делает одинаковые пароли хешированными по-разному. Это повышает безопасность системы, особенно в случае, если у пользователей есть одинаковые пароли.

Пример команды запуска контейнера с передачей переменных окружения:

```
docker run \
--env=CONFIGURATOR_HOST=127.0.0.1 \
--env=FERNET_PASSPHRASE=security_passphrase \
--env=SALT=salt_for_passwords_and_tokens \
...
```

Важно! Когда контейнер запускается с указанием вышеописанных переменных окружений, то старые пароли и токены перестают работать. Необходимо выполнить дополнительные действия по миграции (см. раздел ниже).

6.11.14.1 Добавление шифрования при обновлении

Для того, чтобы добавить дополнительную защиту для уже существующих паролей и токенов, необходимо в команде миграции БД Tasks указать переменные окружения (см. выше). После чего необходимо запустить новый контейнер Tasks с указанием переменных окружения для возможности использования шифрования при создании новых объектов.

6.12 Сервис Admin

Сервис Admin используется для выполнения общих административных процедур:

- Управление учетными записями пользователей;
- Получение информации об объектах, принадлежащих разным учетным записям;
- Создание задач Garbage collection;
- Создание задач Additional extraction;
- Получение отчетов и ошибок по обработанным задачам;
- Отмена и удаление существующих задач.

Сервис Admin имеет доступ ко всем данным, привязанным к разным учетным записям.

В сервисе Admin можно создать аккаунт трех типов — «user», «advanced_user» и «admin». Первые два типа создаются с помощью запроса на создание аккаунта к сервису API, однако третий тип можно создать только с помощью сервиса Admin.

С помощью типа аккаунта «admin» можно авторизироваться в пользовательском интерфейсе (см. ниже) и выполнять вышеописанные задачи. Аккаунт с типом «admin» можно создать как в пользовательском интерфейсе, так и с помощью запроса к ресурсу `/accounts` сервиса Admin. Для создания аккаунта последним способом требуется указать логин и пароль.

> Если аккаунт создается впервые, то необходимо использовать логин и пароль по умолчанию.

Пример CURL-запроса к ресурсу `/accounts` сервиса Admin:

```
curl --location --request POST 'http://127.0.0.1:5010/4/accounts' \
--header 'Authorization: Basic cm9vdEB2aXNpb25sYWJzLmFpOnJvb3Q=' \
--header 'Content-Type: application/json' \
--data '{
  "login": "mylogin@gmail.com",
  "password": "password",
  "account_type": "admin",
  "description": "description"
}'
```

Все запросы к сервису Admin описаны спецификации OpenAPI сервиса Admin.

6.12.1 Пользовательский интерфейс сервиса Admin

Пользовательский интерфейс сервиса Admin предназначен для упрощения работы с административными задачами.

Интерфейс можно открыть в браузере, указав адрес и порт сервиса Admin: `<Admin_server_address>:<Admin_server_port>`.

> Порт сервиса Admin по умолчанию — 5010.

Логин и пароль по умолчанию для доступа к интерфейсу — `root@visionlabs.ai/root`. Также можно использовать логин и пароль по умолчанию в формате Base64 при запросе к ресурсу `/accounts` (см. ниже) — `cm9vdEB2aXNpb25sYWJzLmFpOnJvb3Q=`.

> Можно изменить пароль по умолчанию для сервиса Admin с помощью запроса «Change authorization».

На странице доступно три вкладки:

- Accounts. Вкладка предназначена для предоставления информации о всех созданных аккаунтах и для создания новых аккаунтов.
- Tasks. Вкладка предназначена для работы с задачами Garbage Collection и Additional extraction.
- Info. Вкладка содержит информацию о пользовательском интерфейсе и лицензии LUNA PLATFORM.

6.12.1.1 Вкладка Accounts

На данной вкладке отображаются все существующие аккаунты.

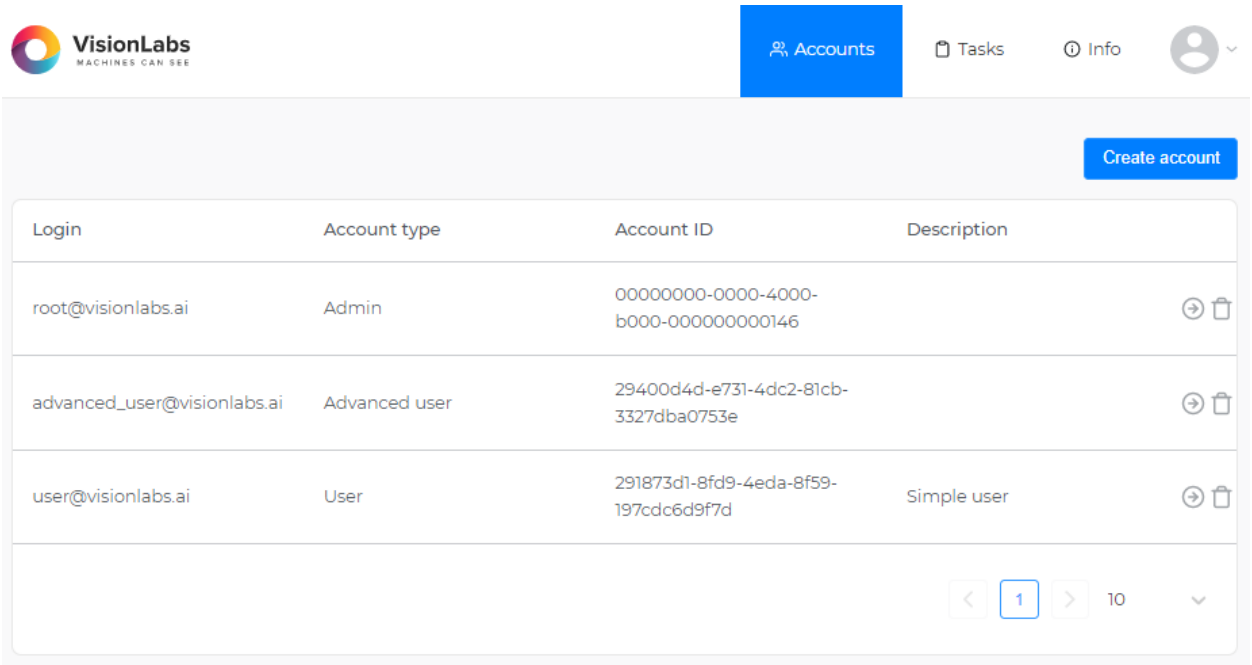


Рис. 36: Вкладка Accounts

Можно управлять существующими аккаунтами с помощью следующих кнопок:

– просмотр информации об аккаунте.

– удаление аккаунта.

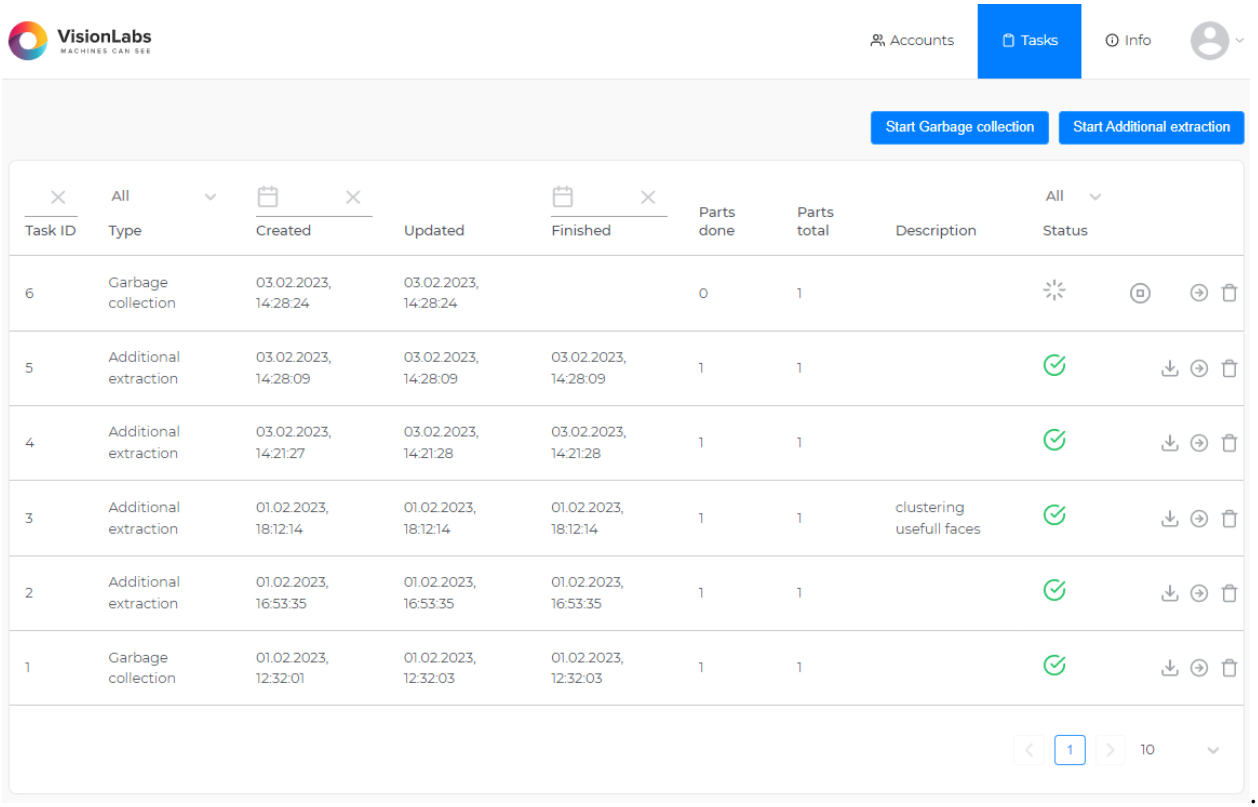
При нажатии кнопки просмотра информации открывается страница, содержащая общую информацию об аккаунте, списках, созданных с помощью данного аккаунта, и лицах.

При нажатии кнопки «Create account» открывается окно создания аккаунта, содержащее стандартные настройки создания аккаунта — логин, пароль, тип аккаунта, описание и желаемый «account_id».

> См. подробную информацию об аккаунтах и их типах в разделе «Аккаунт».

6.12.1.2 Вкладка Tasks

На данной вкладке отображаются выполняемые/выполненные задачи Garbage collection и Additional extraction.



Задачи отображаются в таблице, столбцы которой можно сортировать, а также фильтровать по дате выполнения задач.

При нажатии кнопок «Start Garbage collection» и «Start Additional extraction» открываются окна создания соответствующих задач.

Окно «Garbage collection» содержит следующие настройки, аналогичные параметрам тела запроса «garbage collecting task» к сервису Tasks:

- Description — параметр «description»

- Target — параметр «content > target»
- Account ID — параметр «content > filters > account_id»
- Remove sample — параметр «content > remove_samples»
- Remove image origins — параметр «content > remove_image_origins»
- Delete data before — параметр ”content > create_time__lt”

См. подробную информацию в разделе «Задача Garbage collection».

Окно «Additional extraction» содержит следующие настройки, аналогичные параметрам тела запроса «additional extract task» к сервису Tasks:

- Objects type — параметр «content > filters > object_type»
- Extraction type — параметр «content > extraction_target»
- Descriptor version — параметр «content > options > descriptor_version»
- Description — параметр «description»
- Account ID — параметр «content > filters > account_id»

См. подробную информацию в разделе «Задача Additional extraction».

После создания задачи, начинается её выполнение. Процесс выполнения задачи отображается иконкой  . Задача считается выполненной когда значение «Parts done» совпадает со значением «Parts total» и иконка меняется на  . При необходимости можно остановить выполнение задачи с помощью иконки  .

Для каждой задачи доступны следующие кнопки:

 – скачивание результата выполнения задачи в виде файла формата JSON.

 – переход на страницу подробного описания задачи и ошибок, полученных при её выполнении.

 – удаление задачи.

> Задачи выполняются сервисом Tasks после получения запроса от сервиса Admin.

6.12.1.3 Вкладка Schedules

Данная вкладка предназначена для работы с расписанием задач.

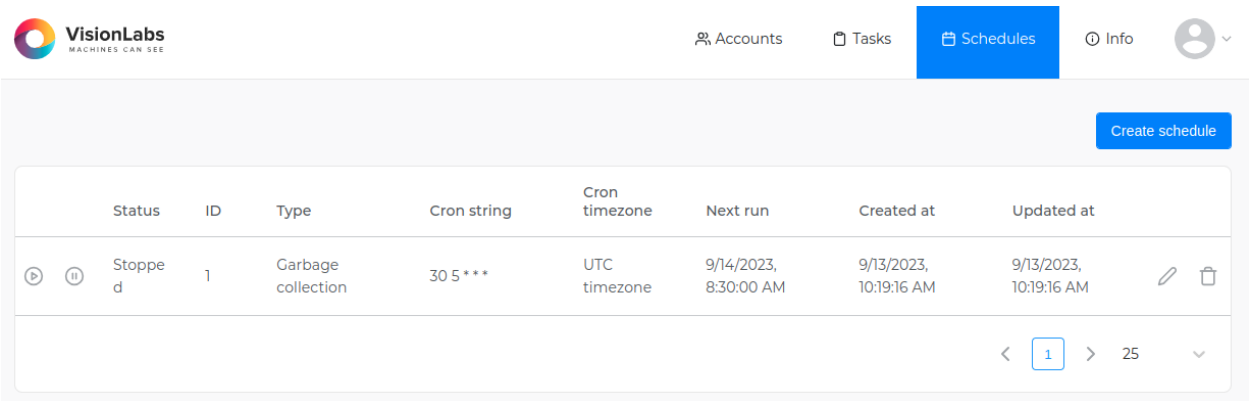


Рис. 37: Вкладка Schedules

Во вкладке отображаются все созданные расписания задач и вся соответствующая информация (статус, идентификатор, Cron-строка и т.д.).

При нажатии на кнопку «Create schedule» открывается окно создания расписания.

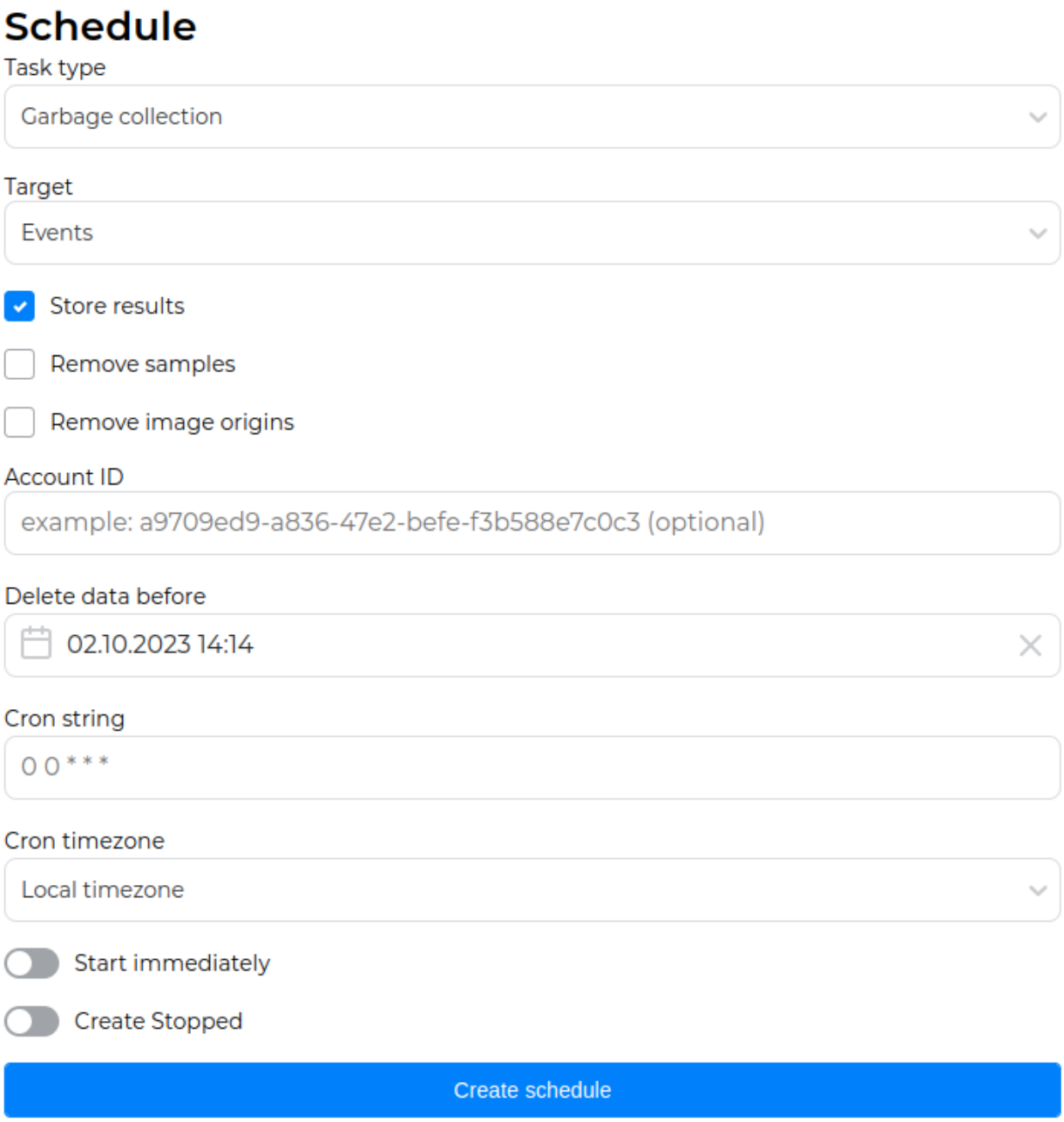


Рис. 38: Окно создания расписания

В окне можно задать настройки расписания для задачи Garbage collection. Параметры в данном окне соответствуют параметрам запроса «create tasks schedule».

После заполнения параметров и нажатия кнопки «Create schedule» расписание появится во вкладке Schedules.

Можно управлять отложенным запуском с помощью следующих кнопок:

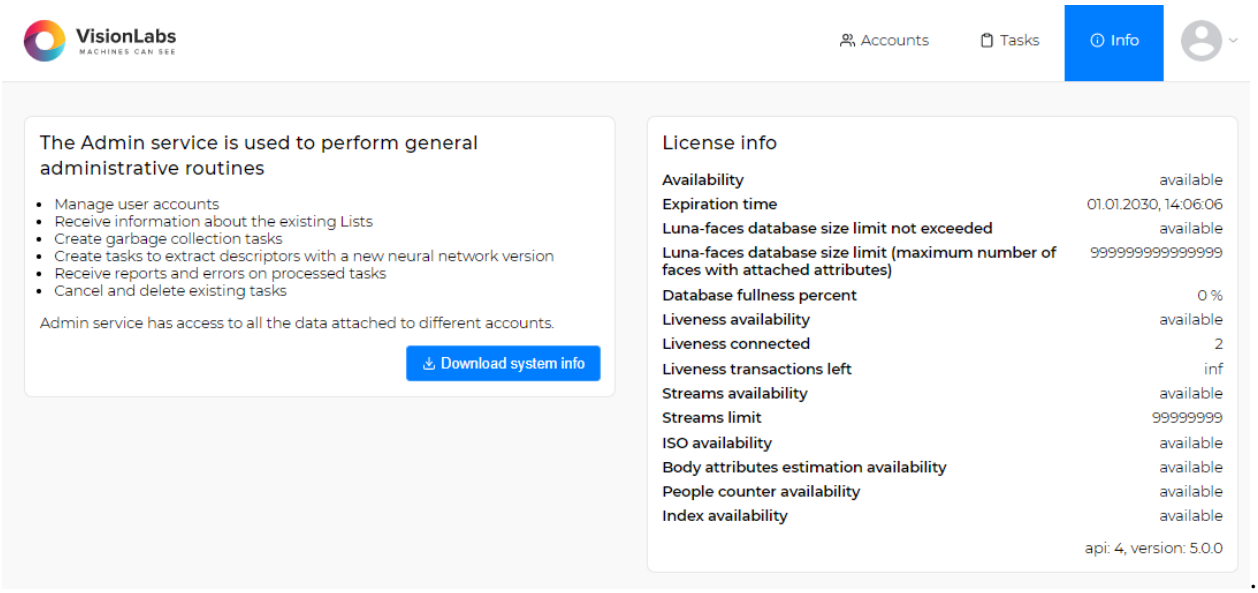
– запуск расписания.

– остановка расписания.

С помощью кнопки можно редактировать расписание. С помощью кнопки можно удалить расписание.

6.12.1.4 Вкладка Info

На данной вкладке отображается полная информация о лицензии и возможности, которые можно выполнить с помощью пользовательского интерфейса Admin.



> См. подробное описание лицензии в разделе «Информация о лицензии».

Нажав на кнопку «Download system info» можно также получить следующую техническую информацию о LP:

- версия LUNA PLATFORM,
- версии сервисов LUNA PLATFORM,
- количество биометрических шаблонов,
- значения конфигурационных файлов,
- информация о лицензии,
- статистика выполненных запросов и оценок (см. «Подсчет статистики выполненных запросов и оценок»).

Эта информация необходима для технической поддержки. При отправке вопроса необходимо приложить к письму полученный файл JSON.

> Получить вышеперечисленную системную информацию также можно с помощью запроса «get system info» к сервису Admin.

6.13 Сервис Configurator

С помощью сервиса Configurator упрощается настройка сервисов LP.

Сервис сохраняет все необходимые конфигурации для всех сервисов LP в одном месте. Можно редактировать конфигурации с использованием пользовательского интерфейса или специальных файлов с ограничениями.

Также можно сохранить в Configurator конфигурации для любого стороннего программного обеспечения.

Общий процесс работы выглядит следующим образом:

- Пользователь редактирует конфигурации в пользовательском интерфейсе;
- Сервис Configurator сохраняет все измененные конфигурации и другие данные в базе данных;
- Сервисы LP запрашивают сервис Configurator во время запуска и получают все необходимые конфигурации. Все сервисы должны быть настроены на использование сервиса Configurator.

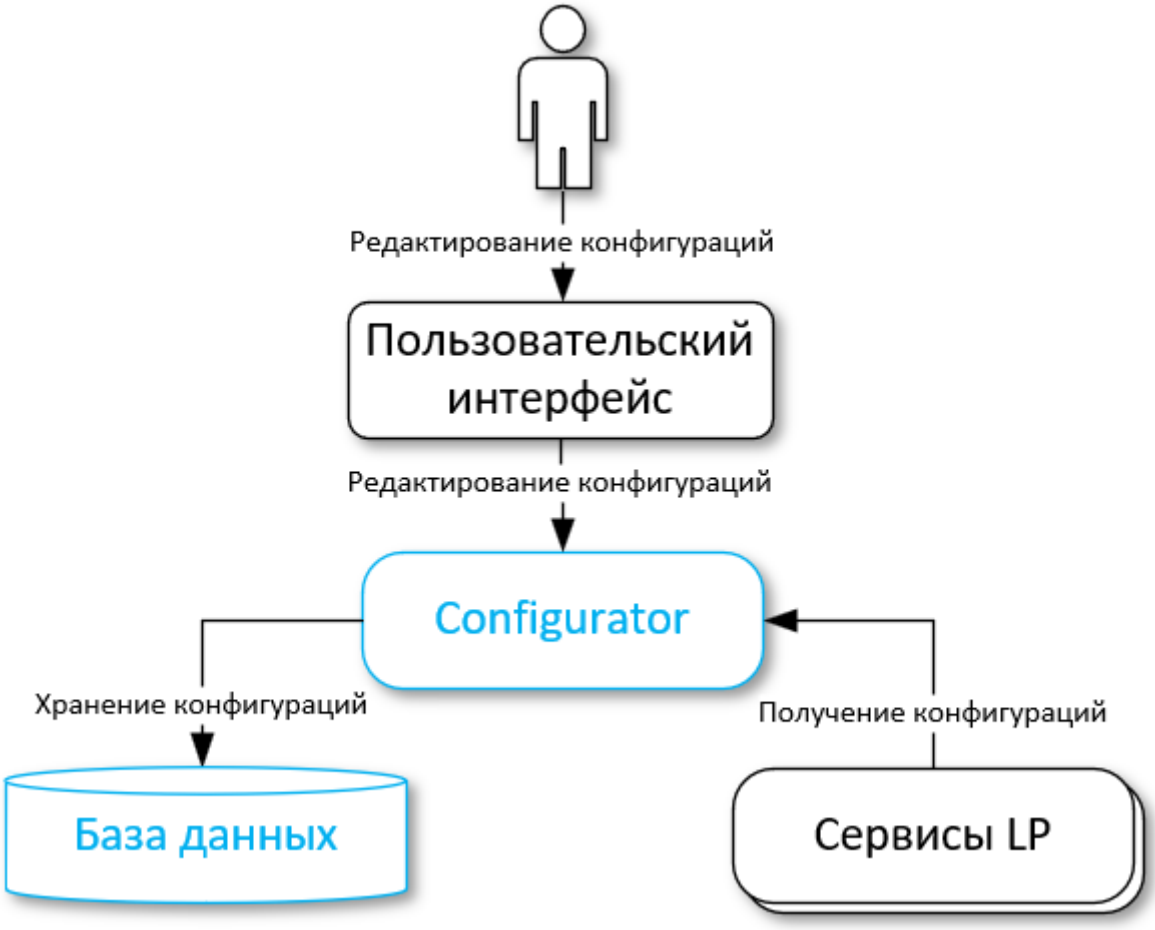


Рис. 39: Работа сервиса Configurator

При установке сервиса Configurator также можно использовать свой файл с ограничениями со всеми обязательными полями для создания ограничений и заполнения базы данных Configurator. Более подробную информацию об этом процессе можно найти в «руководстве разработчика».

Настройки, используемые несколькими сервисами, обновляются для каждого из них. Например, при изменении параметра «LUNA_FACES_ADDRESS» для сервиса Handlers в пользовательском интерфейсе Configurator, этот параметр также будет обновлен для сервисов API, Admin и Python Matcher.

6.13.1 Пользовательский интерфейс сервиса Configurator

Пользовательский интерфейс сервиса Configurator доступен по следующему адресу: `<Configurator_server> :5070`.

URL может отличаться. В этом примере интерфейс сервиса Configurator открывается на сервере сервиса Configurator.

> LP содержит бета-версию пользовательского интерфейса сервиса Configurator. Пользовательский интерфейс тестировался в браузерах Chrome и Yandex. Рекомендуемое разрешение экрана для работы с пользовательским интерфейсом – 1920 x 1080.

В пользовательском интерфейсе Configurator доступны следующие вкладки:

- **Settings**. Все данные в сервисе Configurator хранятся во вкладке **Settings**. На этой вкладке отображаются все имеющиеся настройки. Также можно управлять ими и фильтровать;
- **Limitations**. Эта вкладка используется для создания новых ограничений для настроек. Ограничениями являются шаблоны для файлов JSON, содержащие тип имеющихся данных и другие правила для определения параметров;
- **Groups**. С помощью этой вкладки можно сгруппировать все необходимые настройки. При выборе группы на вкладке **Settings**, будут отображаться только настройки, соответствующие группе. Можно получить настройки в соответствии с фильтрами и/или тегами для одного определенного сервиса.
- **About**. В этой вкладке содержится информация об интерфейсе сервиса Configurator.

6.13.1.1 Настройки

Каждая настройка сервиса Configurator содержит следующие поля:

- Name – название настройки.
- Description – описание настройки;
- ID and Times – уникальный идентификатор настройки;
- Create time – время создания настройки;
- Last update time – время последнего обновления настройки;
- Value – тело настройки;

- Schema – шаблон проверки тела схемы;
- Tag – теги для настройки, используемые для фильтрации настроек для сервисов.

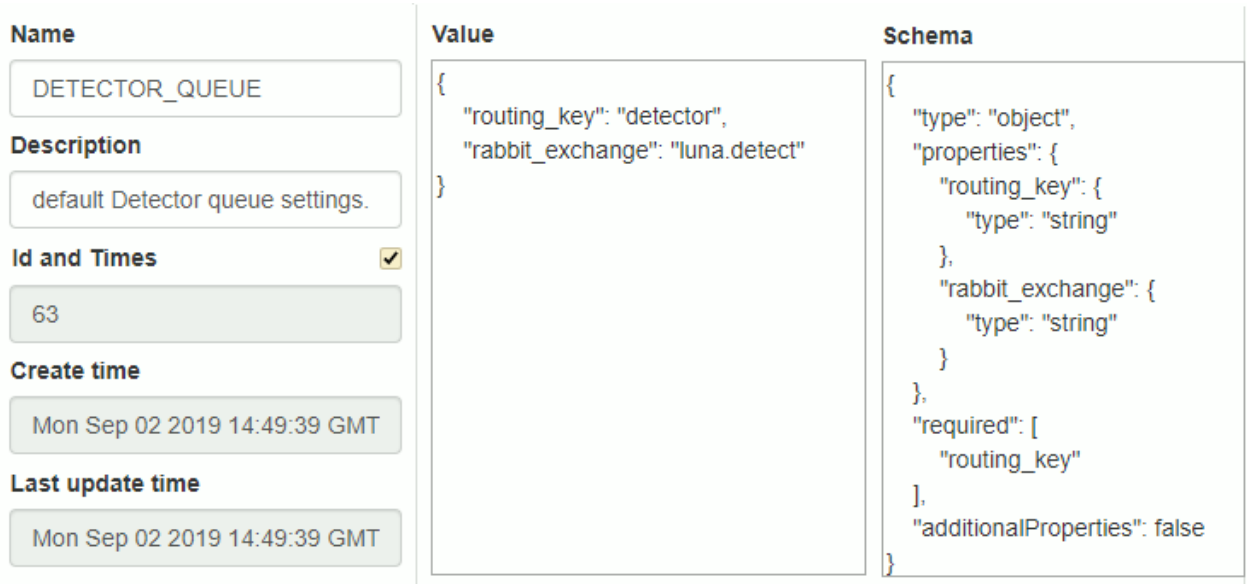


Рис. 40: Интерфейс сервиса Configurator

В поле «Tag» отсутствуют настройки по умолчанию. Необходимо нажать кнопку **Duplicate** и создать новую настройку на основе имеющейся.

Доступны следующие варианты настроек:

- Создание новой настройки – нажать кнопку **Create new**, ввести необходимые значения и нажать **Create**. Также необходимо выбрать уже имеющееся существующее ограничение для настройки. Сервис Configurator попытается проверить значение параметра, если будет стоять флажок напротив **Check on save** и для параметра будет выбрано ограничение;
- Дублирование имеющейся настройки – нажать кнопку **Duplicate** справа от настройки, изменить требуемые значения и нажать **Create**. Сервис Configurator попытается проверить значение настройки, если в левой нижней части экрана будет установлен флажок напротив **Check on save** и будет такая возможность;

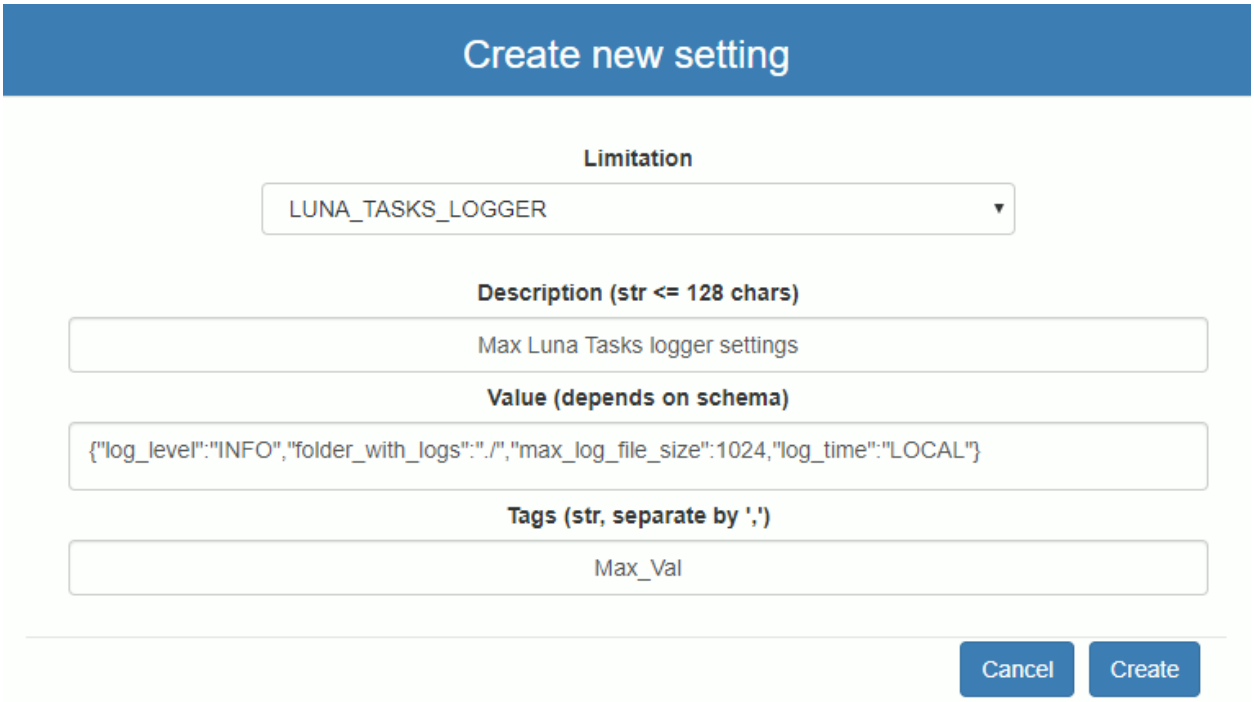


Рис. 41: Окно настройки Dublicate

- Удаление имеющейся настройки – нажать кнопку **Delete** справа от настройки.
- Обновление имеющейся настройки – нужно изменить имя, описание, теги, значение и нажать кнопку **Save** справа от настройки.
- Фильтрование имеющихся настроек по имени, описанию, тегам, именам сервисов, группам – нужно указать фильтры в левой части экрана и нажать **Enter** или нажать кнопку **Search**;

Show limitations – флаг используется для включения отображения ограничений для каждой настройки.

JSON editors – с помощью флага можно переключать режим отображения поля значений. При отсутствии флага отображается наименование параметра и поле его значения. При наличии флага поле Value отображается в виде JSON.

В разделе **Filters** в левой части окна могут отображаться все необходимые настройки в соответствии с указанными значениями. Нужное наименование можно ввести вручную или выбрать из списка:

- **Setting.** При использовании данного фильтра будет отображаться определенная настройка с указанным именем.
- **Description.** При использовании данного фильтра будут отображаться все настройки с указанным описанием или частью описания.
- **Tags.** При использовании данного фильтра будут отображаться все настройки с указанным тегом.

- **Service filter.** При использовании данного фильтра будут отображаться все настройки, относящиеся к выбранному сервису.
- **Group.** При использовании данного фильтра будут отображаться все настройки, принадлежащие указанной группе. Например, можно выбрать, чтобы отображались все сервисы, принадлежащие LP.

6.13.1.2 Использование тегированных настроек

С помощью тегированных настроек можно запустить несколько одинаковых сервисов, которые будут использовать различные настройки из Configurator.

Для этого необходимо выполнить следующие действия:

1. Дублировать или создать новую настройку, указав ей тег. Например, можно дублировать настройку «LUNA_EVENTS_DB» и присвоить ей тег «EVENTS_DB_TAG».
2. Передать следующие аргументы в команду «run.py» соответствующего контейнера:

- `--luna-config` — флаг, содержащий адрес сервиса Configurator
- `--<configuration_name>` — флаг, содержащий настройку и тег

> См. дополнительную информацию по аргументам в разделе «Аргументы сервисов» руководства по установке.

Например, для настройки «LUNA_EVENTS_DB» с тегом «EVENTS_DB_TAG» команда запуска контейнера будет выглядеть следующим образом:

```
docker run \
...
dockerhub.visionlabs.ru/luna/luna-events:v.4.19.3
python3 /srv/luna_events/run.py --luna-config http://127.0.0.1:5070/1 --
    LUNA_EVENTS_DB EVENTS_DB_TAG
```

6.13.1.3 Ограничения

Ограничения используются в качестве схемы проверки параметров сервиса.

Настройки и ограничения имеют одинаковые наименования. При создании ограничения создается новая настройка.

> Ограничения установлены по умолчанию для каждого сервиса LP. Их нельзя изменить.

Каждое ограничение содержит следующие поля:

- **Name** – наименование ограничения.
- **Description** – описание ограничения.

- **Service list** – список сервисов, в которых могут использоваться настройки этого ограничения.
- **Schema** – это объект со схемой JSON для проверки настроек.
- **Default value** – это значение по умолчанию, установленное при создании ограничения.

Для управления ограничениями можно выполнить следующие действия:

- Создание нового ограничения – нажать кнопку **Create new**, ввести необходимые значения и нажать **Create**. Также будет создана настройка со значением по умолчанию;
- Дублирование существующего ограничения – нажать кнопку **Duplicate** справа от ограничения, изменить требуемые значения и нажать **Create**. Также будет создана настройка со значением по умолчанию;
- Обновление значения ограничения – изменить name/description/service list/validation schema/default values и нажать кнопку **Save** справа от ограничения;
- Фильтрование существующего ограничения по именам, описаниям и группам;
- Удаление существующего ограничения – нажать кнопку **Delete** справа от ограничения.

6.13.1.4 Группы

У группы есть наименование и описание.

Можно:

- Создать новую группу – нажать кнопку **Create new**, ввести имя группы и, при необходимости, описание и нажать **Create**;
- Отфильтровать существующие группы по именам групп и/или именам ограничений – задать фильтры с левой стороны и нажать **RETURN** или нажать кнопку **Search**;
- Обновить описание группы – обновить существующее описание и нажать кнопку **Save** справа от группы;
- Обновить список, привязанный к ограничениям – чтобы отменить привязку ограничения, нужно нажать кнопку «-» справа от имени ограничения, чтобы привязать ограничение, нужно ввести его имя в поле внизу списка ограничений и нажать кнопку «+». Чтобы принять изменения, нужно нажать кнопку **Save**;
- Удалить группу – нажать кнопку **Delete** справа от группы.

6.13.2 Дамп-файл с настройками LP

Дамп-файл с настройками LP содержит все настройки всех сервисов LP.

6.13.2.1 Получение дамп-файла

Можно извлечь существующие настройки сервиса из Configurator, создав дамп-файл. Это может понадобиться для сохранения текущих настроек сервиса.

Чтобы получить дамп-файл необходимо воспользоваться следующими командами:

- wget: `wget -O settings_dump.json 127.0.0.1:5070/1/dump`;
- curl: `curl 127.0.0.1:5070/1/dump > settings_dump.json`;
- редактор текста.

Будут получены текущие значения, имеющиеся в сервисе Configurator.

6.13.2.2 Применение дамп-файла

Чтобы применить сохраненные настройки, нужно использовать скрипт `db_create.py` с аргументом командной строки`--dump-file` (за которым следует имя созданного дамп-файла): `base_scripts/db_create.py --dump-file settings_dump.json`.

Применить дамп-файл можно **только к пустой базе данных с созданными таблицами**. Если требуется обновление настроек, перед его применением следует удалить всю группу «limitations» из дамп-файла.

```
    "limitations":[
      ...
    ],
```

Для применения дамп-файла нужно выполнить следующие действия:

1. Войти в контейнер сервиса Configurator;
2. Запустить скрипт `python3 base_scripts/db_create.py --dump-file settings_dump.json`

> Ограничения из существующих файлов ограничений заменяются ограничениями из дамп-файла, если наименования ограничений совпадают.

6.13.3 Файл с ограничениями

6.13.3.1 Получение файла с ограничениями

Файл с ограничениями содержит ограничения указанного сервиса. Он не содержит имеющиеся настройки и их значения.

Чтобы загрузить файл с ограничениями для одного или нескольких сервисов, необходимо выполнить следующие команды:

1. Войти в контейнер сервиса Configurator;
2. Создать выходной каталог base_scripts/results: `mkdir base_scripts/results`;

3. Запустить скрипт base_scripts/get_limitation.py: `python3 base_scripts/get_limitation.py --service luna-image-store luna-handlers --output base_scripts/results/my_limitations.json`.

Обратите внимание на параметры скрипта base_scripts/get_limitation.py:

- `--service` для указания одного или нескольких имен сервисов (обязательно);
- `--output` для указания каталога или файла для сохранения выходных данных. Значение по умолчанию: current_dir/{timestamp}_limitation.json (необязательно).

6.13.4 Авторизация

В сервисе Configurator доступна возможность использования авторизации типа BasicAuth для запросов, логически требущих авторизации.

Использование авторизации позволяет обеспечить дополнительную степень безопасности для шифрования и защиты от несанкционированного доступа.

Для включения авторизации нужно задать следующие настройки из новой секции «LUNA_CONFIGURATOR_AUTHORIZATION» сервиса Configurator:

- «USE_AUTHORIZATION» - включение/выключение авторизации
- «LUNA_CONFIGURATOR_USER» - логин, который будет использоваться другими сервисами для авторизации
- «LUNA_CONFIGURATOR_PASS» - пароль, который будет использоваться другими сервисами для авторизации

Задать настройки можно двумя способами:

- передача настроек через переменную окружения `VL_SETTINGS` при старте сервиса. Например:

```
env "VL_SETTINGS.LUNA_CONFIGURATOR.USE_AUTHORIZATION=1"
env "VL_SETTINGS.LUNA_CONFIGURATOR.LUNA_CONFIGURATOR_USER=luna"
env "VL_SETTINGS.LUNA_CONFIGURATOR.LUNA_CONFIGURATOR_PASS=root"
```

- передача настроек в конфигурационный файл сервиса Configurator. Конфигурационный файл сервиса Configurator расположен по следующему пути в поставке `example-docker/luna_configurator/configs/luna_configurator_postgres.conf`

Для того, чтобы остальные сервисы LUNA PLATFORM могли выполнять запросы к сервису Configurator нужно передать им логин и пароль. Это можно сделать с помощью следующих способов:

- передача настроек «LUNA_CONFIGURATOR_USER» и «LUNA_CONFIGURATOR» через переменную окружения `VL_SETTINGS` при старте сервиса (см. пример выше)

- передача настроек «LUNA_CONFIGURATOR_USER» и ”LUNA_CONFIGURATOR в конфигурационный файл сервиса при его использовании

6.14 Сервис Licenses

6.14.1 Общая информация

Сервис Licenses содержит информацию о доступных лицензируемых возможностях и их ограничениях.

Получить информацию о лицензии можно тремя способами:

- с помощью запроса «get license» к сервису Licenses
- с помощью запроса «get system info» к сервису Admin
- с помощью пользовательского интерфейса сервиса Admin

Также можно использовать запрос «get platform features», в ответе на который можно получить информацию о состоянии лицензии, включенных функциях лицензии («face_quality», «body_attributes» и «liveness») и состоянии опциональных сервисов Image Store, Events, Tasks и Sender из настройки «ADDITIONAL_SERVICES_USAGE» сервиса Configurator.

Если отключить какую-то лицензируемую функцию, и попытаться использовать запрос, требующий этой функции, то будет возвращена ошибка 33002 с описанием «License problem Failed to get value of License feature {value}».

6.14.2 Информация о лицензии

Лицензия LP содержит следующие параметры:

- Дату окончания лицензии.
- Максимальное количество лиц с прикрепленными биометрическими шаблонами или базовыми атрибутами.
- Доступность функционала OneShotLiveness.
- Текущее количество транзакций OneShotLiveness.
- Доступность функционала Deepfake.
- Доступность функционала для проверки изображений на соответствие стандарту ISO/IEC 19794-5:2011.
- Доступность функционала оценки параметров тел.
- Доступность функционала оценки количества людей на изображении.
- Возможность использования сервиса Lambda.
- Возможность использования сервиса Indexed Matcher в модуле LUNA Index Module.
- Максимально допустимое одновременное количество потоков, которые могут быть взяты единовременно в обработку из FaceStream.
- Максимально допустимое одновременное количество потоков, которые могут быть взяты единовременно в обработку из сервиса Video Manager.

При заказе лицензии нужно сообщить в техническую поддержку о необходимости использования какого-либо из вышеперечисленных параметров.

Параметры «Возможность использования сервиса Indexed Matcher в модуле LUNA Index Module» и «Максимально допустимое одновременное количество потоков, которые могут быть взяты единовременно в обработку из FaceStream» описаны в документации LUNA Index Module и FaceStream соответственно.

Для некоторых параметров доступны уведомления при приближении к лимиту. Уведомления работают в виде отправки сообщений в логи соответствующего сервиса. Например, при приближении к допустимому количество созданных лиц с биометрическими шаблонами, в логах сервиса Faces будет выводиться следующее сообщение: «License limit exceeded: 8 % of the available license limit is used. Please contact VisionLabs for license upgrade or delete redundant faces». Уведомления работают за счет постоянного мониторинга, реализованного с помощью базы данных Influx. Данные мониторинга сохраняются в соответствующие поля базы данных Influx.

> См. подробную информацию в разделе «Мониторинг».

6.14.2.1 Дата окончания лицензии

После истечения срока действия лицензии невозможно продолжать использование LUNA PLATFORM.

По умолчанию уведомление об окончании срока действия лицензии отправляется за две недели до истечения срока действия.

Когда срок действия лицензии истекает, сервис API выдает сообщение «License has expired. Please contact VisionLabs for license extension.»

Сервис Licenses пишет данные о сроке истечения лицензии в логи и базу мониторинга Influx в поле «license_period_rest».

6.14.2.2 Максимальное количество лиц

Сервис Faces проверяет количество оставшихся лиц на основе информации о максимально возможном количестве лиц из сервиса Licenses. Учитываются только лица со связанными биометрическими шаблонами или лица со связанными базовыми атрибутами.

Процент созданных лиц записывается в логах сервиса Faces и отображается в пользовательском интерфейсе Admin.

Сервис Faces пишет данные о созданных лицах в логи и базу мониторинга Influx в поле «license_faces_limit_rate».

Система начинает присылать уведомления, когда остается 15% от общего количества доступных лиц. При превышении максимального количества доступных лиц, в журналах появится сообщение «License limit exceeded: 8 % of the available license limit is used. Please contact VisionLabs for license upgrade or delete redundant faces». К лицам нельзя прикреплять атрибуты, если количество лиц превышает 110%.

Последствия отсутствия параметра

При отключении данного параметра, будет невозможно выполнить следующие запросы:

- «create face»
- «generate events» с указанием обработчика с «policies» > «storage_policy» > «face_policy» > «store_face»

6.14.2.3 OneShotLiveness

Для оценки Liveness с помощью эстиматора OneShotLiveness доступна безлимитная лицензия или лицензия с ограниченным количеством транзакций.

Каждое использование Liveness в запросах уменьшает счётчик транзакций. После исчерпания лимита транзакций будет невозможно использовать оценку Liveness в запросах. На запросы не использующие Liveness или с отключённой оценкой Liveness исчерпание лимита не влияет, они продолжают работать в обычном режиме.

Сервис Licenses содержит информацию о количестве оставшихся транзакций Liveness. Оно отображается в ответе ресурса «/license».

Сервис Remote SDK пишет данные о количестве доступных транзакций Liveness в логи и базу мониторинга Influx в поле «liveness_balance».

Система начинает присылать уведомления при достижении 2000 оставшихся транзакций Liveness (данный порог задан в системе).

См. раздел «Описание OneShotLiveness» для более подробной информации о работе Liveness.

Последствия отсутствия параметра

При отключении данного параметра, будет невозможно выполнить оценку Liveness (параметр «estimate_liveness») в следующих запросах:

- «sdk»
- «detect faces»
- «generate events» с указанием обработчика с «policies» > «detect_policy» > «estimate_liveness»
- «perform verification» с указанием обработчика с «policies» > «detect_policy» > «estimate_liveness»
- «estimator task» с указанием обработчика с «policies» > «detect_policy» > «estimate_liveness»
- «predict liveness»

6.14.2.4 Оценка параметров тел

Данный параметр позволяет выполнять оценку параметров тел. В лицензии может быть задано два значения — 0 или 1. Для данного параметра не предназначен мониторинг.

Последствия отсутствия параметра

При отключении данного параметра, будет невозможно выполнить оценку параметров тел (параметры «estimate_upper_body», «estimate_lower_body», «estimate_body_basic_attributes», «estimate_accessories») в следующих запросах:

- «sdk»
- «generate events» с указанием обработчика с «policies» > «detect_policy» > «body_attributes»

6.14.2.5 Оценка количества людей

Данный параметр позволяет выполнять оценку количества людей. В лицензии может быть задано два значения — 0 или 1. Для данного параметра не предназначен мониторинг.

Последствия отсутствия параметра

При отключении данного параметра, будет невозможно выполнить оценку количества людей (параметр «estimate_people_count») в следующих запросах:

- «sdk»
- «generate events» с указанием обработчика с «policies» > «detect_policy» > «estimate_people_count»

6.14.2.6 Проверка по стандарту ISO/IEC 19794-5:2011

Данный параметр позволяет выполнять различные проверки изображений на соответствие стандарту ISO/IEC 19794-5:2011. В лицензии может быть задано два значения — 0 или 1. Для данного параметра не предназначен мониторинг.

Последствия отсутствия параметра

При отключении данного параметра, будет невозможно выполнить следующие запросы:

- «iso»
- «detect faces» с параметром «estimate_face_quality»
- «generate events» с указанием обработчика с «policies» > «detect_policy» > «face_quality»
- «estimator task» с указанием обработчика с «policies» > «detect_policy» > «face_quality»
- «perform verification» с указанием обработчика с «policies» > «detect_policy» > «face_quality»

6.15 Сервисы видеоаналитики

Сервисы видеоаналитики предназначены для обработки видеофайлов или видеопотоков с целью выполнения различных аналитических задач. Эти задачи могут включать в себя подсчет людей, отслеживание людей на кадре, детектирование драк и многое другое.

> Далее в документации для обозначения обрабатываемой сущности (видеофайла или видеопотока) будет использоваться понятие **поток**.

Работа сервисов видеоаналитики основана на взаимодействии трех ключевых компонентов: аналитик, агентов и сервиса Video Manager.

Примечание. Для возможности создания потоков требуется лицензируемая функция, определяющая максимально допустимое одновременное количество потоков, обрабатываемых сервисом Video Agent. Сервис Video Manager проверяет наличие лицензии каждый раз при создании потока, а также проверяет отчеты (если они есть) для определения количества обрабатываемых потоков сервисом Video Agent. См. подробную информацию о лицензируемых функциях в разделе «Сервис Licenses».

Аналитика

Аналитика представляет собой набор алгоритмов, которые выполняют определенные задачи по обработке потоков. Для каждой аналитики предусмотрен набор параметров, который может быть настроен пользователем. Video Manager хранит информацию о доступных аналитиках и передает ее агентам, которые реализуют соответствующие алгоритмы.

Агент

Агент — это компонент, который реализует алгоритмы видеоаналитики. Он принимает потоки, выполняет заданную аналитику и отправляет результаты через механизм Callback'ов.

В качестве агента можно использовать сервис Video Agent, либо написать собственного агента. См. пример кода в руководстве разработчика сервиса Video Manager.

Сервис Video Agent предоставляет следующие аналитики:

- видеоаналитика подсчета количества людей
- видеоаналитика отслеживания людей

Video Manager

Video Manager выступает в роли координатора между пользователями и агентами. Он содержит информацию о доступных аналитиках и распределяет видеопотоки на обработку среди агентов. Video Manager также отвечает за управление потоками, следит за их статусом и координирует автоматический перезапуск потоков в случае возникновения ошибок.

См. подробную информацию о взаимодействии сервиса Video Manager и агента в разделе «Взаимодействие Video Manager и агента».

Базовый принцип работы выглядит следующим образом:

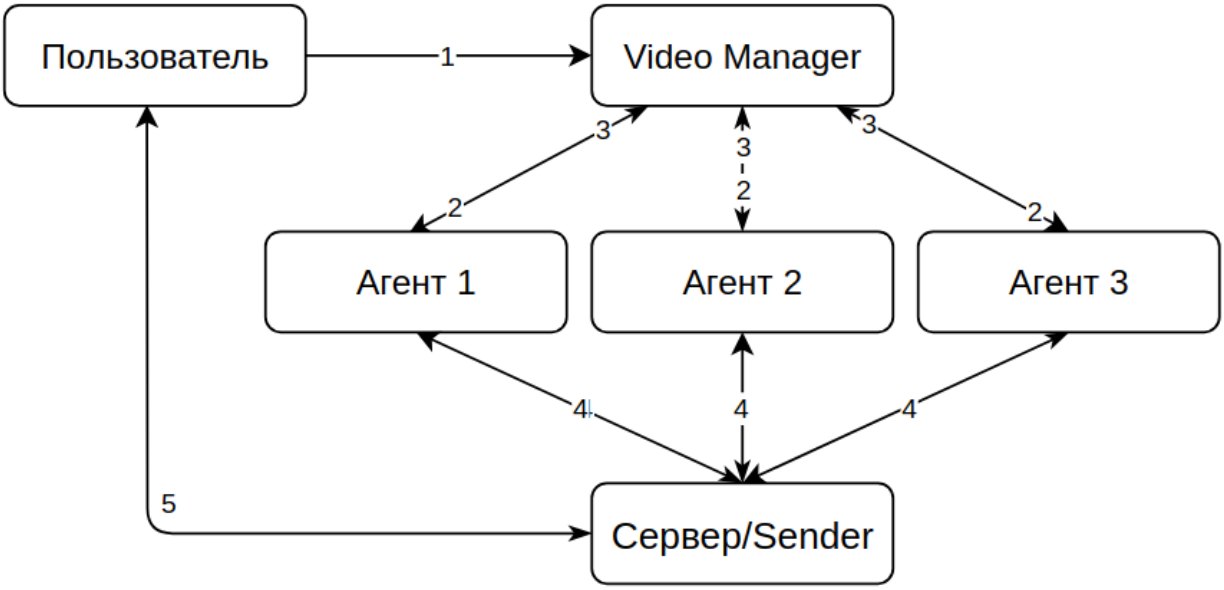


Рис. 42: Базовый принцип работы видеосервисов

1. Пользователь выполняет запрос на создание потока с помощью запроса «create stream».
2. Сервис Video Manager направляет поток агенту, способному обработать все аналитики, переданные в запросе.
3. Агент обрабатывает поток и отправляет статус обработки и возможные ошибки сервису Video Manager.
4. Агент отправляет результаты обработки в соответствии с выбранным типом Callback'ов (см. раздел «Отправка и сохранение событий»).

> Также можно получить результаты обработки, выполнив запрос `stream events ws handshake`, позволяющий установить веб-сокет соединение для получения результатов обработки конкретного потока в реальном времени с помощью указания «stream_id».

Пользователь также может пересоздавать поток, получать статус обработки потока, останавливать и запускать поток и др. См. подробную информацию в группе запросов streams спецификации OpenAPI.

См. подробную информацию о том как пользователь использует сервисы видеоаналитики в разделе «Взаимодействие пользователя и Video Manager».

См. раздел «Быстрое начало по созданию потока» ниже для изучения базовых шагов по созданию потока.

См. подробную информацию в разделе «Взаимодействие Video Manager и агента».

6.15.1 Отправка и сохранение событий

Агент может отправлять результаты обработки через механизм Callback'ов с типами:

- «http», предварительно настроив сервер для приема сообщений
- «luna-ws-notification», предварительно установив веб-сокет соединение
- «luna-event»

Типы «http» и «luna-ws-notification» являются стандартными. См. раздел «Отправка событий в сторонний сервис».

Тип «luna-event» предназначен для сохранения событий в базу данных Events с использованием механизма обобщенных событий. Получить такие события можно с помощью запросов «get general events» и «get general event», указав в качестве фильтра полученный идентификатор «stream_id».

6.15.2 Аналитики

С точки зрения сервиса Video Manager, аналитика представляет собой объект, содержащий имя, неизвестный набор параметров и, при необходимости, описание и документацию. Аналитику создает либо агент, либо администратор, отправляя запрос «create analytic» в сервис Video Manager.

> В сервисе Video Agent доступно две аналитики - `people_count` и `human_tracking`. Аналитики автоматически регистрируется при запуске сервиса Video Agent.

Video Manager распределяет потоки по агентам, в том числе опираясь на знание того, какой агент готов сделать ту или иную аналитику. Когда агент или администратор создает аналитику, он может указать набор параметров, которые будут использоваться при запуске аналитики. Однако при создании запроса на обработку потока пользователь может указать и другие параметры, которые будут использоваться в рамках этой аналитики (используя параметр «analytics» > «parameters» запроса «create stream»).

Подробная информация о доступных параметрах и их описаниях различна для каждой аналитики. Она доступна в документации соответствующей аналитики и может быть получена в ответ на запрос «get analytics».

6.15.3 Взаимодействие пользователя и Video Manager

Для выполнения аналитики потока пользователь должен создать поток, указав желаемую аналитику.

Для создания потока необходимо сделать запрос «create stream». Для пересоздания потока необходимо сделать запрос «put stream», в этом случае будут использованы новые данные потока, а версия потока будет увеличена на 1.

После появления нового потока он переходит в статус «pending».

Когда поток обрабатывается одним из доступных агентов (см. раздел «Распределение потоков»), он будет иметь статус «in_progress».

Пока происходит обработка потока, агент отправляет обратную связь сервису Video Manager, который создает логи, которые может получить пользователь, выполнив запрос «get streams logs».

Обработку потока можно остановить:

- по запросу пользователя (запрос «patch stream» с параметром «action» = «stop»). В таком случае статус потока будет «stop».
- по запросу пользователя (запрос «remove stream»). В таком случае поток будет удален.
- когда поток закончится. В таком случае статус потока будет «done».
- при возникновении фатальной ошибки. В таком случае статус потока будет «failure».

Во всех случаях, за исключением ручного удаления логов с помощью запроса «delete stream logs», логи потоков будут доступны после выполнения запроса «get streams logs».

Во всех случаях, за исключением удаления потока с помощью запроса «remove stream», данные потока будут доступны после выполнения запроса «get stream».

6.15.4 Взаимодействие Video Manager и агента

Взаимодействие сервиса Video Manager и агента описано ниже:

1. Агент или администратор должен зарегистрировать в сервисе Video Manager аналитику с которой агент умеет работать, выполнив запрос к ресурсу `/1/analytic` сервиса Video Manager.
2. При старте агент должен выполнить запрос к ресурсу `/1/agent` сервиса Video Manager на регистрацию самого себя. В данном запросе указываются следующие параметры:
   - имя агента (обязательно)
   - максимальное количество потоков, которое может обрабатывать агент (обязательно)
   - набор аналитик, с которыми может работать агент (обязательно)
   - описание агента

   В ответе на запрос агент получит уникальный идентификатор «agent_id».
3. Агент периодически должен выполнять запросы к ресурсу `/1/agent/{agent_id}/streams` сервиса Video Manager для получения/остановки потока на обработку.

   Как только агент получил поток на обработку, сервис Video Manager запускает счетчик потоков, обрабатываемых данным агентом.

Обратите внимание, что если агент не сделает такой запрос вовремя, статус агента будет помечен как «not_ready» и агент будет исключен из списка агентов, которые могут обрабатывать потоки (подробности см. в разделе «Управление статусом агента в случае отсутствия запроса агента»).

4. Во время обработки, агент должен отправлять обратную связь о статусе обработки потока к ресурсу `/1/agent/{agent_id}/streams/feedback`.

   Отчет содержит идентификатор потока, его статус, ошибку, версию потока и время генерации отчета. После отправки отчета пользователь может получить логи обработки с помощью запроса «get streams logs».

Обратите внимание, что пользовательский агент должен вышеописанные действия. См. пример кода в разделе «Agent interaction» в руководстве разработчика сервиса Video Manager.

Сервис Video Agent автоматически выполняет вышеописанные действия.

6.15.5 Потоки

Потоки создает пользователь с помощью запроса «create stream».

Пользователь может задать имя потока, его описание, добавить информацию о городе или улице, настроить автоматический перезапуск потока, задать параметры группировки потока, аналитки, также нижеперечисленные общие данные обработки потока:

- «type» - тип источника: видеофайл или видеопоток
- «reference» - адрес источника
- «rotation» - угол поворота кадра камеры
- «downloadable» - определяет необходимость предварительного скачивания видеофайла перед его обработкой

  Предварительная загрузка необходима для:

  - **Успешного декодирования видеофайлов**. Для некоторых файлов могут возникнуть ошибки декодирования, если файлы не будут предварительно сохранены.
  - **Длительных процессов обработки**. Если обработка видео занимает значительное время, хранение предотвращает проблемы, связанные с разрывом соединения.

  Обратите внимание, что иногда политики безопасности запрещают сохранение видеофайлов. В таких случаях предварительное скачивание может быть не применимо.

- «timestamp_source» - определяет, откуда будут браться временные метки для видеоанализа.

  Доступны следующие значения:

  - «pts» - Использует временные метки видео, если они существуют, для точного отображения времени воспроизведения. Метки не всегда могут быть корректными (см. ниже).

- «server» - Применяет серверное время для видеопотока, что гарантирует единообразие и синхронизацию с другими событиями.
- «frame_rate» - Применяет частоту кадров для видеофайлов, что помогает приблизительно определить временные метки.
- «auto» (по умолчанию) - Автоматически выбирает источник времени, сначала проверяя временные метки («pts»), а затем переключаясь на серверное время («server») или частоту кадров («frame_rate»), если метки некорректны.

Корректные метки для видеофайла означают, что временные метки (PTS) должны быть близки к нулю, то есть время от начала видео до метки должно быть относительно небольшим (абсолютное значение не должно превышать 10^5 секунд).

Корректные метки для видеопотока означают, что время видеопотока отличается от времени сервера менее чем на 1 день.

- «pts» > «start_time» - смещение для временных меток видео (PTS)

  Указание смещения позволяет синхронизировать начало обработки видео с конкретным моментом времени. Это может быть полезно, например, если видео обрабатывается частями и необходимо, чтобы временные метки новых частей продолжали последовательность с предыдущих. Таким образом, при нарезке большого видео на маленькие фрагменты, временные метки будут отображаться так, как если бы обрабатывалось одно целое видео.

6.15.5.1 Формат потоков, поддерживаемый сервисом Video Agent

Сервис Video Agent может обрабатывать потоки в различных форматах. Если формат потока не соответствует поддерживаемым, возникает ошибка. При этом ведется запись логов и поток считается неудачным. В общем, сервис Video Agent поддерживает обработку всех потоков, которые могут быть обработаны с использованием `ffmpeg`, но существует известное ограничение на формат пикселей. Сервис Video Agent поддерживает следующие форматы пикселей: `BGR`, `BGR_32F`, `NV12`, `P10`, `P12`, `RGB`, `RGB_32F`, `RGB_32F_PLANAR`, `RGB_PLANAR`, `YCBCR`, `YUV420`, `YUV420_10bit`, `YUV422`, `YUV444`, `YUV444_10bit`.

Если поток имеет формат, который не поддерживается, его можно перекодировать, например, с помощью `ffmpeg`:

```
ffmpeg -rtsp_transport tcp -i входной<поток_> -rtsp_transport tcp -pix_fmt
   yuv420p -c:v copy -f rtsp выходной<поток_>
```

Важно! Описание выше распространяется только на сервис Video Agent. Если у пользователя собственный агент, то пользователь самостоятельно настраивает формат обработки потоков.

6.15.5.2 Статусы потока

Поток может иметь следующие статусы:

- «pending» — поток взят в работу, но пока не найдено ни одного агента. Такой статус может быть сразу после создания, пересоздания потока и во время автоматического перезапуска
- «in_progress» — поток обрабатывается агентом
- «done» — поток полностью обработан
- «restart» — сервер перезапустил обработку потока с помощью автоматического перезапуска
- «failure» — от агента был получен отчет об ошибке обработки потока
- «cancel» — обработка потока была отменена, т.к. поток был удален из сервиса Video Manager
- «stop» — поток был остановлен с помощью запроса «patch stream»

6.15.5.3 Жизненный цикл потока

Жизненный цикл потока представлен ниже:

1. Поток создается с помощью запроса «create stream». Статус изменяется на «pending»
2. Агент принимает поток для обработки. Статус изменяется на «in_progress».
3. **Опционально.** Выполняется остановка/продолжение обработки потока:
   - Выполняется запрос «patch stream» с параметром «action» = «stop». Статус изменяется на «stop»
   - Выполняется запрос «patch stream» с параметром «action» = «resume». Статус изменяется на «pending» до того момента, как очередной агент не возьмет поток в обработку. При этом статус изменится на «in_progress»
4. Окончание обработки:
   - Окончание потока. Агент останавливает обработку и отправляет обратную связь сервису Video Manager. Статус изменяется на «done».
   - Поток удаляется с помощью запроса «remove stream».
   - Обработка заканчивается из-за ошибки, которую отправляет агент в обратной связи. Статус потока меняется на «failure».

6.15.5.4 Автоматический перезапуск потоков

Возможность автоматического перезапуска потоков актуальна только для потоков со статусом «failure». Параметры автоматического перезапуска (возможность перезапуска, максимальное количество попыток перезапуска, задержка между попытками) задаются пользователем для каждого потока в разделе «autorestart» запроса «create stream». Параметры и статус автоматического перезапуска можно получить с помощью запроса «get stream».

Ниже перечислены статусы автоматического перезапуска:

- «disabled» — автоматический перезапуск отключен пользователем (отключен параметр «restart»)

- «enabled» — автоматический перезапуск включен, но в данный момент не активен поскольку поток не находится в статуса «failure»
- «in_progress» — автоматический перезапуск в процессе
- «failed» — превышено допустимое количество попыток автоматического перезапуска и ни одна попытка не увенчалась успехом

6.15.5.5 Распределение потоков

Сервис Video Manager распределяет потоки, предоставленные пользователями агентам, следующим образом:

- выбирает все потоки, которые требуют обработки (обработке подлежат только потоки со статусом «pending»). См. раздел «Статусы потока» для более подробной информации.
- выбирает все доступные агенты, количество потоков обработки которых в текущем режиме не достигает максимального количества потоков, доступных для одновременной обработки агентом. См. раздел «Взаимодействие Video Manager и агента» для более подробной информации.
- в порядке очередности создания потоков выбирает агента для каждого потока, удовлетворяющего следующим условиям:
  - агент может обрабатывать аналитику, указанную для потока
  - есть свободный слот для потока (агент предоставляет параметр `max_stream_count` - максимальное количество потоков, доступных ему для одновременной обработки, поэтому количество одновременных потоков для агента не может превышать это число)
- вносит необходимые записи в базу данных для ответа на запрос агента в соответствии с предыдущей логикой
- агент получает 2 типа потоков через запрос «get agent streams» — потоки, которые необходимо обработать, и потоки, обработка которых должна быть остановлена. Поток, однажды упомянутый в ответе на запрос, как поток, обработка которого должна быть начата, больше не будет упоминаться в ответах на этот запрос до тех пор, пока его обработку не потребуется остановить. То же самое относится и к потокам, обработку которых необходимо остановить — они больше не будут упоминаться в ответах на этот запрос до тех пор, пока их обработку не потребуется начать снова.

Описанная процедура будет выполняться как периодическая фоновая задача главного экземпляра. Период выполнения можно настроить с помощью параметра `luna_video_manager_streams_agent_sea`. Рекомендуется проконсультироваться со специалистами VisionLabs перед изменением данного параметра.

6.15.5.6 Процесс перезапуска потоков

Ниже описан процесс обработки потоков с включенным автоматическим перезапуском. Описанные действия будут выполняться как периодическая фоновая задача главного экземпляра сервиса Video Manager, период выполнения можно настроить с помощью параметра `luna_video_manager_streams_autorestarter_interval`. Рекомендуется проконсультироваться со специалистами VisionLabs перед изменением данного параметра.

Когда происходит попытка автоматической перезагрузки потока данных, происходят следующие изменения:

- статус потока изменяется сначала на «restart», а затем на «pending»;
- счетчик попыток автоматической перезагрузки «current_attempt» увеличивается на 1;
- запись времени последней попытки «last_attempt_time» обновляется, чтобы отразить актуальное время.

Для того чтобы произошла перезагрузка, должны соблюдаться следующие условия:

- статус потока должен находиться в состоянии «failure»;
- должен быть включен автоматический перезапуск потока (параметр «restart»);
- значение текущей попытки автоматической перезагрузки «current_attempt» должно быть равно «null» или меньше, чем максимальное число попыток «attempt_count»;
- время последней попытки автоматической перезагрузки «last_attempt_time» должно быть равно «null» или разница между текущим временем и временем последней попытки должна быть больше или равна задержке «delay».

Если соблюдаются условия ниже, то произойдет сбой автоматического перезапуска потока (прекращение попыток перезапуска):

- статус потока находится в состоянии «failure»;
- статус автоматического перезапуска потока находится в состоянии «in_progress»;
- значение текущей попытки автоматической перезагрузки «current_attempt» равно значению максимального числа попыток «attempt_count».

Если соблюдаются условия ниже, то произойдет завершение работы автоматического перезапуска потока:

- статус потока не равен статусу «failure»;
- статус автоматической перезагрузки потока находится в состоянии «in_progress»;
- время последней попытки автоматической перезагрузки «last_attempt_time» равно «null» или разница между текущим временем и временем последней попытки больше или равна задержке «delay».

6.15.5.7 Группировка потоков

Потоки можно группировать. Группировка призвана объединять потоки с множеством камер в логические группы. Например, можно группировать потоки по территориальному признаку.

Поток может привязан к нескольким группам.

Группа создается с помощью запроса «create group». Для создания группы нужно указать обязательный параметр «group_name». При необходимости можно указать описание группы.

Поток может быть привязан к группе с помощью параметров «group_name» или «group_id» во время создания потока (запрос «create stream»).

Если поток был привязан к группе, то при запросе «get stream» или «get streams» группа будет отражена в поле «groups».

6.15.6 Работа с несколькими экземплярами

Каждый экземпляр сервиса Video Manager может обслуживать ограниченное количество пользовательских запросов, а каждый агент может анализировать ограниченное количество потоков, поэтому эти сервисы можно масштабировать таким образом, чтобы каждый сервис Video Manager имел доступ к одной и той же базе данных и каждый агент имеет доступ к одному из экземпляров сервиса Video Manager.

Все запущенные экземпляры сервиса Video Manager выбирают главный экземпляр для корректного выполнения ключевых процессов, таких как:

- распределение потоков между агентами
- автоматический перезапуск потоков
- обработка потоков в случае не появления обратной связи
- управление статусом агента случае отсутствия запроса агента

Эти процессы должны выполняться только одним экземпляром, чтобы избежать состояния гонки, когда несколько экземпляров пытаются одновременно выполнять одни и те же задачи, что может привести к некорректной работе системы.

6.15.6.1 Выбор главного экземпляра

Для выбора главного экземпляра каждый экземпляр сервиса пытается получить блокировку базы данных, обновляя поле в таблице `single_process_lock`. Если экземпляру удается получить блокировку, он объявляется главным экземпляром и сохраняет это состояние, регулярно отправляя в базу данных сигнал «heartbeat». Этот сигнал позволяет другим экземплярам знать, что главный экземпляр активен и выполняет свои задачи. Если сигнал «heartbeat» от главного экземпляра перестает поступать, это означает, что главный экземпляр больше не активен, и любой другой экземпляр может попытаться стать новым главным экземпляром, получив блокировку базы данных.

Таким образом, выбор главного экземпляра обеспечивает координацию и предотвращает конфликтные ситуации, гарантируя, что только один экземпляр выполняет критические задачи в любой момент времени.

6.15.6.2 Обработка потоков в случае не появления обратной связи

Процесс обработки видеопотоков подразумевает получение обратной связи для каждого потока, находящегося в состоянии «in_progress». Сервис Video Manager периодически проверяет время последнего получения обратной связи по каждому потоку. Если обратная связь не была получена в установленное время, статус потока изменится на «restart», что означает попытку заново начать обработку потока. Затем статус потока сразу изменяется на «pending», чтобы поставить его в очередь на обработку.

Проверка условий для обновления статуса потоков выполняется как периодическое фоновое задание главным экземпляром. Период выполнения можно настроить с помощью параметра `luna_video_manager_stream_status_obsoleting_interval`. Потоки будут считаться устаревшими, если время последнего получения обратной связи превышает установленный период, который настраивается с помощью параметра `luna_video_manager_stream_status_obsoleting_period`.

Рекомендуется проконсультироваться со специалистами VisionLabs перед изменением данных параметров.

6.15.6.3 Управление статусом агента в случае отсутствия запроса агента

Взаимодействие с агентами подразумевает периодические запросы к сервису Video Manager (см. раздел «Взаимодействие Video Manager и агента»). Сервис Video Manager периодически проверяет время последнего запроса агентов на обработку потоков, и если это время было слишком давно, статус агента изменяется на «not_ready», что означает исключение агента из очереди распределения потоков.

Если такой агент позже сделает запрос на обработку потоков, его статус будет обновлен на «ready», и агент будет добавлен в список агентов, которые могут обрабатывать потоки.

Проверка условий для обработки потоков в случае отсутствия запроса агента выполняется как периодическое фоновое задание главным экземпляром. Период выполнения можно настроить с помощью параметра `luna_video_manager_agent_status_obsoleting_interval`. Агенты будут считаться не готовыми, если последний запрос от агента не был получен вовремя, что можно настроить с помощью параметра `luna_video_manager_agent_status_obsoleting_period`.

Рекомендуется проконсультироваться со специалистами VisionLabs перед изменением данных параметров.

6.15.7 Настройка получения результатов аналитики

Результат выполнения аналитики можно получить **только с помощью механизма Callback'ов** через веб-хуки или веб-сокеты. Соответственно, для получения результатов аналитики должно быть установлено соответствующее соединение.

Если используются веб-хуки, то должен быть развернут сервер, принимающий уведомления. Если используются веб-сокеты, то сервис Sender должен быть включен в настройке «ADDITIONAL_SERVICE_USAGE» и должно быть установлено соединение с сервисом Sender (см. запрос «ws handshake for general events»).

Также можно получать результаты эстимации (не события) в режиме реального времени с помощью запроса «stream events ws handshake». Так, например, если выполняется аналитика количества людей, то можно получать координаты людей в тот момент, когда они появляются на кадре.

6.15.8 Быстрое начало по созданию потока

В данном примере будут описаны базовые действия для создания потока, его обработки и просмотра результатов обработки. В качестве агента будет использоваться сервис Video Agent.

1) Подготовьте тело запроса «create stream».

Секция «analytics» > «parameters» отличается для каждого типа аналитики. Узнать параметры конкретной аналитики можно с помощью соответствующих спецификаций в комплекте поставки по пути docs/ReferenceManuals или на сайте с онлайн-документацией.

Онлайн-документация:

- Спецификация аналитики отслеживания людей
- Спецификация аналитики подсчета количества людей

> Также можно получить документацию аналитики с помощью запроса «get analytic documentation», указав идентификатор аналитики, который можно получить с помощью запроса «get analytics». Запрос «get analytic documentation» подразумевает обращение к ресурсу `/6/analytics/{analytic_id}/docs`. Рекомендуется открывать данный ресурс в браузере или указывать заголовок «Accept» = «text/html» для корректного отображения HTML-документации.

Содержимое параметров аналитики необходимо добавить в тело запроса «create stream».

Например, можно создать следующее тело запроса для аналитики `people_count`:

```
{
    "name": "name_example",
    "description": "description_example",
    "data": {
        "type": "videofile",
        "reference": "https://example.com/humantracking.mp4",
        "rotation": 0
    },
    "location": {
```

```
            "city": "Moscow",
            "area": "Central",
            "district": "Basmanny",
            "street": "Podsosensky lane",
            "house_number": "23 bldg.3",
            "geo_position": {
                "longitude": 36.616,
                "latitude": 55.752
            }
        },
        "autorestart": {
            "restart": 0,
            "attempt_count": 10,
            "delay": 60
        },
        "analytics": [
            {
                "analytic_name": "people_count",
                "parameters": {
                    "parameters": {
                        "probe_count": 2,
                        "image_retain_policy": {
                            "mimetype": "PNG"
                        }
                    },
                    "callbacks": [
                        {
                            "type": "luna-event",
                        }
                    ],
                    "targets": [
                        "coordinates",
                        "overview"
                    ]
                }
            }
        ]
    }
```

Сохраните полученный идентификатор «stream_id».

2) Выполните запрос «get general events» или «get general event» с приминением фильтра по «stream_id».

6.16 Сервис Streams Retranslator

Сервис Streams Retranslator обеспечивает ретрансляцию видеопотоков, предоставляя их в формате HLS, удобном для отображения в пользовательских интерфейсах. Для выполнения ретрансляции используются утилиты FFmpeg и MediaMTX, работающие в контейнере сервиса.

Важно! При запуске контейнера можно прокинуть дополнительные порты для MediaMTX и HLS-потока, чтобы обеспечить доступ к ним через нестандартные порты (см. пример запуска сервиса в документации по установке).

Ретрансляция запускается с помощью запроса «get stream retransmission». В запросе необходимо указать идентификатор потока. Опционально можно указать качество трансляции через параметр «quality». Значение параметра определяет высоту кадра в пикселях. Например, чтобы транслировать поток с высотой 720 пикселей вместо оригинальных 1080, задайте «quality=720». Ширина будет рассчитана автоматически, сохраняя исходное соотношение сторон.

В результате запроса возвращаются следующие данные:

- `url` — ссылка на HLS-поток (например, `http://127.0.0.1:8888/84406ce5-db60-4be1-b741-781f7979ccc5/index.m3u8`, где `127.0.0.1:8888` — хост и порт, а `84406ce5-db60-4be1-b741-781f7979ccc5` — идентификатор ретрансляции).
- `type` — тип трансляции (на текущий момент только HLS).
- `quality` — качество трансляции (в пикселях).
- `token` — JWT-токен для доступа к HLS-потоку.

Примечание. Хост и порт, возвращаемые в ответе, необходимо настроить через параметр «EXTERNAL_LUNA_STREAMS_HLS_RETRANSMISSION_ADDRESS» для того, чтобы использовать эти данные в веб-интерфейсе. Стандартные значения не подойдут если сервис запускается не там же, где потребляется HLS-поток или же если при старте контейнера был проброшен нестандартный порт для HLS (например, вместо `8888` проброшен `8899`).

Данные о трансляции сохраняются в Redis и существуют до завершения ретрансляции.

Продолжительность ретрансляции контролируется параметром «luna_streams_retraslator_ignored_restream_ttl». Этот параметр задаёт время (по умолчанию 60 секунд), в течение которого поток считается востребованным (например, при наличии зрителей). Если в течение указанного времени поток не востребован, ретрансляция автоматически завершится.

При остановке ретрансляции запрос на её повторный запуск создаст новую ссылку и токен. Если запрос поступает до завершения текущей ретрансляции, пользователь получит ту же самую ссылку на трансляцию и токен, поскольку она уже активна.

6.16.1 Принцип работы

1. Пользователь отправляет запрос на рестриминг RTSP-потока с указанием качества (опционально).

2. Сервис:
    - Запускает трансляцию через FFmpeg.
    - Сохраняет ссылку на HLS и токен в Redis.
    - Возвращает ссылку и токен пользователю.
3. Пользователь внедряет ссылку и токен в интерфейс (пример).

6.16.2 Масштабирование

Сервис Streams Retranslator имеет некоторые особенности масштабирования.

Правильный способ использования нескольких экземпляров Streams Retranslator — это разделение запросов между сервисами по различным диапазонам `stream_id` для каждого экземпляра.

Важно! Адрес и порт, на которых работает балансировщик нагрузки, должны соответствовать значениям параметров из группы «EXTERNAL_LUNA_STREAMS_HLS_RETRANSMISSION_ADDRESS».

Каждый поток имеет уникальный идентификатор `stream_id`, представленный в формате `uuid4`. Этот идентификатор создается случайным образом при создании потока, что предполагает их равномерное распределение.

Примеры разделения `uuid4` на диапазоны:

- От `00000000-0000-4000-8000-000000000000` до `80000000-0000-4000-8000-000000000000`
- От `90000000-0000-4000-8000-000000000000` до `ffffffff-ffff-4fff-bfff-ffffffffffff`
- От `00000000-0000-4000-8000-000000000000` до `30000000-0000-4000-8000-000000000000`
- От `40000000-0000-4000-8000-000000000000` до `80000000-0000-4000-8000-000000000000`
- От `90000000-0000-4000-8000-000000000000` до `a0000000-0000-4000-8000-000000000000`
- От `c0000000-0000-4000-8000-000000000000` до `ffffffff-ffff-4fff-bfff-ffffffffffff`

Разделяя запросы по описанному принципу, можно равномерно распределять нагрузку между сервисами.

См. примеры в разделе «Scaling» в руководстве разработчика Streams Retranslator.

6.16.3 Пример HTML-фрагмента для ретрансляции

Ниже представлен пример, позволяющий отобразить поток на веб-странице. Замените `url` и `token` своими значениями:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>HLS Stream</title>
    <script src="https://cdn.jsdelivr.net/npm/hls.js@latest"></script>
</head>
<body>
    <h1>HLS Stream</h1>
    <video id="video" controls></video>
    <script>
        const video = document.getElementById('video');
        const url = "http://127.0.0.1:8888/84406ce5-db60-4be1-b741-781
           f7979ccc5/index.m3u8";
        const token = "eyJhbGciOiJ";

        if (Hls.isSupported()) {
            const hls = new Hls();
            hls.attachMedia(video);
            hls.loadSource(url);
            hls.on(Hls.Events.MEDIA_ATTACHED, () => {
                hls.config.xhrSetup = xhr => xhr.setRequestHeader('
                   Authorization', 'Bearer ' + token);
            });
            hls.on(Hls.Events.MANIFEST_PARSED, () => video.play());
        } else if (video.canPlayType('application/vnd.apple.mpegurl')) {
            video.src = url;
            video.play();
        } else {
            alert('HLS is not supported in this browser.');
        }
    </script>
</body>
</html>
```

6.17 Сервис Lambda

Сервис Lambda предназначен для работы с пользовательскими модулями, имитирующими функционал отдельного сервиса. Сервис позволяет написать и использовать собственный обработчик или написать внешний сервис, который будет тесно взаимодействовать с LUNA PLATFORM и сразу иметь несколько функций, типичных для сервисов LP (таких как логирование, автоматическая перезагрузка конфигураций и т.д.).

Сервис Lambda создает образ Docker, а затем запускает его в кластере Kubernetes. Без Kubernetes невозможно управлять пользовательским модулем. **Полноценная работа с сервисом Lambda возможна только при разворачивании сервисов LUNA PLATFORM в Kubernetes**. Для использования необходимо самостоятельно развернуть сервисы LUNA PLATFORM в Kubernetes или обратиться за консультацией к специалистам VisionLabs. При необходимости можно использовать Minikube для локальной разработки и тестирования, обеспечивая таким образом Kubernetes-подобную среду без необходимости управления полноценным продакшн Kubernetes-кластером.

> Не следует путать данный функционал с механизмом плагинов. Плагины предназначены для реализации узкого целенаправленного функционала, в то время как сервис Lambda позволяет реализовывать функционал полноценных сервисов.

> Настоятельно рекомендуется узнать как можно больше об объектах и механизмах LUNA PLATFORM (особенно об обработчиках), прежде чем приступать к работе с данным сервисом.

Пользовательский модуль, запущенный в кластере Kubernetes, называется **lambda**. Информация о созданной lambda хранится в базе данных Lambda.

Количество создаваемых lambda не ограничено. Для каждой lambda доступна возможность добавить свою спецификацию OpenAPI.

Для работы с сервисом Lambda нужна **отдельная лицензируемая функция**. Если функция недоступна, то при запросе на создание lambda будет возвращаться соответствующая ошибка.

Примечание. Описание, приведенное ниже, предназначено для общего знакомства с функционалом сервиса Lambda. См. руководство разработчика сервиса Lambda для более подробной информации. В руководстве разработчика доступен раздел «Quick start guide», позволяющий начать работу с сервисом.

6.17.1 Перед началом работы

Перед началом работы с сервисом Lambda, необходимо ознакомиться со всеми требованиями и правильно задать настройки сервиса.

6.17.1.1 Требования к коду и архиву

Модуль пишется на языке Python и должен быть передан в сервис Lambda в ZIP-архиве.

Код и архив должны соответствовать определенным требованиям, основные из которых перечислены ниже:

- должен использоваться Python версии 3.11 или выше;
- для разработки необходима библиотека «luna-lambda-tools», доступная в VisionLabs PyPi;
- архив не должен быть запаролен.

Также файлы в архиве должны иметь определенную структуру. См. подробную информацию в разделе «Requirements» руководства разработчика сервиса Lambda.

6.17.1.2 Требования к окружению

Для работы с сервисом Lambda необходимы следующие требования к окружению:

- наличие запущенных сервисов Licenses и Configurator*;
- наличие бакета S3 для хранения архивов;
- наличие Docker registry для хранения образов;
- наличие кластера Kubernetes.

* во время своей работы, lambda будет дополнительно взаимодействовать с некоторыми сервисами LUNA PLATFORM. Перечень сервисов зависит от типа lambda (см. «Типы lambda»).

> При необходимости можно настроить TTL для хранения архивов в S3. См. раздел «Миграция для добавления TTL к объектам в S3».

Должен быть обеспечен доступ на запись/чтение в бакет хранилища S3 и настроены определенные права доступа в кластере Kubernetes. Необходимо переместить базовые образы Lambda на свой Docker registry. Команды для переноса образов приведены в документации по установке LUNA PLATFORM.

> При старте сервиса Lambda выполняется проверка Docker registry и базовых образов.

См. подробную информацию в разделе «Requirements» руководства разработчика сервиса Lambda.

6.17.1.3 Настройки сервиса Lambda

В настройках сервиса Lambda необходимо указать следующие данные:

- местоположение кластера Kubernetes (см. настройку «CLUSTER_LOCATION»):
  - «internal» — сервис Lambda работает в кластере Kubernetes и не требует других дополнительных настроек;
  - «remote» — сервис Lambda работает с удаленным кластером Kubernetes и правильно определенными настройками «CLUSTER_CREDENTIALS» (хост, токен и сертификат);

  - «local»- сервис Lambda работает там же, где запущен кластер Kubernetes.

  В классическом варианте работы с сервисом Lambda предполагается использование параметра «internal».

- настройки бакета S3 (см. настройку «S3»);
- адрес реестра (registry) для хранения образа Docker (см. настройку «LAMBDA_REGISTRY»);
- адреса небезопасных реестров (registry), к которым может понадобиться доступ во время создания lambda (см. настройку «LAMBDA_INSECURE_REGISTRIES»).

См. подробную информацию в разделе «Configuration requirements» руководства разработчика сервиса Lambda.

6.17.1.4 Настройка сущностей lambda

Для запущенных сущностей lambda доступен определенный набор настроек. Их можно задать в Configurator, отфильтровав настройки по имени «luna-lambda-unit».

> Настройки для сервиса Lambda и настройки для сущностей lambda отличаются.

Настройки содержат адреса сервисов, тайм-ауты подключения, настройки логирования и др., позволяющие сущностям lambda эффективно взаимодействовать с сервисами LUNA PLATFORM. См. все доступные настройки в разделе «Настройки lambda».

> Настройки распространяются на все lambda одновременно.

6.17.2 Типы lambda

Lambda может быть трех типов:

- Тип **Handlers-lambda**, предназначенный для замены функционала классического обработчика.
- Тип **Standalone-lambda**, предназначенный для реализации самостоятельного функционала для выполнения тесной интеграции с LUNA PLATFORM.
- Тип **Tasks-lambda**, предназначенный для реализации дополнительных пользовательских длинных типов задач.

У каждого типа есть определенные требования к сервисам LUNA PLATFORM, фактические настройки которых будут автоматически использоваться для обработки запросов. Перед началом работы пользователь должен определиться какая lambda ему необходима.

6.17.2.1 Handlers-lambda

Примеры возможного функционала:

- выполнение верификации с возможностью сохранения какого-либо события;

- сравнение двух изображений без выполнения остального функционала классического обработчика;
- добавление собственной логики фильтрации в функционал сравнения;
- обход определенных ограничений LUNA PLATFORM (например, указать количество максимальное кандидатов больше 100);
- встраивание нейронной сети SDK в обход LUNA PLATFORM.

Во время своей работы, Handlers-lambda будет взаимодействовать со следующими сервисами LUNA PLATFORM:

- Configurator — для получения настроек,
- Faces — для работы лицами и списками,
- Remote SDK — для выполнение детекций, эстимаций и извлечений,
- Events* — для работы с событиями,
- Python Matcher / Python Matcher Proxy** — для классического/перекрестного сравнения лиц или тел
- Image Store* — для хранения БО и исходных изображений лиц или тел

> Сервис Lambda не будет проверять подключение к отключенным сервисам и выдаст ошибку, если пользователь попытается сделать запрос к отключенному сервису.

> Для запуска сервиса Lambda требуется только наличие сервисов Configurator и Licenses.

* сервис может быть отключен в настройке «ADDITIONAL_SERVICES_USAGE»

** сервис по умолчанию отключен, для включения см. настройку «ADDITIONAL_SERVICES_USAGE»

Lambda-обработчик может использоваться в двух случаях:

- как пользовательский обработчик, который имеет свою собственную схему ответа, которая может отличаться от ответа классических обработчиков и не может быть должным образом использована в других сервисах LUNA PLATFORM
- как пользовательский обработчик, который имитирует ответ классического обработчика. К такому случаю предъявляются некоторые требования:
  - ответ должен соответствовать схеме ответа запроса на генерацию события;
  - обработчик должен правильно обрабатывать входящие данные, чтобы другие сервисы могли его использовать, в противном случае нет гарантии совместимости с другими сервисами. То есть если такой обработчик подразумевает распознавание лиц — модуль должен возвращать информацию о распознавании лиц в ответ, если обработчик подразумевает обнаружение тел — модуль должен возвращать обнаружение тел в ответ и т.д.

  Например, если Lambda-обработчик удовлетворяет вышеуказанным условиям, то его можно использовать в задаче Estimator в качестве классического обработчика.

См. подробную информацию и примеры кода для Handlers-lambda, в разделе «Handlers lambda development» руководства разработчика сервиса Lambda.

6.17.2.2 Standalone-lambda

Примеры возможного функционала:

- фильтрация входящих изображений по формату для последующей отправки в сервис Remote SDK;
- создание сервиса для отправки уведомлений по аналогии с сервисом Sender;
- создание сервиса для записи видеопотока и сохранения в виде видеофайла в сервис Image Store для последующей обработки приложением FaceStream.

Во время своей работы, Standalone-lambda будет взаимодействовать как минимум с сервисом Configurator, который позволяет lambda получать свои настройки (например, настройки логирования).

> Для запуска сервиса Lambda требуется только наличие сервисов Configurator и Licenses.

См. подробную информацию и пример кода для Standalone-lambda, в разделе «Standalone lambda development» руководства разработчика сервиса Lambda.

6.17.2.3 Tasks-lambda

Примеры возможного функционала:

- открепить лица от списка, если лица не похожи на указанное лицо
- удалить дубликаты лиц из списка
- рекурсивно найти события, похожие на указанное изображение

Во время своей работы, Tasks-lambda будет взаимодействовать со следующими сервисами LUNA PLATFORM:

- Configurator
- Faces
- Python Matcher
- Remote SDK
- Tasks
- Events*
- Image Store*
- Handlers*

* сервис может быть отключен в настройке «ADDITIONAL_SERVICES_USAGE»

После создания lambda необходимо выполнить запрос «lambda task» для создания задачи Lambda. Результаты задачи/подзадачи можно получить с помощью стандартных запросов сервиса Tasks.

Задача Lambda создается согласно общему процессу создания задач, **за исключением того**, что вместо «рабочего процесса» Tasks используется сервис Lambda.

При написании кода для Tasks-lambda рекомендуется разбивать каждую задачу на подзадачи для удобства представления процесса и распараллеливания выполнения задач.

> При необходимости можно создать расписание для Tasks-lambda.

См. подробную информацию и примеры кода для Tasks-lambda, в разделе «Tasks lambda development» руководства разработчика сервиса Lambda.

6.17.3 Создание lambda

Для создания lambda требуется выполнить следующие действия:

1. Написать код на языке Python в соответствии с типом будущей lambda и требованиями к коду.
2. Переместить файлы в архив в соответствии с требованиями к архиву.
3. Выполнить запрос «create lambda», указав следующие обязательные данные:

- «archive» — адрес до архива с пользовательским модулем;
- «credentials» > «lambda_name» — имя для создаваемой lambda;
- «parameters» > «lambda_type» — тип создаваемой lambda («handlers», «standalone» или «tasks»).

Также опционально в секции «deploy_parameters» можно задать количество подов, выделить ресурсы на поды, задать пространства имен (namespace), а также включить использование GPU.

Кроме того можно задать метки для запуска lambda с использованием конкретных узлов Kubernetes с помощью параметра «deploy_parameters» > «selector». Это позволяет более детально контролировать развертывание lambda, позволяя администраторам указывать узлы, на которых должны быть размещены lambda в зависимости от их потребностей в ресурсах. Применяя метки к узлам Kubernetes и используя параметр «selector», администраторы могут управлять выделением ресурсов не только для отдельных lambda, но и для групп lambda с аналогичными требованиями к ресурсам.

> При необходимости можно задать список дополнительных команд Docker для создания lambda-контейнера. См. раздел «Lambda — Archive requirements» сервиса Lambda руководства разработчика.

В ответе на успешный запрос будет выдан идентификатор «lambda_id».

Создание lambda состоит из нескольких этапов, а именно:

- создание образа Docker:
  - получение предоставленного ZIP-архива;

  - дополнение архива необходимыми файлами;
  - сохранение архива в хранилище S3;
  - публикация образа в реестр (registry).
- создание сервиса в кластере Kubernetes.

Во время создания lambda можно выполнять следующие запросы:

- «get lambda image creation status» для получения статуса создания образа Docker («in_-progress», «error», «completed», «not_found»)
- «get lambda image creation logs» для получения логов создания образа Docker
- «get lambda status» для получения статуса создания lambda («running», «waiting», «terminated», «not_found»)

> См. диаграмму последовательности создания lambda в разделе «Диаграмма создания lambda».

> См. подробное описание процесса создания lambda в разделе «Creation pipeline» руководства разработчика сервиса Lambda.

6.17.4 Создание обработчика для Handlers-lambda

Если предполагается использовать lambda как пользовательский обработчик, имитирующий ответ классического обработчика, то необходимо создать обработчик, указав «handler_type» = «2» и полученный «lambda_id».

Во время создания обработчика, сервис Handlers выполнит проверку работоспособности (healthcheck) до кластера Kubernetes.

Полученный «handler_id» может быть использован в запросах «generate events» или «estimator task».

6.17.5 Использование lambda

В таблице ниже приведены ресурсы для работы с lambda в зависимости от её типа.

Ресурс	Тип lambda	Формат тела запроса и ответа
«/lambdas/{lambda_id}/proxy»	Standalone-lambda, Handlers-lambda с собственной схемой ответа	Собственные
«/handlers/{handler_id}/events»	Handlers-lambda	Соответствующие спецификации OpenAPI

Ресурс	Тип lambda	Формат тела запроса и ответа
«/tasks/estimator»	Handlers-lambda	Соответствующие спецификации OpenAPI

> См. диаграмму последовательности обработки lambda в разделе «Диаграмма обработки lambda».

Каждая lambda имеет свой собственный API, описание которого также доступно с помощью запроса «get lambda openapi documentation».

Каждый ответ lambda будет содержать несколько заголовков, включая:

- «Luna-Request-Id» — классический внешний идентификатор запроса LUNA PLATFORM;
- «Lambda-Version» — содержит текущую лямбда-версию.

Полезные запросы при работе с lambda:

- «get lambda status» для получения статуса создания lambda («running», «waiting», «terminated», «not_found», «pending»)
- «get lambda» для получения полной информации о созданной lambda (время создания, имя, статус и пр.)
- «get lambda logs» для получения логов создания lambda

6.17.6 Создание lambda с поддержкой GPU

Доступна возможность создания lambda, которая может использовать ресурсы графического процессора. Для возможности использования GPU, в запросе на создание lambda должен быть включен параметр «deploy_parameters» > «enable_gpu».

Lambda поддерживает только графические процессоры NVIDIA. Для получения дополнительной информации об использовании графических процессоров см. документацию по Kubernetes.

Если доступен только один графический процессор, но требуется больше, можно включить общий доступ к графическому процессору (см. официальную документацию).

> Сервис Lambda не управляет ресурсами кластера, включая выделение графических процессоров (GPU). Управление ресурсами осуществляется администратором Kubernetes.

Кроме того, при разворачивании LUNA PLATFORM в Kubernetes рекомендуется настроить манифест определенным образом для всех контейнеров, использующих GPU (включая контейнер сервиса Lambda) во избежание проблем с видимостью устройств. См. подробную информацию в разделе «Настройка манифеста для поддержки GPU» руководства по установке.

6.17.7 Обновление lambda

Базовый образ для контейнеров lambda периодически обновляется. Этот образ содержит необходимые модули для взаимодействия с сервисами LUNA PLATFORM. Для применения обновлений требуется пересоздать lambda, чтобы контейнер мог быть перестроен на основе нового образа. После обновления базового образа и библиотеки «luna-lambda-tools», функциональность lambda может быть нарушена.

Можно обновить lambda с помощью запроса «update lambda», используя последний базовый образ. Рекомендуется иметь резервную копию архива на S3.

См. подробную информацию про механизм обновления, стратегии резервного копирования и восстановление из резервной копии в разделе «Lambda updates» руководства разработчика сервиса Lambda.

6.18 Backport 3

Сервис Backport 3 используется для обработки запросов LUNA PLATFORM 3 с помощью LUNA PLATFORM 5.

Несмотря на то, что большинство запросов выполняется так же, как в LUNA PLATFORM 3, имеются ограничения. Более подробная информация приведена в разделе «Функции и ограничения Backport 3».

Более подробная информация об API Backport 3 приведена в «Backport3ReferenceManual.html».

6.18.1 Новые ресурсы Backport 3

6.18.1.1 Оценка Liveness

Backport 3 выполняет оценку Liveness в дополнение к функциям LUNA PLATFORM 3. См. раздел «liveness > predict liveness» в спецификации OpenAPI сервиса .

6.18.1.2 Обработчики

В сервисе Backport 3 предусмотрено несколько обработчиков: *extractor*, *identify*, *verify*. С помощью этих обработчиков можно выполнить несколько действий в одном запросе:

- «handlers» > «face extractor» – можно извлечь БШ из изображения, создать персону с этим БШ, добавить персону в заранее определенный список.
- «handlers» > «identify face» – можно извлечь БШ из изображения и сравнить БШ с предопределенным списком кандидатов.
- «handlers» > «verify face» – можно извлечь БШ из изображения и сравнить БШ с БШ персоны.

> См. описание персон в документации LUNA PLATFORM 3.

Описание обработчиков и все параметры обработчиков приведены в следующих разделах:

- «handlers» > «patch extractor handler»
- «handlers» > «patch verify handler»
- «handlers» > «patch identify handler»

Запросы основаны на обработчиках и в отличие от стандартных запросов «descriptors» > «extract descriptors», «matching» > «identification», и «matching» > verification», приведенные выше запросы являются более гибкими.

Можно изменить уже существующие обработчики, добавив оценку дополнительных параметров в запросы. Например, можно указать пороговые значения углов положения головы или включить-/выключить оценку базовых атрибутов.

Обработчики создаются для каждой новой учетной записи в момент ее создания. Созданные обработчики содержат параметры по умолчанию.

У каждого обработчика есть соответствующий обработчик в сервисе Handlers. Параметры обработчиков хранятся в базе данных *luna_backport3*.

Каждый обработчик поддерживает запросы *GET* и *PATCH*, поэтому можно получать и обновлять параметры всех обработчиков.

У каждого обработчика есть своя версия. Версия увеличивается с каждым *PATCH* запросом. При удалении текущего обработчика, версия сбрасывается до *1*:

- Для запросов с методами *POST* и *GET*:

  Если в сервисе Handlers и/или Backport 3 нет обработчика для указанного действия, он создается с параметрами по умолчанию.

- Для запросов с *PATCH* методами:

  Если в сервисе Handlers и/или Backport 3 нет обработчика для указанного действия, создается новый обработчик включающий набор политик по умолчанию и набор политик из запроса.

6.18.2 Архитектура Backport 3

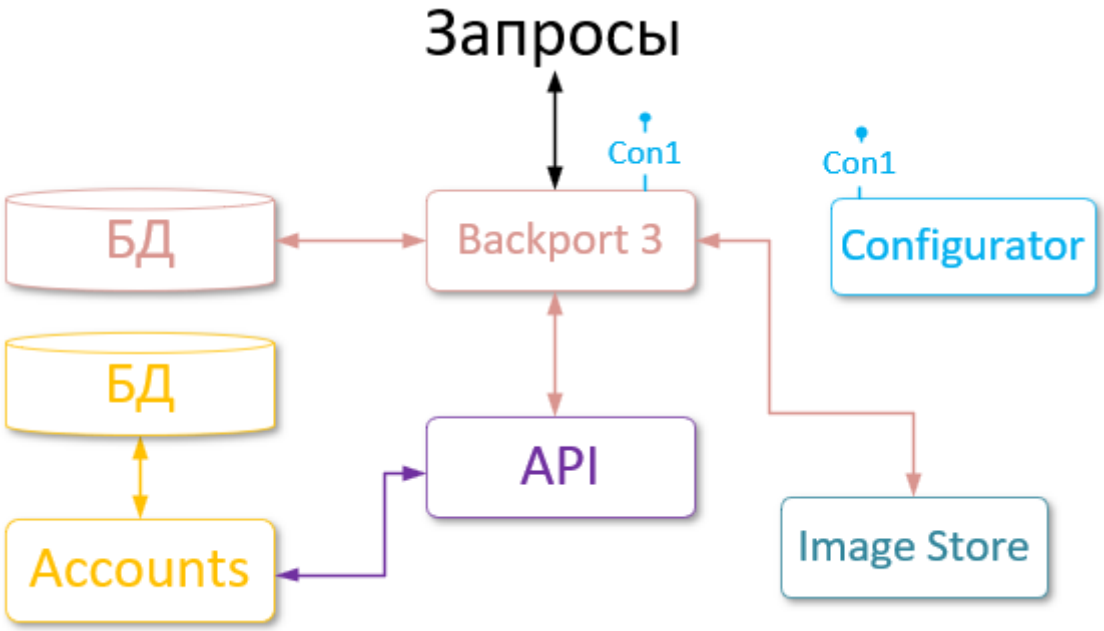


Рис. 43: Взаимодействие сервисов Backport 3 и LP 5

Сервис Backport 3 взаимодействует с сервисом API и с помощью него отправляет запросы в LUNA PLATFORM 5. В свою очередь сервис API взаимодействует с сервисом Accounts для проверки аутентификационных данных.

У сервиса Backport 3 есть своя собственная база данных (см. «Описание базы данных Backport3»). Некоторые из его таблиц приравниваются к таблицам базы данных Faces LP 3. Это позволяет создавать и использовать такие же сущности (лица, токены аккаунтов и аккаунты), которые существовали в LP 3.

Сервис Backport использует Image Store для хранения портретов (см. описание портретов в документации LUNA PLATFORM 3).

Настроить Backport 3 можно с помощью сервиса Configurator.

6.18.3 Функции и ограничения Backport 3

Следующие функции имеют основные отличия:

Для следующих ресурсов при использовании метода POST по умолчанию задана 56 версия биометрических шаблонов:

- `/storage/descriptors`
- `/handlers/extractor`
- `/handlers/verify`
- `/handlers/identify`
- `/matching/search`

  Тем не менее, можно загрузить существующие БШ версий 52, 54, 56. Более старые версии БШ больше не поддерживаются.

- Оценка параметра *saturation* больше не поддерживается для ресурса */storage/descriptors* при отправке запроса с методом *POST*, значение всегда устанавливается равным 1.
- Для ресурса */storage/descriptors* при отправке запроса с методом *POST* оценка атрибута *eyeglasses* больше не поддерживается. В структуре *attributes* в ответе не будет параметра *eyeglasses*.
- Для ресурса */storage/descriptors* при отправке запроса с методом *POST* пороговые значения угла положения головы по-прежнему можно отправлять в качестве плавающих значений в диапазоне [0, 180], но они будут автоматически округлены до целых значений. Как и раньше, пороги вне диапазона [0, 180] не учитываются.

6.18.4 Модуль сбора мусора (Garbage collection)

Согласно логике LUNA Platform 3, мусор – это биометрические шаблоны, которые не связаны ни с персонами, ни со списками.

Для нормальной работы системы необходимо регулярно удалять мусор из баз. Для этого требуется запустить скрипт очистки системы `remove_not_linked_descriptors.py` из директории ./base_scripts/gc/.

Согласно архитектуре Backport 3, с помощью этого скрипта из сервиса Faces удаляются лица, которые не связаны с какими-либо персонами или списками из базы данных Luna Backport 3.

6.18.4.1 Процесс выполнения скрипта

Процесс выполнения скрипта состоит из нескольких этапов:

1) В базе данных Faces создается временная таблица. См. дополнительную информацию о временных таблицах для oracle или postgres.
2) Получаются ID лиц, не привязанных к спискам. ID хранятся во временной таблице.
3) Пока временная таблица содержит данные, выполняются следующие операции:
   - Получается пакет ID из временной таблицы. Получаются первые 10 тыс. (или меньше) идентификаторов лиц.
   - Получаются `filtered ids` – идентификаторы, которых нет в таблице `person_face` базы данных Backport 3.
   - `Filtered ids` удаляются из базы данных Faces. Если некоторые лица не удается удалить, выполнение скрипта останавливается.
   - `Filtered ids` удаляются из базы данных Backport 3 (foolcheck). Выводится предупреждение.
   - ID удаляются из временной таблицы.

6.18.4.2 Запуск скрипта GC

```
docker run --rm -t --network=host --entrypoint bash dockerhub.visionlabs.ru/
    luna/luna-backport-3:v.0.11.48 -c "python3 ./base_scripts/gc/
    remove_not_linked_descriptors.py"
```

Результат будет содержать информацию о количестве удаленных лиц и количестве людей с лицами.

6.19 Backport 4

Сервис Backport 4 используется для обработки запросов LUNA PLATFORM 4 с помощью LUNA PLATFORM 5.

Несмотря на то, что большинство запросов выполняется так же, как в LUNA PLATFORM 4, все же есть некоторые ограничения. Более подробная информация приведена в разделе «Функции и ограничения Backport 4».

Более подробна информация о сервисе API Backport 4 приведена в спецификации OpenAPI сервиса Backport 4.

6.19.1 Архитектура Backport 4

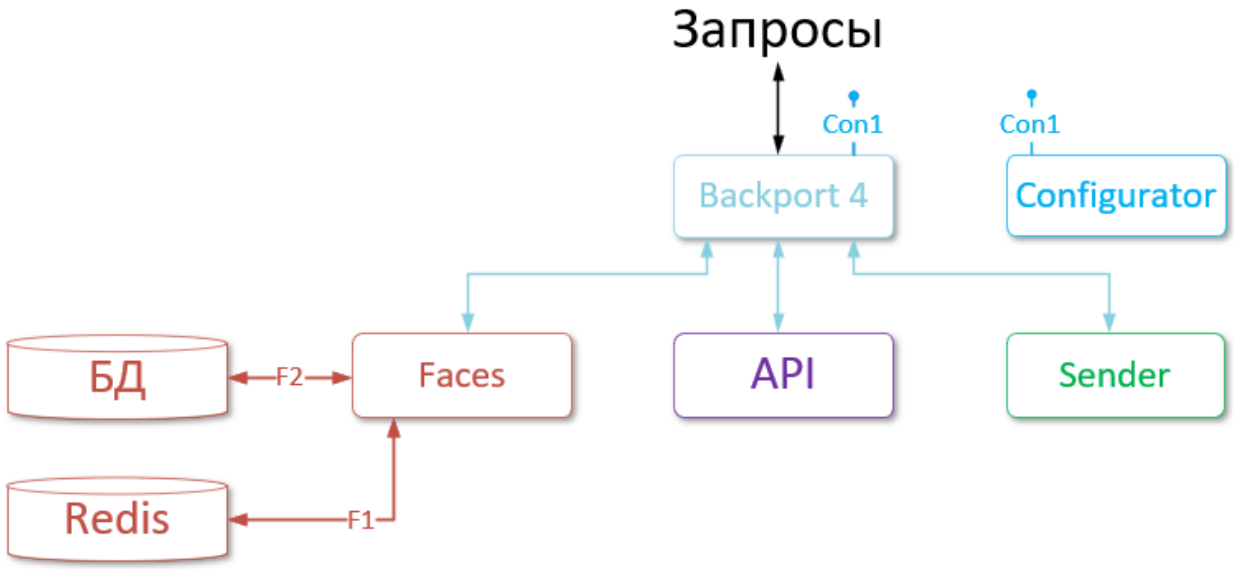


Рис. 44: Взаимодействие сервисов Backport 4 и LP 5

Сервис Backport 4 взаимодействует с сервисом API и с помощью него отправляет запросы в LUNA PLATFORM 5.

Backport 4 напрямую взаимодействует с сервисом Faces для получения информации о количестве существующих атрибутов.

Backport 4 напрямую взаимодействует с сервисом Sender. Все запросы к сервису Sender отправляются с помощью сервиса Backport 4. См. запрос «ws» > «ws handshake» в спецификации OpenAPI сервиса Backport 4.

Настроить Backport 4 можно с помощью сервиса Configurator.

6.19.2 Функции и ограничения Backport 4

Следующие функции имеют основные отличия:

Текущие версии сервисов LUNA PLATFORM выдаются по запросу к ресурсу /version. Например, выдаются версии следующих сервисов:

- «luna-faces»
- «luna-events»
- «luna-image-store»
- «luna-python-matcher» или «luna-matcher-proxy»
- «luna-tasks»
- «luna-handlers»
- «luna-api»
- «LUNA PLATFORM»
- «luna-backport4» – текущая версия

Журнал изменений ресурсов:

- Ресурс `/attributes/count` доступен без каких-либо параметров запроса и не поддерживает аккаунтинг. Ресурс работает с временными атрибутами.
- Ресурс `/attributes` по методу GET: допускается параметр запроса `attribute_ids` вместо параметров запроса `page`, `page_size`, `time__lt` и `time__gte`. Таким образом, можно получать атрибуты по их идентификаторам, а не по фильтрам. Ресурс работает с временными атрибутами.
- Ресурс `/attributes/<attribute_id>` по методам GET, HEAD, DELETE и ресурс `/attributes/<attribute_id>/samples` по методу GET взаимодействуют с временными атрибутами и выдают данные атрибутов, если срок действия атрибута не истек. В противном случае выдается ошибка «Not found».
- Если уже использован атрибут для создания лица, необходимо использовать `face_id` для получения данных атрибута. В этом случае `attribute_id` из запроса эквивалентен `face_id`.
- С помощью ресурса `/faces` можно создавать несколько лиц с одним и тем же `attribute_id` .
- С помощью ресурса `/faces/<face_id>` по методу DELETE можно удалить лицо без удаления его атрибута.
- С помощью ресурса `/faces/<face_id>` по методу PATCH можно изменить атрибут лица, сделав первый запрос на изменение `event_id`, `external_id`, `user_data`, `avatar` (при необходимости) и второй запрос на изменение атрибута (при необходимости).
- Если необходимо изменить `attribute_id` лица, сервис попытается изменить его с помощью данных временного атрибута, если временный атрибут существует. Или же сервис попытается изменить его с помощью данных атрибутов лица с `face_id` = `attribute_id`.
- Политика сравнения ресурсов `/handlers` теперь имеет ограничение на сравнение по

умолчанию, которое задано с помощью настройки `MATCH_LIMIT` из файла config.py сервиса Backport 4.

- Ресурс `/events/stats` по методу POST: использование `attribute_id` в объекте `filters` запрещено, так как это поле больше не хранится в базе данных. Будет выдаваться ответ со статус кодом 403.
- `Attribute_id` в событиях не пустой и равен `face_id` для обратной совместимости. Задача Garbage collection недоступна, поскольку все атрибуты временны и будут удаляются автоматически. По запросу к ресурсу `/tasks/gc` выдается статус код 400.
- Столбец `attribute_id` не добавляется в задачу Reporter, а также этот столбец игнорируется, если он указан в запросе. Столбцы `top_similar_face_id`, `top_similar_face_list`, `top_similar_face_similarity` заменяются столбцом top_match в отчете, если какой-либо из этих столбцов передается в запросе задачи Reporter.
- В результате выполнения задачи Linker всегда создаются новые лица из событий и игнорируются лица, созданные при запросе обработки событий.
- Ресурс `/matcher` не проверяет наличие указанных лиц, поэтому ошибка `FacesNotFound` никогда не выдается. Если пользователь укажет несуществующего кандидата типа «faces», сообщения об ошибке не будет, и не будет выполнено фактическое сравнение с этим лицом.
- Ресурс `/matcher` проверяет, есть ли у эталона с атрибутом типа идентификатор атрибута лица или идентификатор временного атрибута, и выполняет подстановку типа. Следовательно, он обеспечивает отправку эталонов для сравнения таким же способом, как это было сделано в предыдущей версии.
- Ресурс `/matcher` учитывает ограничения сравнения. По умолчанию максимальное количество эталонов или кандидатов ограничено 30. Если необходимо превысить эти ограничения, следует настроить параметры `REFERENCE_LIMIT` и `CANDIDATES_LIMIT`.
- Добавлен ресурс `/ws`. В LUNA PLATFORM 4 API не было ресурса `/ws`, поскольку это был отдельный ресурс сервиса Sender. Этот добавленный ресурс аналогичен ресурсу сервиса Sender, за исключением того, что `attribute_id` лиц кандидатов аналогичен `face_id`.
- Ресурс `/handlers` выдает ошибку «Invalid handler with id {handler_id}», если обработчик был создан в LUNA PLATFORM 5 API и не поддерживается в LUNA Backport 4.

6.20 Пользовательский интерфейс Backport 4

Сервис пользовательского интерфейса используется для визуального представления функций LP. Он не содержит все функции, имеющиеся в LP. С помощью пользовательского интерфейса можно:

- Загружать фотографии и создавать лица с их помощью;
- Создавать списки;
- Сравнивать имеющиеся лица;
- Показать имеющиеся события;
- Показать имеющиеся обработчики.

Вся информация в пользовательском интерфейсе отображается в соответствии с данными аккаунта, указанными в файле конфигурации сервиса пользовательского интерфейса (./luna-ui/browser/env.js).

> Пользовательский интерфейс работает одновременно только с одним аккаунтом, который должен иметь тип «user».

Необходимо открыть браузер и ввести адрес пользовательского интерфейса. Значение по умолчанию: <server_url>:4200.

Можно выбрать страницу в левой части окна.

6.20.0.1 Страница списков/лиц

Стартовая страница пользовательского интерфейса — **Lists/Faces**. На ней есть все лица и списки, созданные с помощью указанного в конфигурации аккаунта.

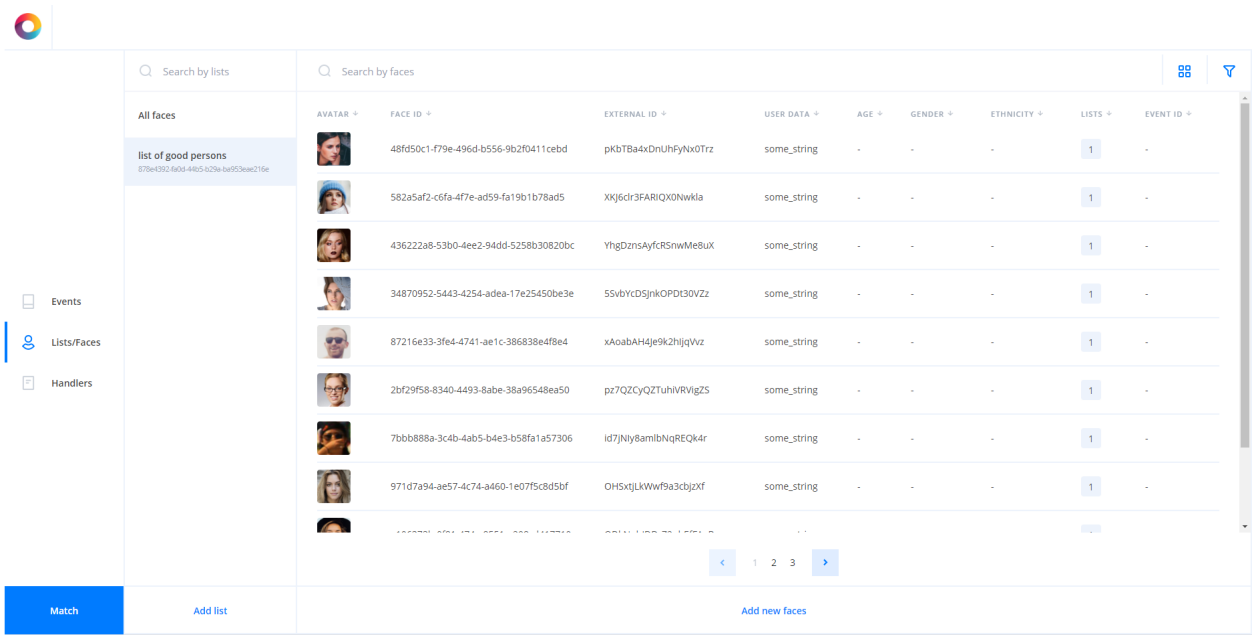


Рис. 45: Страница списков/лиц

В левом столбце рабочей области отображаются имеющиеся списки. Можно создать новый список, нажав кнопку **Add list**. В появившемся окне можно указать данные пользователя для списка.

В правом столбце показаны все созданные лица с разбивкой на страницы.

С помощью кнопки **Add new faces** можно создать новые лица.

На первом этапе необходимо выбрать фотографии, из которых будут созданы лица. Можно выбрать одно или несколько изображений с одним или несколькими лицами на них.

После выбора изображений в новом диалоговом окне будут показаны все найденные лица.

Все правильно предварительно обработанные изображения будут иметь отметку Done. Если изображение не соответствует ни одному из требований, для него будет отображаться ошибка.

Доступна кнопка **Next step**.

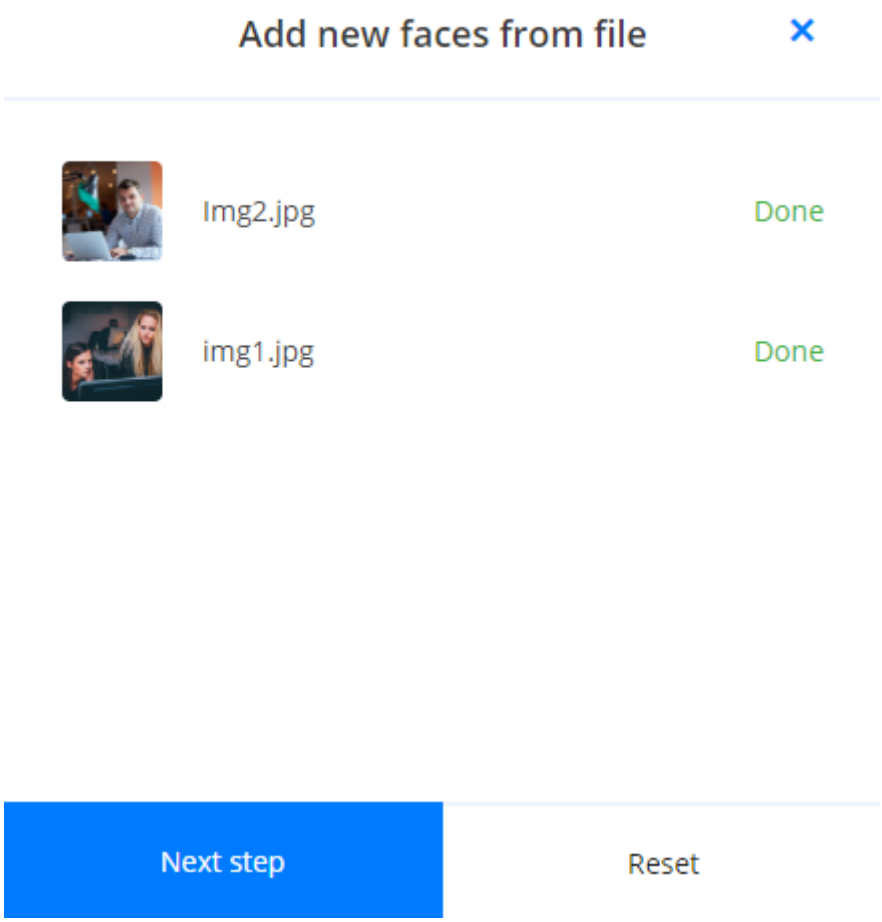


Рис. 46: Выбрать изображения

На следующем этапе необходимо выбрать атрибуты, которые нужно извлечь для лиц.

Для этого нужно нажать кнопку **Next step**.

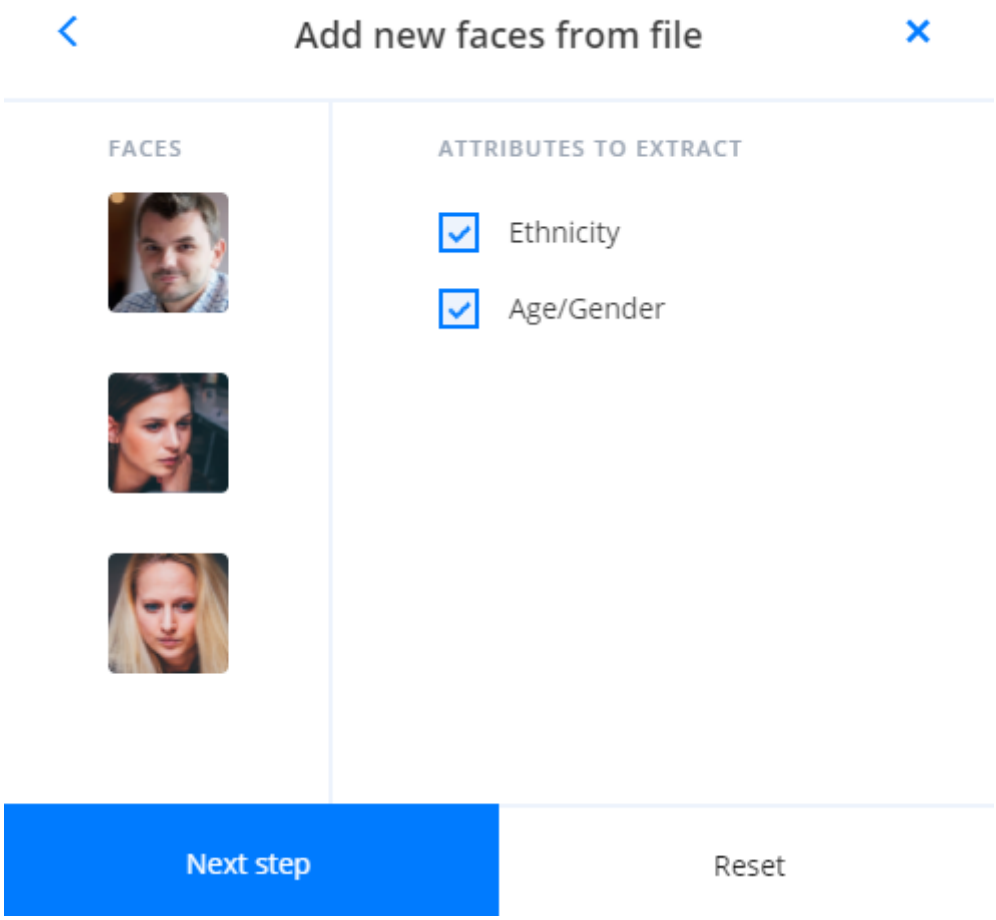


Рис. 47: Выбрать атрибуты

На следующем этапе необходимо указать данные пользователя и внешний идентификатор (external_id) для каждого лица. Также можно выбрать списки, в которые будет добавлено каждое лицо. Для создания лиц нужно нажать кнопку **Add Faces**.

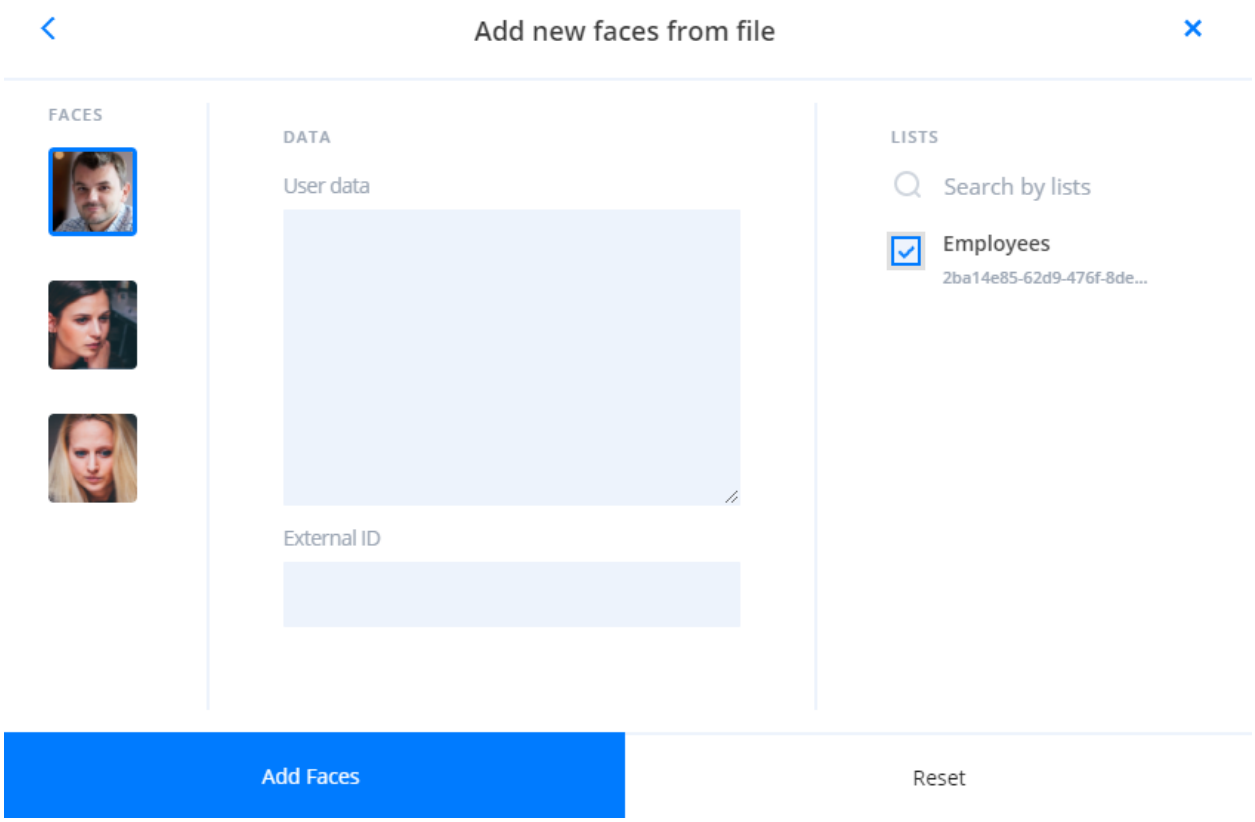
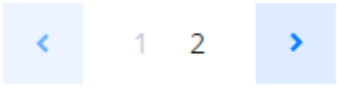


Рис. 48: Добавить данные пользователя, внешний идентификатор и указать списки

Страницы можно менять с помощью кнопок со стрелками.



Можно изменить отображение лиц и отфильтровать их с помощью кнопок в правом верхнем углу.

Фильтр лиц. Можно фильтровать лица по идентификатору, внешнему идентификатору или идентификатору списка;

/ Изменение вида имеющихся лиц.

6.20.0.2 Страница сервиса Handlers

На странице Handlers отображаются все обработчики, созданные с помощью указанного в конфигурации аккаунта.

Вся информация об указанных политиках обработчика отображается при выборе обработчика.

Можно отредактировать или удалить обработчик с помощью значков редактирования и удаления .

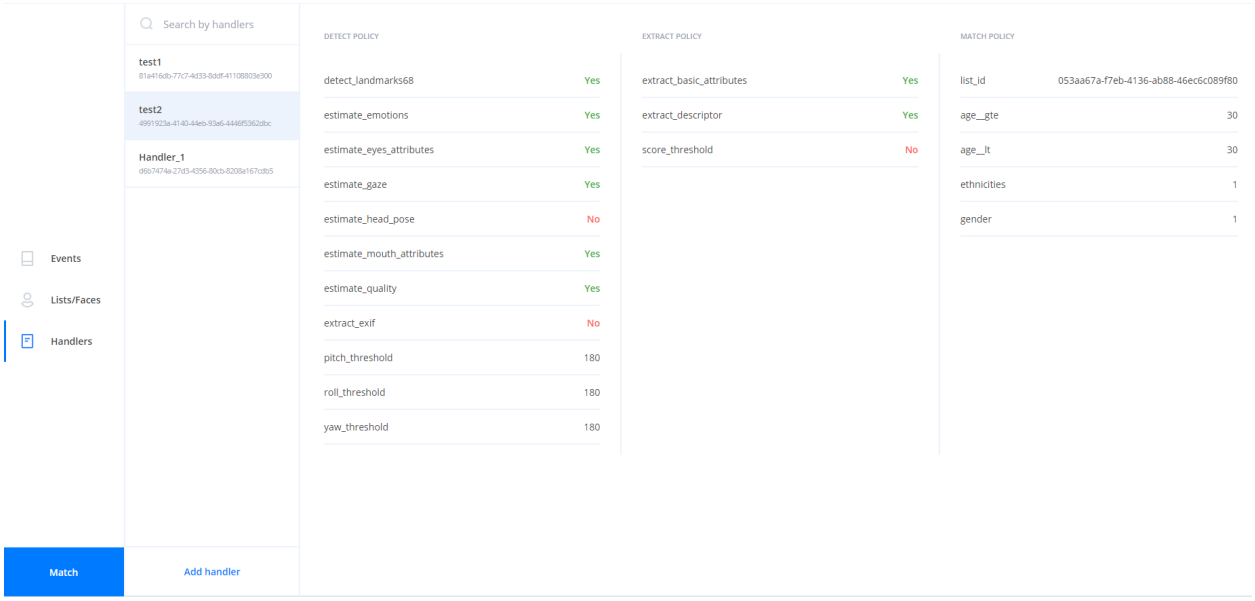


Рис. 49: Страница сервиса Handlers

6.20.0.3 Страница сервиса Events

На странице Events отображаются все события, созданные с помощью указанного в конфигурации аккаунта.

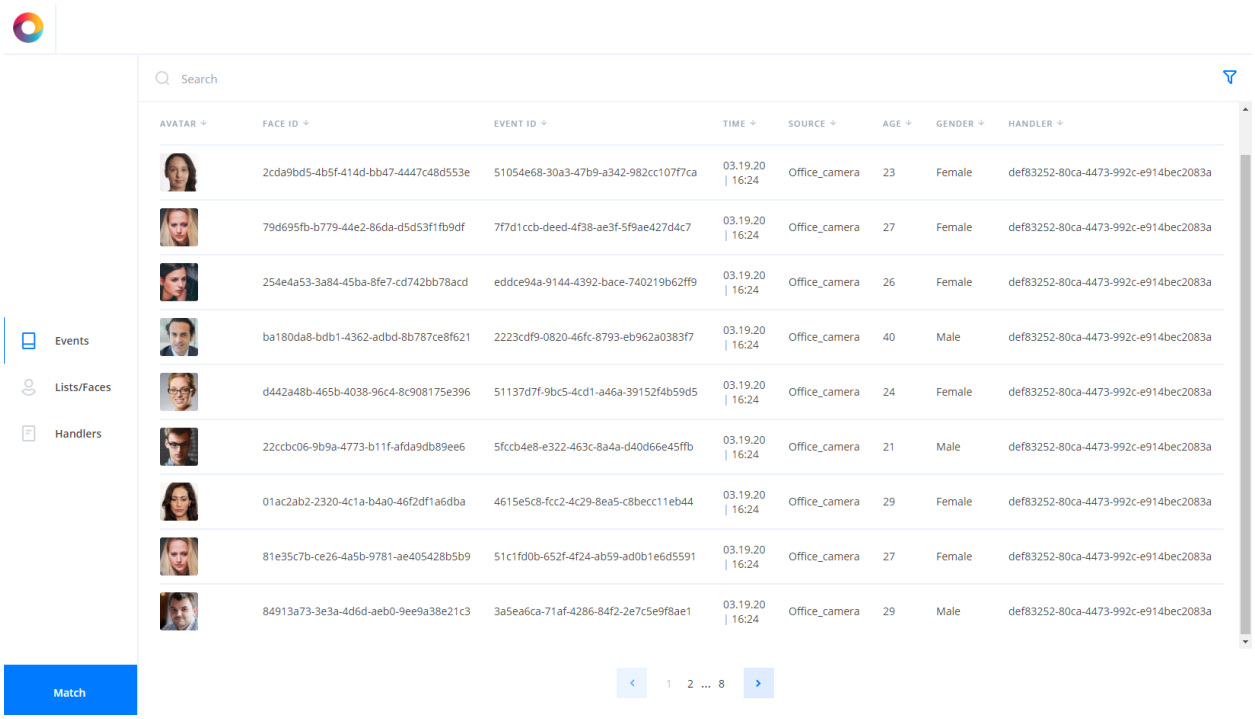


Рис. 50: Страница сервиса Events

Также на ней есть фильтры для отображения событий .

6.20.1 Общая информация

Можно отредактировать или удалить элемент (лицо, список или обработчик) с помощью соответствующих значков и . Значки появляются при наведении курсора на элемент.

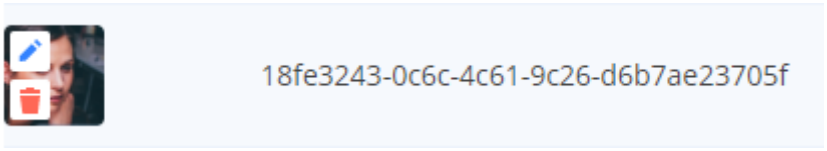


Рис. 51: Иконки для элемента

6.20.2 Диалог сравнения

С помощью кнопки **Matching** в нижнем левом углу окна можно выполнить сравнение.

После нажатия на кнопку можно выбрать количество полученных результатов для каждого эталона.

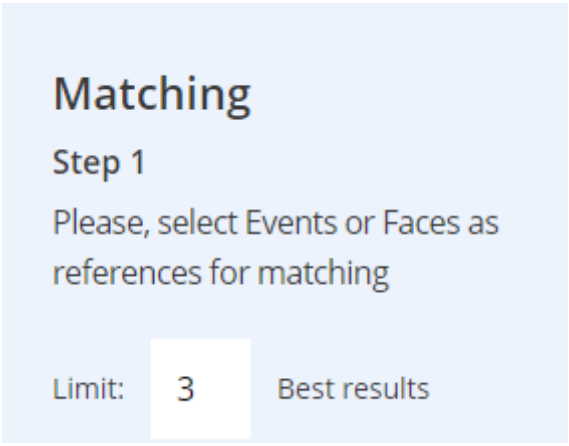


Рис. 52: Выбор количества результатов

На первом этапе необходимо выбрать эталоны для сравнения. Можно выбрать лица и/или события в качестве эталонов.

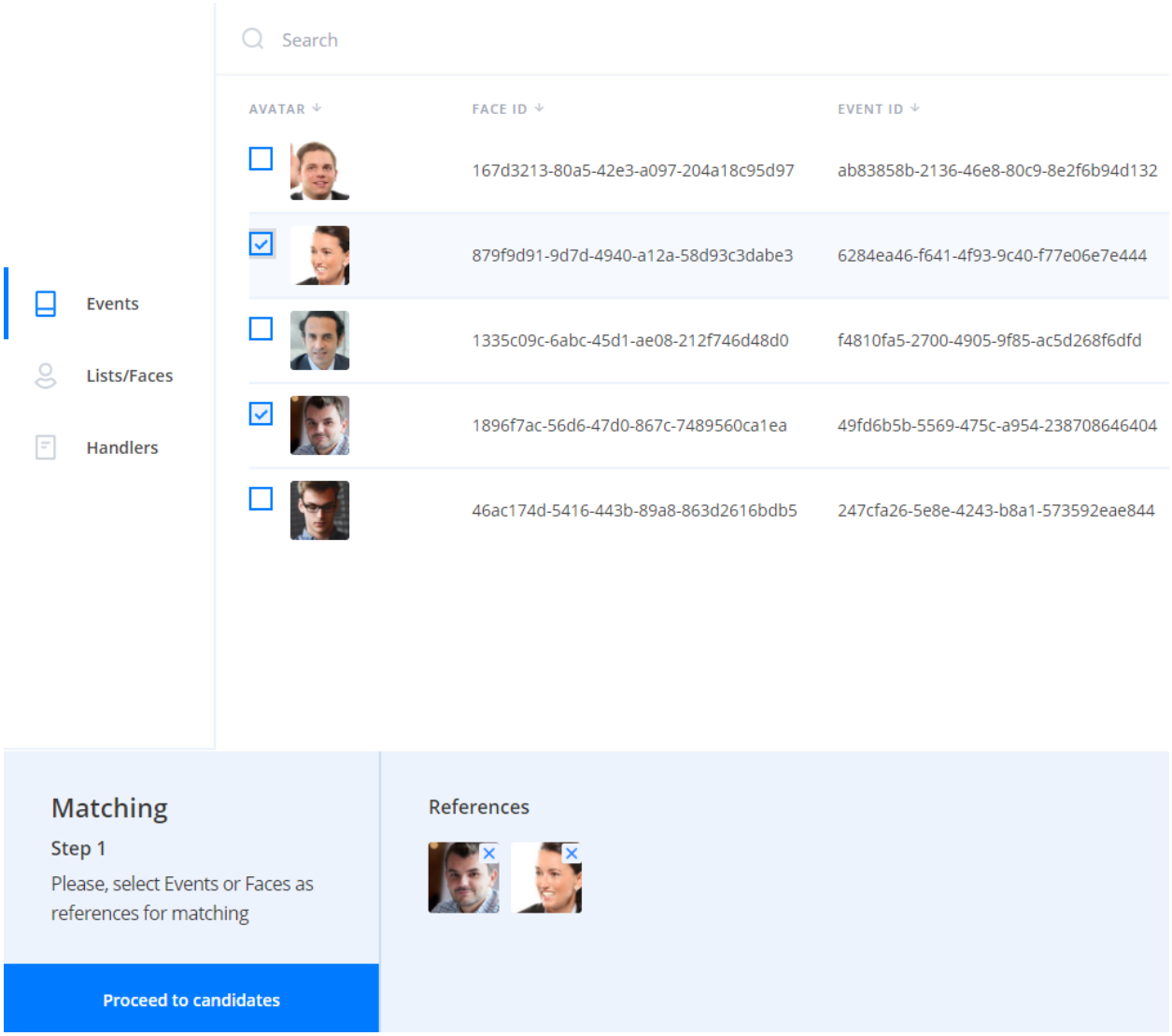


Рис. 53: Выбор эталонов

На втором этапе необходимо выбрать кандидатов для сравнения. Можно выбрать лица или списки в качестве кандидатов.

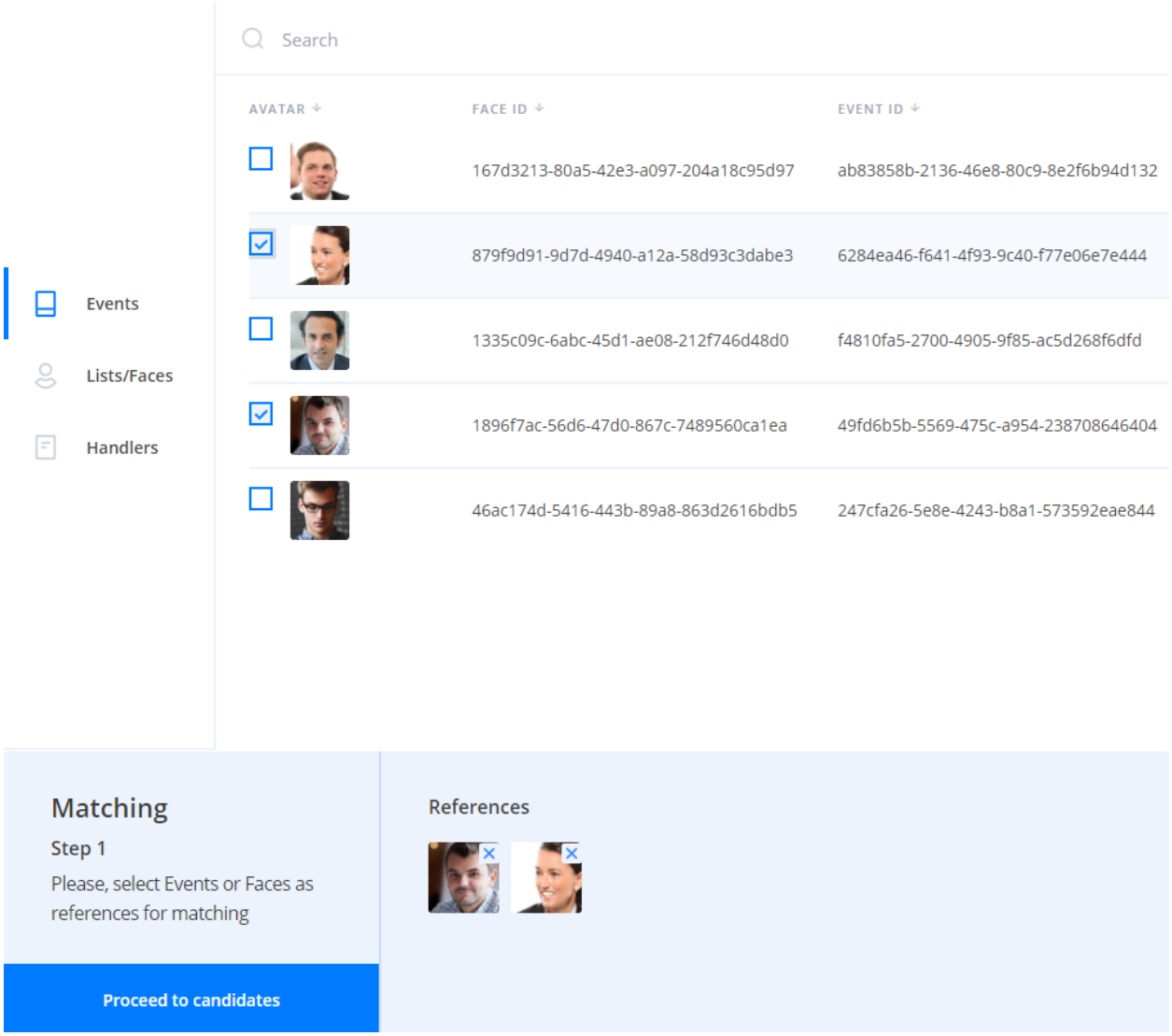


Рис. 54: Выбор кандидатов

На последнем этапе необходимо нажать кнопку **Start matching**, чтобы получить результаты.

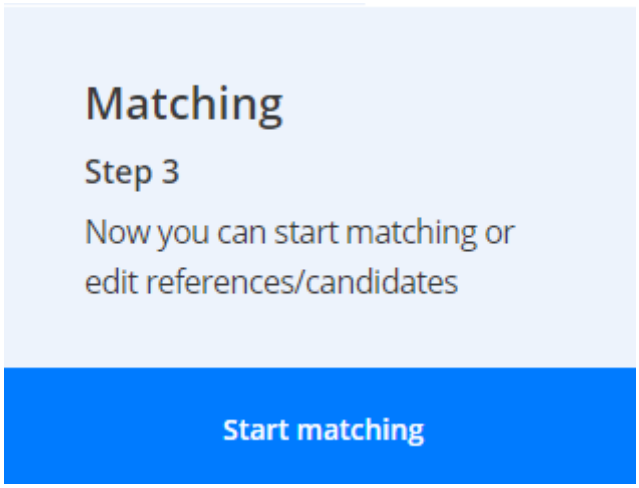


Рис. 55: Начало сравнения

6.21 Потребление ресурсов сервисами

Ниже приведена таблица, описывающая наиболее значительное потребление ресурсов сервисами. Сервисы Remote SDK и Python Matcher выполняют наиболее ресурсозатратные операции.

Сервис	Наиболее потребляемые ресурсы
Remote SDK	CPU, RAM, GPU Сервис Remote SDK выполняет математическое преобразование изображений и извлечение биометрических шаблонов. Эти операции требуют значительных вычислительных мощностей. Для вычислений можно использовать как CPU, так и GPU. Использование GPU предпочтительно, т.к. обработка запросов происходит эффективнее. Однако, поддерживаются не все типы видеокарт.
Python Matcher	CPU, RAM Python Matcher выполняет сравнение по спискам. Сравнение требует ресурсов CPU, однако также следует выделить максимально возможный объем RAM под каждого экземпляра Python Matcher. RAM используется для хранения биометрических шаблонов, полученных из базы данных. Таким образом, сервису Python Matcher не требуется запрашивать каждый БШ из базы данных. При распределении экземпляров на нескольких серверах следует учитывать производительность каждого сервера. Например, если крупная задача выполняется несколькими экземплярами Python Matcher, а один из них находится на сервере с низкой производительностью, выполнение всей задачи в целом может замедлиться.
Postgres	SSD, CPU, RAM
Image Store	SSD, CPU, RAM
Handlers	CPU
Tasks	RAM
API	CPU
Faces	CPU
Events	CPU
Backport 3	CPU
Backport 4	CPU

Сервис	Наиболее потребляемые ресурсы
Lambda	RAM, CPU Количество RAM зависит от размера передаваемого архива. См. раздел «Сервис Lambda».
Sender, Admin, Accounts, Licenses	В обычных сценариях не должны потреблять много ресурсов.

7 Дополнительная информация

7.1 Описание OneShotLiveness

Технология Liveness позволяет обнаруживать атаки на биометрическое предъявление. Такая атака означает ситуацию, когда злоумышленник пытается использовать видео или фото другого человека для того, чтобы обойти систему распознавания и получить доступ к личным данным этого человека.

Для оценки Liveness в LUNA PLATFORM, используется эстиматор LUNA SDK OneShotLiveness.

Атаки на биометрическое предъявление бывают следующих основных видов:

- Атака с использованием распечатанного фото. Используются одно или несколько изображений другого человека.
- Атака с воспроизведением видео. Используется видеоряд с другим человеком.
- Атака с распечатанной бумажной маской. Злоумышленник закрывает свое лицо изображением лица другого человека, вырезанным из фото.
- Атака с 3D-маской. Злоумышленник надевает 3D-маску с изображением лица другого человека.

Для Liveness доступна возможность лицензировать количество выполненных транзакций. Также можно выбрать между безлимитной лицензией и лицензией с ограниченным числом транзакций (Максимально в ключе может быть задано 16 777 215 транзакций). После исчерпания лимита транзакций будет невозможно использовать оценку Liveness в запросах. На запросы не использующие Liveness или с отключённой оценкой Liveness исчерпание лимита не влияет, они продолжают работать в обычном режиме.

7.1.1 Результаты проверки Liveness

Алгоритм Liveness использует одно изображение для обработки и возвращает следующие данные:

- Liveness probability (вероятностная оценка Liveness) [0..1]. Здесь 1 означает реального человека, 0 — подмена. Этот параметр показывает вероятность того, что на изображении присутствует живой человек, то есть это не атака на биометрическое предъявление. В целом, расчетная вероятность должна превышать порог Liveness.
- Prediction (вердикт). На основе вышеперечисленных данных LUNA PLATFORM выдаёт следующие вердикты:
  - 0 (spoof). Проверка выявила, что человек не является реальным.
  - 1 (real). Проверка выявила, что человек является реальным.

7.1.2 Запросы для проверки Liveness

Liveness используется в следующих ресурсах:

- «/liveness»,
- «/sdk»,
- «/handlers»,
- «/verifiers».

События можно фильтровать по результатам оценки Liveness в ресурсах «handlers/{handler_id}/events» и «/verifiers/{verifier_id}/verifications», т.е. можно исключить результаты «spoof» или «real» из обработки изображений.

Фильтрация по результатам оценки Liveness доступна в следующих сценариях:

- при выполнении операций сравнения;
- при выполнении задач;
- при отправке данных через веб-сокет.

Можно также задать параметр оценки Liveness при создании и сохранении событий вручную в ресурсе «handlers/{handler_id}/events/raw».

Для многократно загружаемых изображений можно агрегировать результаты по результатам оценки Liveness для получения более точных данных.

7.1.2.1 Требования Liveness

Ниже перечисленные требования к обрабатываемому изображению:

Параметры	Требования
Минимальное разрешение для мобильных устройств	720x960 пикселей
Максимальное разрешение для мобильных устройств	1080x1920 пикселей
Минимальное разрешение для веб-камер	1280x720 пикселей
Максимальное разрешение для веб-камер	1920x1080 пикселей
Сжатие	Нет
Биометрический образец	Нет
Обрезка изображения	Нет
Наложение эффектов	Нет

Параметры	Требования
Маска	Нет
Количество лиц в кадре	1
Ширина bbox детекции лица	Более 200 пикселей
Отступ от краев кадра	Более 10 пикселей
Положение головы	от -20 до +20 градусов для наклона головы, рыскания и крена
Качество изображения	Лицо в кадре не должно быть переэкспонировано (light), недоэкспонировано (dark) или размыто (blur).

См. раздел «Качество изображения» для подробной информации по пороговым значениям качества изображения.

7.1.3 Порог Liveness

Для корректировки оценки Liveness используется **порог Liveness**.

Порог Liveness — порог, ниже которого система будет считать результат атакой на биометрическое предъявление («spoof»). Данный порог задается в настройке «real_threshold» из секции «LUNA_REMOTE_SDK_LIVENESS_ESTIMATOR_SETTINGS» сервиса Configurator. Значение порога по умолчанию — 0.5 (50%).

Для изображений, полученных с мобильных устройств, рекомендуется задавать порог **0.5**. Для изображений, полученных с веб-камеры, рекомендуется задавать порог **0.364**. При этом значении порога получаются приблизительно равные результаты по точности работы алгоритма.

7.1.3.1 Изменение порога на ресурсах «/handlers» и «/verifiers»

В ресурсах «/handlers» и «/verifiers» есть один дополнительный параметр — «liveness_threshold», который задаёт порог Liveness threshold

Задание данного параметра переопределяет значение настройки «real_threshold» из секции «LUNA_REMOTE_SDK_LIVENESS_ESTIMATOR_SETTINGS» сервиса Configurator.

7.1.3.2 Изменение порога на ресурсах «/liveness» и «/sdk»

Для ресурсов «/liveness» и «/sdk» нет дополнительных параметров для переопределения порога. Порог задается в настройке «real_threshold» из секции «LUNA_REMOTE_SDK_LIVENESS_ESTIMATOR_SETTINGS» сервиса Configurator.

7.2 Видеоаналитика

В LUNA PLATFORM доступна возможность выполнения видеоаналитики с помощью двух способов:

- ресурса «videosdk»
- сервисов видеоаналитики

Основным отличием двух способов является формат сохранения данных. Ресурс «videosdk» позволяет обработать видеофайл и получить результат обработки в теле ответа. Полученная информация доступна только в теле ответа и никуда не сохраняется. Сервис Video Agent же позволяет обработать видеопоток или видеофайл, и отправить результат обработки в стороннюю систему. Более того, поскольку оба способа преследуют разные цели, то для них может быть свой набор определенных параметров и значений. Например, для сервисов видеоаналитики доступна возможность задать момент генерации события (в начале трека, в конце трека, периодически), тогда как в ресурсе «videosdk» такой возможности нет. Особенности наличия тех или иных параметров приведены в соответствующих разделах ниже.

Важно! На данный момент функционал видеоаналитики находится в статусе бета-тестирования. Входные и выходные схемы могут быть изменены в будущих релизах без поддержки обратной совместимости.

Видеоаналитика — это набор функций, которые обрабатывают кадр за кадром и оценивают полезные данные.

Важным понятием видеоаналитики является понятие **трека**. Трек представляет собой длительное непрерывное наблюдение, в котором могут генерироваться несколько событий через определенные промежутки времени. Трек нужен для организации и агрегации данных о группах людей, которые могут появляться и перемещаться на видео.

При необходимости можно повернуть видео или видеопоток перед его обработкой на угол 90, 180 или 270 градусов.

В процессе выполнения видеоаналитики генерируется определенное количество событий по некоторым правилам, где каждое событие имеет начало и конец. События содержат специфическую информацию конкретной видеоаналитики. В ответе также содержится базовая метаинформация о видео (количество кадров, частота кадров, длительность видео).

Важно! Событие в видеоаналитике никак не связано с событиями, генерируемыми по обработчикам.

Настройки обработки видео, такие как количество рабочих процессов декодера, тип процессора для декодирования видео, временная директория для хранения видео пр., можно задать в группе настроек «LUNA_REMOTE_SDK_VIDEO_SETTINGS» сервиса Remote SDK в случае использования запроса «videosdk» и «LUNA_VIDEO_AGENT_VIDEO_SETTINGS» в случае использования сервисов видеоаналитики.

7.2.1 Видеоаналитика подсчета количества людей

Трек начинается тогда, когда на последнем кадре из указанного количества подряд идущих кадров (параметр «probe_count») появляется указанное число людей (параметр «people_count_threshold»). Например, если минимальное количество людей равно 10, а количество подряд идущих кадров равно 3, то в случае наличия на 3 кадрах 10 человек начнется трек с 3го кадра. Если на 2 кадрах 10 человек, а на 3 кадре 9 человек, то трек не начнется.

По умолчанию обрабатываются каждые 10 кадров. При необходимости можно настроить частоту обработки кадров с помощью параметра «rate» (например, обрабатывать каждые 3 секунды или каждые 3 кадра).

Логика генерации событий видеоаналитики отличается для запросов «videosdk» и сервисов видеоаналитики.

При выполнении аналитики с помощью ресурса «videosdk», количество событий будет равно количеству треков.

При выполнении видеоаналитики с помощью сервиса Video Agent, события видеоаналитики можно генерировать в следующих случаях (политика «event_policy»):

- когда трек начинается («trigger» > «start»)
- когда трек заканчивается («trigger» > «end»)
- периодически пока трек существует («trigger» > «period» и «interval»)

Например, если в кадре постоянно находятся 100 человек в течение часа, сервис может генерировать события каждые 5 минут, чтобы отслеживать изменения или подтверждать постоянное количество людей.

В теле запросов можно задать параметр «targets», принимающий следующие значения:

- «coordinates» — вычисление координат людей
- «overview» — получение изображения с максимальным количеством людей и включение сохранения изображений. При необходимости можно детально настроить качество, формат и максимальный размер сохраняемого изображения с помощью группы параметров «image_retain_policy».

Можно задать как оба значения, так и ни одного. Если значения не заданы, то будет выполнен базовый анализ, содержащий только информацию о видеозаписи, количестве людей и кадрах, связанных с ними.

Для такой видеоаналитики также доступна возможность настроить область интереса ROI или область интереса DROI на кадре (координаты «x», «y», ширина, высота, единицы измерения — проценты или координаты). Принцип работы области интереса ROI аналогичен FaceStream.

В каждом событии видеоаналитики подсчета количества людей содержится следующая информация:

- «event_id» — идентификатор события
- «track_id» — идентификатор трека, связывающий события, генерируемые в рамках одного непрерывного наблюдения
- «people_count» — максимальное количество людей, обнаруженное на обработанном сегменте видео
- «video_segment» — информация о сегменте видео (время начала события, время конца события, номер первого кадра, номер последнего кадра)
- «frames_estimations» (только ресурс «videosdk») — массив с информацией об оценках на каждом кадре, содержащий следующую информацию:
  - координаты людей
  - время, соответствующее кадру
  - количество людей
- «overview» — информация о кадре на видео, где было найдено больше всего людей:
  - изображение
  - время, соответствующее кадру
  - координаты людей

7.2.2 Видеоаналитика отслеживания людей

В случае видеоаналитики отслеживания людей, трек — объект, содержащий информацию о положении одного человека на последовательности кадров. Трек создается тогда, когда на последнем кадре из указанного количества подряд идущих кадров (параметр «probe_count») появляется детекция лица, тела или тела с лицом (зависит от типа детектора в параметре «detector_type»). Треков может быть столько, сколько людей было обнаружено на кадре.

При выполнении видеоаналитики с помощью запроса «videosdk», то для одного трека может быть только одно событие.

Например, если «probe_count» = «3» и на двух подряд идущих кадрах есть детекции, а на третьем нет, то трек и событие не начнутся. Если в какой-то момент отслеживания человека на двух подряд идущих кадрах были детекции, а на третьем кадре детекция отсутствует, то трек и событие закончатся.

При выполнении видеоаналитики с помощью сервиса Video Agent, события видеоаналитики можно генерировать в следующих случаях (политика «event_policy»):

- когда трек начинается («trigger» > «start»)
- когда трек заканчивается («trigger» > «end»)
- периодически пока трек существует («trigger» > «period» и «interval»)

Во время отслеживания человека доступна возможность выполнить оценку атрибутов лиц, тел, изображения, а также получить изображение, в котором была лучшая детекция. Перечень оценок задается в параметре «targets» и отличается в зависимости от способа выполнения аналитики

(«videosdk» или сервиса Video Agent). Оценка может быть выполнена по одному лучшему снимку из трека или по нескольким (параметр «face_samples»/«body_samples» > «count»). При необходимости можно фильтровать лучшие снимки по параметрам «score», «head_pitch», «head_roll», «head_yaw».

При необходимости можно включить повторную идентификацию тела (ReID). Функция ReID используется для повышения точности отслеживания путем объединения различных треков одного человека.

Для такой видеоаналитики также доступна возможность настроить область интереса ROI или область интереса DROI на кадре (координаты «x», «y», ширина, высота, единицы измерения — проценты или координаты). Принцип работы области интереса ROI аналогичен FaceStream.

В каждом событии видеоаналитики отслеживания людей содержится следующая информация:

- «event_id» — идентификатор события видеоаналитики
- «track_id» — идентификатор трека, связывающий события, генерируемые в рамках одного непрерывного наблюдения
- «people_count» — максимальное количество людей, обнаруженное на обработанном сегменте видео
- «video_segment» — информация о сегменте видео (время начала события, время конца события, номер первого кадра, номер последнего кадра)
- «frames_estimations» — массив с информацией об оценках на каждом кадре, содержащий следующую информацию:
  - номер кадра
  - время, соответствующее кадру
  - количество людей
- «overview» — информация о кадре на видео, где было найдено больше всего людей:
  - изображение
  - время, соответствующее кадру
  - координаты людей

При использовании сервиса Video Agent, в каждом событии также отображается идентификатор потока и местоположение, задаваемое в запросе на создание потока.

7.2.3 roi

ROI задаёт область интереса в которой происходит оценка.

Указанная прямоугольная область вырезается из кадра и именно это изображение в дальнейшем обрабатывает LUNA PLATFORM.

> Правильное использование параметра «roi» существенно повышает производительность видеоанализа.

Область интереса ROI на исходном кадре задается параметрами «x», «y», «width», «height» и «mode», где:

- «x» и «y» – координаты верхней левой точки области интереса ROI;
- «width» и «height» — ширина и высота обрабатываемой области кадра;
- «mode» – режим указания «x», «y», «width» и «height». Доступно два режима:
  - «abs» — параметры «x», «y», «width» и «height» задаются в пикселях;
  - «percent» — параметры «x», «y», «width» и «height» задаются в процентах от текущего размера кадра.

  Если поле «mode» не указывается в теле запроса, то будет использовано значение «abs».

При значениях ширины и высоты, равных «0», областью интереса считается весь кадр.

Система координат на изображении задается аналогично рисунку ниже.

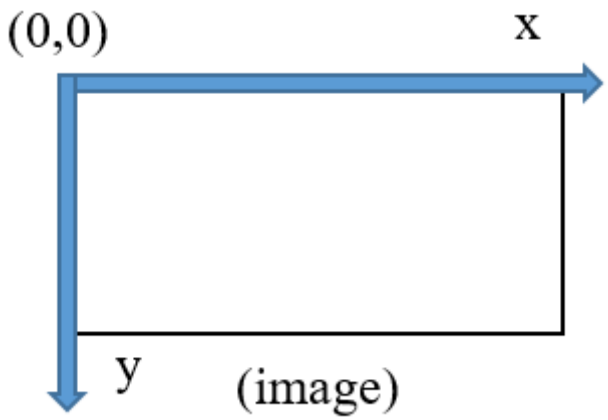


Рис. 56: Система координат ROI

Ниже приведен пример расчета ROI в процентах:

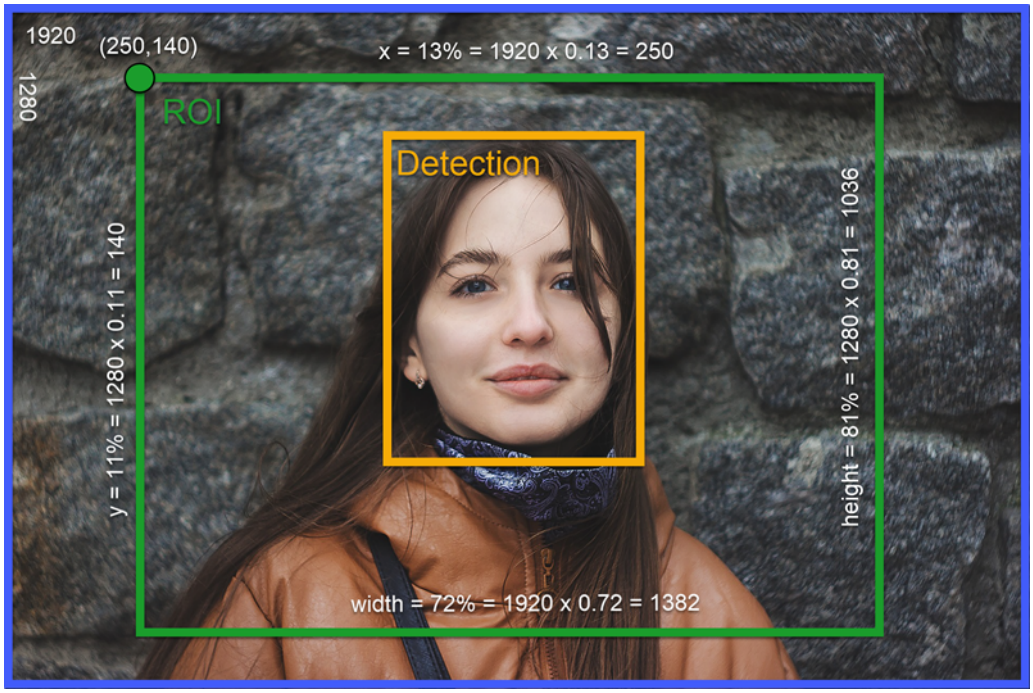


Рис. 57: Пример расчета ROI в процентах

7.2.4 droi

DROI задаёт область интереса относительно исходного кадра. Если ROI предназначен для оптимизации эстимации соответствующих свойств, то DROI работает как фильтр после выполненной эстимации и предназначен для реализации бизнес-логики. Можно использовать DROI как вместе с ROI, так и отдельно. Например, если после обработки по области ROI количество людей на кадре было равно N, то после выполнения дополнительной фильтрации по области DROI количество людей на кадре может быть уменьшено до M.

Область интереса DROI на исходном кадре задается следующими параметрами:

- «area» — геометрия области интереса. Параметр представлен в виде массива полигонов. Каждый полигон представлен массивом объектов, где каждый объект содержит координаты вершины полигона (x и y). Так, например, можно задать треугольную область интереса.
- «mode» – режим указания «x», «y». Доступно два режима:
  - «abs» — параметры «x», «y» задаются в пикселях;
  - «percent» — параметры «x», «y» задаются в процентах от текущего размера
- «form» — формат области интереса. Может принимать значения: «common» - общий формат и «wkt» (Well-known text) - текстовый формат представления геометрических объектов.

7.3 Фильтры

В LUNA PLATFORM доступна возможность фильтровать объекты по определенным правилам. Например, можно сравнивать лица только из определенного списка или получать события за какой-либо промежуток времени.

Как правило фильтры расположены в поле «filters» различных запросов.

7.3.1 Фильтры по операторам сравнения

Для некоторых фильтров доступно указание операторов сравнения с помощью суффикса ”__operator”. Оператор используется как для временных фильтров (с какой-то даты, за какое-то время), так и для обычных фильтров (за какой-то возраст человека, за перечень идентификаторов в лексикографическом порядке).

__lt

Фильтр «less than» используется для поиска значений меньше заданного. Например, при использовании фильтра ”create_time__lt«:»2022-08-11T09:11:41.674Z”в запросе «get events» будут возвращены события с временем создания, предшествующим указанной дате и времени, т.е. события созданные до 11 августа 2022 года, 09:11:41.674 по времени UTC.

__lte

Фильтр «less than or equal to» используется для поиска значений меньше или равных заданному. Например, при использовании фильтра ”face_id__lte«:»2046d39b-410d-481e-b9de-ede6a0c7367f”в фильтрах запроса «matching faces» для кандидатов, будут возвращены только те кандидаты, чьи идентификаторы начинаются с «2046d39b-410d-481e-b9de-ede6a0c7367f» или меньше этого значения в лексикографическом порядке.

__gte

Фильтр «greater than or equal to» используется для поиска значений больше или равных заданному. Например, при использовании фильтра ”similarity__gte«:»0.9”в политике «storage_policy» > «face_sample_policy» > «filters» > «match» запроса «create handler» можно настроить правило сохранения биометрического образца лица в БД Faces только в том случае, если политика «matching_policy» определила, что схожесть кандидата с эталоном больше или равна «0.9».

7.3.2 Фильтры now-time

Для временных фильтров «create_time», «insert_time» и «end_time» доступен формат задания времени относительно текущего времени (формат «now-time»). Например, при создании расписания для задачи Garbage collection можно указать фильтр ”create_time__lt«:»now-30d что позволит удалить все объекты кроме тех, что создавались за последние 30 дней.

Необходимо помнить, что:

- при создании расписания, текущее время будет отсчитываться не от создания расписания, а от создания задачи расписанием в соответствии с cron-выражением;
- при генерации события по обработчику в котором указан фильтр «now-time» (например в политике «matching_policy» > «candidates»), текущее время будет отсчитываться от момента генерации события.

> Для использования времени относительно генерации события можно задавать политики непосредственно в запросе на генерацию события с помощью схемы «multipart/form-data» или с помощью задачи Estimator с созданием нового обработчика.

В этом формате время задаётся по следующему шаблону `'now-(\d+)[smhdwMy]'`, где «[d+]» — число, «[smhdwMy]» — необходимый период: m (минуты), h (часы), d (дни), w (недели), M (месяцы), y (годы).

7.4 Пользовательские интерфейсы

7.4.1 Веб-сервис LUNA CLEMENTINE 2.0

Для выполнения основных запросов LP для создания объектов и выполнения операций сравнения может использоваться веб-сервис LUNA CLEMENTINE 2.0.

Данный сервис не поставляется вместе с LUNA PLATFORM 5, его необходимо скачивать и настраивать отдельно.

7.4.2 Интерфейсы сервисов Backport 3 и Backport 4

В LUNA PLATFORM 5 доступны интерфейсы для взаимодействия с API LP 4 и LP 3. Интерфейсы предназначены только для тех пользователей, которые предпочли не переходить на новую версию API LUNA PLATFORM 5. Они не включают в себя большинство функций, доступных в LP 5, и никогда не будут включать.

Сервис	Описание	Данные для входа по умолчанию	Порт по умолчанию
Backport 4	Интерфейс для работы с API LP 4	-	4200
Backport 3	Интерфейс для работы с API LP 3	Для входа в интерфейс требуется создать аккаунт на вкладке Sign Up.	4100

7.4.3 Прочие интерфейсы

Данные интерфейсы нужны для упрощения работы с сервисами LUNA PLATFORM.

Сервис	Описание	Данные для входа по умолчанию	Порт по умолчанию
Configurator	Интерфейс для работы с настройками LUNA PLATFORM. Позволяет в одном месте настроить параметры всех сервисов. Особенно удобен, если включена автоперезагрузка сервисов после обновления настроек.	-	5070

Сервис	Описание	Данные для входа по умолчанию	Порт по умолчанию
Admin	Интерфейс для выполнения административных задач LUNA PLATFORM: создание аккаунтов через GUI, запуск и контроль выполнения длинных задач (задача Garbage collection task и задача Additional extraction).	root@visionlabs.ai/root	5010
InfluxDB	Интерфейс позволяет просматривать данные мониторинга сервисов LUNA PLATFORM.	luna/password	8086
Grafana (LUNA Dashboards)	Интерфейс для визуализации данных мониторинга LUNA PLATFORM, хранящихся в InfluxDB. Для него написаны дашборды для визуализации и фильтрации информации. Также можно использовать систему агрегации логов Grafana Loki.	admin/admin	3000

7.5 Отключаемые сервисы

Некоторые неиспользуемые основные сервисы можно отключить. При отключении сервиса, его основные функции не будут работать. Так, например, если отключить сервис Tasks, то при попытке выполнить задачу будет возвращена ошибка или если отключить сервис Sender и использовать обработчик с политикой «notification_policy», то это также приведет к ошибке.

В итоге все ресурсы, которые требуют наличия отключенного сервиса будут возвращать ошибки вида `<Service_name> service is disabled` с кодом ответа 403.

Если в процессе работы LUNA PLATFORM произойдет сбой в работе какого-либо сервиса или он будет вручную остановлен, то работа зависимых сервисов завершится ошибкой. Например, при остановке сервиса Licenses, в логах сервиса Faces будет отображена следующая ошибка:

```
[0000001 2023-08-11 13:00:51.042000] ERROR: luna-faces
   1690887528,00000000-0000-4000-a000-000000000001: Check connection to Luna
    Licenses : GET:http://127.0.0.1:5120/version:Cannot connect to host
   127.0.0.1:5120 ssl:default [Connection refused]
```

> Соответствующая ошибка будет возвращена при попытке сделать запрос, который зависит от проблемного сервиса.

Затем работа сервисов, зависимых от сервиса Faces, также будет завершена аналогичной ошибкой.

Проверить зависимость сервисов друг от друга можно с помощью диаграммы взаимодействия или можно найти настройку вида «service_name_ADDRESS» в настройках сервиса, для которого требуется определить зависимые сервисы. Например, в настройках сервиса Faces есть настройка «LUNA_LICENSES_ADDRESS», отвечающая за соединение с сервисом Licenses, а в настройках сервиса Admin есть настройка «LUNA_FACES_ADDRESS», отвечающая за соединение с сервисом Faces.

7.5.1 Отключение сервиса Configurator

Сервис Configurator отключается посредством остановки контейнера.

При отключении сервиса Configurator, остальные сервисы будут использовать конфигурационные файлы `config.conf`, расположенные в директории `srv/<service_name>/configs` соответствующего контейнера.

7.5.2 Процесс отключения сервисов

- откройте пользовательский интерфейс Configurator `http://<configurator_server_ip>:5070`;

- введите название настройки «ADDITIONAL_SERVICE_USAGE» в поле «Setting name» и нажмите «Apply Filters»;
- установите значение для необходимого сервиса «false»;
- сохраните изменения, нажав кнопку «Save».

7.5.3 Последствия отключения сервиса Image Store

При отключении сервиса Image Store, существуют некоторые особенности, перечисленные ниже:

- объекты типа `images`, `objects`, `samples` и политики сохранения биометрических образцов в обработчиках/верификаторах будут недоступны.
- пропадет возможность использования всех задач, кроме задач Garbage Collection, Linker и Estimator, однако для этих задач есть ряд ограничений:
  - Garbage Collection, Estimator, Linker: результаты задач/подзадач не будут сохраняться
  - Garbage Collection, Estimator, Linker: после завершения подзадачи, статус задачи будет обновлен на `Done` и идентификатор результата задачи будет равен `None`
  - Garbage Collection: удаление биометрических образцов станет недоступным

  Если сервис Image Store отключается после того, как были сгенерированы события с включенной политикой «image_origin_policy», то при использовании задачи Garbage Collection и параметра `remove_image_origins`, сервис Tasks будет по-прежнему пытаться удалять исходные изображения, имеющие внешний URL-адрес.

7.5.3.1 Последствия отключения сервиса Image Store для Backport 3

В сервисе Backport 3 доступна настройка «BACKPORT3_ENABLE_PORTRAITS», позволяющая отключить возможность использования портретов, но оставить возможность использования остального функционала сервиса Image Store. Если использование сервиса Image Store отключено в настройке «ADDITIONAL_SERVICES_USAGE», то вышеописанная настройка также должна быть отключена.

При отключении сервиса Image Store, БО и портреты станут недоступны, как и ресурсы «get portrait» и «get portrait thumbnail».

7.5.4 Последствия отключения сервиса Handlers

При отключении сервиса Handlers:

- запуск сервиса API приведет к отсутствию возможности использования следующих запросов:
  - «detect faces»;
  - «extract attributes»;

  - «estimator task»;
  - все запросы на ресурс «/handlers»;
  - все запросы на ресурс «/verifiers».
- запуск сервиса Tasks приведет к отсутствию возможности выполнения задач «Additional extraction» и «Estimator»;
- запуск сервиса Admin приведет к отсутствию возможности выполнения задачи «Additional extraction»;

7.5.5 Последствия отключения сервиса Faces

При отключении сервиса Faces:

- работа с лицами, атрибутами и списками станет недоступна;
- политика «storage_policy» > «face_policy» (и, соответственно, «link_to_lists_policy») обработчика/верификатора станет недоступна;
- станет невозможно указывать лица в качестве кандидатов для сравнения в политике «match_policy» обработчика;
- будет отсутствовать возможность работы с задачей Linker, а задачи Clustering, Reporter, Exporter, Cross-matching, Garbage Collection, Additional extraction и ROC-curve calculating во всех сервисах не смогут работать с лицами и атрибутами.

7.5.6 Последствия отключения сервиса Events

При отключении сервиса Events:

- можно будет создать обработчик с политикой сохранения события, однако само событие будет невозможно сгенерировать;
- все запросы к ресурсу «/events» не будут работать;
- запросы к ресурсам «/handlers/{handler_id}/events», «/handlers/{handler_id}/events/raw» не будут работать;
- параметр «event_id» в запросе «create face» не будет работать;
- сравнение по событиям-кандидатам будет недоступно;
- фильтры по событиям в задачах будут недоступны;
- использование запроса «ws handshake» не будет иметь смысла.

7.5.7 Последствия отключения сервиса Tasks

При отключении сервиса Tasks:

- все запросы к ресурсу «/tasks» (группы параметров «task processing», «task info», «task errors») не будут работать;
- использование задач во всех сервисах будет недоступным.

> Задачи, выполняемые в пользовательском интерфейсе сервиса Admin, также недоступны при отключении сервиса Tasks.

7.5.8 Последствия отключения сервиса Sender

При отключении сервиса Sender:

- запрос «ws handshake» не будет работать;
- политика «notification_policy» в обработчике не будет работать.

7.5.9 Отключение дополнительных сервисов

Дополнительные сервисы также могут быть отключены, если ранее были включены.

Если сервис Python Matcher Proxy ранее использовался, то его отключение приведет к невозможности использования плагинов сравнения или модуля LIM.

Если сервис Lambda ранее использовался, то его отключение приведет к невозможности отправки запросов к ресурсам «/lambdas» и работе с созданной lambda.

Если сервисы Backport 3 или Backport 4 ранее использовались, то их отключение приведет к невозможности обработки запросов LUNA PLATFORM 3/4 с помощью LUNA PLATFORM 5.

> Включение сервисов Backport 3, Backport 4, User Interface 3 и User Interface 4 регулируется только запуском соответствующих контейнеров. В настройках «ADDITIONAL_SERVICE_USAGE» нет параметров, включающих данные сервисы.

7.6 Шифрование биометрических шаблонов

Для повышения безопасности и предотвращения несанкционированного использования биометрических шаблонов, они могут быть получены в зашифрованном виде. Это позволяет защитить биометрические шаблоны от кражи и последующего использования в других системах.

Примечание. Шифрование добавляет дополнительные вычислительные затраты, что приводит к замедлению процессов, таких как сравнение по базе, сравнение с использованием LUNA Index Module и синхронизация кеша в Cached Matcher.

В результате шифрования биометрические шаблоны хранятся в зашифрованном виде в базах данных Events, Faces или Attributes и в том же виде передаются во внешние системы. Соответственно, если включено шифрование, то на запросы, которые возвращают биометрический шаблон, будет получен именно зашифрованный биометрический шаблон.

Важно! LUNA PLATFORM не предназначена для одновременного использования зашифрованных и незашифрованных биометрических шаблонов. В случае обнаружения обоих видов биометрических шаблонов, сервис Python Matcher не запустится, выдав ошибку «Check is failed, encryption hash is not unique». Поэтому, при включении шифрования, необходимо остановить сервисы, ограничив тем самым все запросы, перевести все биометрические шаблоны на новую версию шифрования и только потом заново запускать сервисы. Крайне рекомендуется выполнения бекапа БД перед манипуляциями с шифрованием.

Примечание. Плагины для сравнения также необходимо обновить в соответствии с логикой шифрования. LUNA Index Module поддерживает работу с зашифрованными биометрическими шаблонами.

7.6.1 Формат зашифрованных биометрических шаблонов

Зашифрованные биометрические шаблоны имеют следующий формат: `<encrypted_descriptor><tag><nonce><hash>`.

- `encrypted_descriptor` — Зашифрованный биометрический шаблон.
- `tag` — Данные, используемые для аутентификации сообщения.
- `nonce` — Инициализационный вектор для шифрования.
- `hash` — Хэш-сумма ключа шифрования и алгоритма.

7.6.2 Включение и настройка шифрования

Шифрование можно включить и настроить с помощью группы параметров `DESCRIPTOR_ENCRYPTION`.

Для включения шифрования необходимо установить параметр `enabled` в значение `true`. Это активирует шифрование биометрических шаблонов в системе. Параметр `algorithm` задает используемый алгоритм шифрования (по умолчанию `aes256-gcm`).

Параметры шифрования указываются в разделе `params`. Здесь необходимо указать источник ключа шифрования с помощью параметра `source`. Поддерживаются два типа источников: `raw` и `vaultKV`.

- При использовании `raw`, ключ шифрования указывается напрямую в параметре `key`.
- При использовании `vaultKV`, ключ шифрования необходимо получать из хранилища Hashicorp Vault. В этом случае параметр `key` должен содержать URL для получения ключа и токен аутентификации. Пример конфигурации для `vaultKV`:

```
{
    "enabled": true,
    "algorithm": "aes256-gcm",
    "params": {
        "source": "vaultKV",
        "key": {
            "url": "https://vault.example.com/v1/secret/data/encryption_key"
                ,
            "token": "s.XYZ12345"
        }
    }
}
```

> Hashicorp Vault — это инструмент для управления секретами и защиты конфиденциальных данных, таких как ключи шифрования.

Содержимое хранилища ключей/значений `Vault` должно быть в следующем формате:

```
{
    "key": "...",
    "algorithm": "..."
}
```

7.6.3 Управление шифрованием биометрических шаблонов

При необходимости можно добавить шифрование для уже существующих биометрических шаблонов, заменить существующее шифрование или выполнить дешифрование биометрических шаблонов в БД Faces, БД Attributes или БД Events.

Для этого необходимо выполнить миграцию биометрических шаблонов с помощью скрипта `descriptors_encryption.py`, расположенного в контейнерах сервисов Faces и Events, передав в качестве аргументов скрипта данные о БД Faces/Attributes/Events, и передав в качестве переменных окружения соответствующий ключ и алгоритм следующим образом:

```
docker run \
--env=OLD_ENCRYPTION_KEY=<your_old_encryption_key> \
--env=NEW_ENCRYPTION_KEY=<your_new_encryption_key> \
--env=ENCRYPTION_ALGORITHM=aes256-gcm \
--rm \
...
dockerhub.visionlabs.ru/luna/luna-faces:v.4.13.6 \
python3 ./base_scripts/descriptors_encryption.py --luna-config=http
    ://127.0.0.1:5070/1 --LUNA_FACES_DB=<LUNA_FACES_DB_TAG> --DATABASE_NUMBER
    =<DATABASE_NUMBER_TAG> --LUNA_ATTRIBUTES_DB=<ATTRIBUTES_DB_TAG>
```

Важно! Скрипт необходимо выполнять когда сервис остановлен.

В зависимости от сценария (добавление/обновления/дешифрования), определенные переменные окружения могут не использоваться. Например, при добавлении шифрования к биометрическим шаблонам без шифрования, переменная `OLD_ENCRYPTION_KEY` не будет использоваться.

См. подробную информацию о скрипте миграции биометрических шаблонов в разделе «Управление шифрованием биометрических шаблонов» в руководствах по обновлению в разделе «Дополнительная информация».

7.7 Особенности работы с сервисами

При работе с разными сервисами необходимо учитывать некоторые нюансы, описанные в этом разделе.

7.7.1 Автоориентация повернутого изображения

Не рекомендуется отправлять повернутые изображения на LP, поскольку они не обрабатываются должным образом и их необходимо поворачивать. Существует два метода автоматической ориентации повернутого изображения – на основе данных EXIF изображения (параметр запроса) и с использованием алгоритмов LP (настройка Configurator). Оба метода автоматической ориентации изображения можно использовать вместе.

> Если автоматическая ориентация не используется, изображение будет поворачиваться с помощью механизма создания биометрического образца и будет создаваться изображение с произвольным углом поворота.

7.7.1.1 Автоориентация на основе данных EXIF

Этот способ ориентации изображения выполняется в параметрах запроса с помощью параметра «use_exif_info». С его помощью можно включать или отключать автоматическую ориентацию изображения на основе EXIF данных.

Данный параметр доступен и включен по умолчанию в следующих ресурсах:

- «/detector»
- «/verifiers/{verifier_id}/verifications»
- «/sdk»
- «/handlers/{handler_id}/events»

Параметр «use_exif_info» нельзя использовать с биометрическими образцами. Если задан параметр «warped_image» или задано соответствующее значение параметра «image_type», то параметр «use_exif_info» будет игнорироваться.

7.7.1.2 Автоориентация на основе настроек сервиса Configurator

Этот метод ориентации изображения используется в сервисе Configurator с помощью настройки «LUNA_REMOTE_SDK_USE_AUTO_ROTATION». Если эта настройка включена и входное изображение повернуто на 90, 180 или 270 градусов, то LP поворачивает изображение под соответствующим углом. Если эта настройка включена, но входное изображение не повернуто, LP не будет поворачивать изображение.

> Выполнение автоматической ориентации требует значительного количества ресурсов сервера.

Настройку «LUNA_REMOTE_SDK_USE_AUTO_ROTATION» нельзя использовать с биометрическими образцами. Если задан параметр «warped_image» или задано соответствующее значение параметра «image_type», то настройка «LUNA_REMOTE_SDK_USE_AUTO_ROTATION» будет игнорироваться.

7.7.2 Особенности сохранения исходных изображений

URL-адрес исходного изображения можно сохранить в поле «image_origin» созданных событий при обработке запроса «/handlers/{handler_id}/events».

Для этого при создании обработчика необходимо указать параметр «store_image» в «image_origin_policy».

Затем необходимо задать изображение для обработки в запросе «generate events».

Если «use_external_references»=0 и URL-адрес внешнего изображения был передан в запросе «generate events», то это изображение будет сохранено в хранилище Image Store, а идентификатор сохраненного изображения будет добавлен в поле «image_origin» сгенерированного события.

С помощью параметра «use_external_references» можно сохранить внешнюю ссылку вместо сохранения самого изображения:

- Если «use_external_references»=1 и URL-адрес внешнего изображения был передан в запросе «generate events», то этот URL-адрес будет добавлен в поле «image_origin». Само изображение не будет сохранено в Image Store.
- Если «use_external_references»=1 и биометрический образец был передан в запросе «generate events» и включен параметр «face_sample_policy > store_sample», URL-адрес биометрического образца в Image Store будет сохранен в поле «image_origin». Таким образом, можно избежать дублирования изображения. Если внешний URL-адрес слишком длинный (более 256 символов), сервис сохранит изображение в Image Store.

Также можно непосредственно указать URL-адрес исходного изображения, используя ресурс «/handlers/{handler_id}/events». Для этого необходимо использовать в запросе схему тела «application/json» или «multipart/form-data». URL-адрес необходимо указывать в поле «image_origin» запроса.

Если поле «image_origin» не пустое, предоставленный URL-адрес будет использоваться в созданном событии независимо от политики «image_origin_policy».

> Изображение, указанное в поле «image_origin», не будет обрабатываться в запросе. Оно используется только как исходное изображение.

7.8 Нейросети

С помощью нейронных сетей выполняется оценивание параметров лиц или тел и извлечение их биометрических шаблонов.

Существует два типа нейронных сетей — для выполнения оценивания и для извлечения БШ.

Нейросети для выполнения обнаружения и оценивания расположены в контейнере сервиса Remote SDK в формате `<estimation_name>_<architecture>.plan`, где `<estimation_name>` — название нейронной сети, `<architecture>` используемая архитектура (`cpu-avx2` или `gpu`). Такого рода нейросети называются **детекторами** и **эстиматорами** соответственно.

Нейросети для извлечения биометрических шаблонов расположены в контейнере сервиса Remote SDK в формате `cnn<model>_<architecture>.plan`, где `<model>` — модель нейронной сети, `<architecture>` используемая архитектура (`cpu-avx2` или `gpu`).

Для работы с нейросетью используется конфигурационный файл формата `cnndescriptor_<model>.conf`, где `<model>` — модель нейронной сети. Конфигурационные файлы для всех нейросетей также расположены в контейнере сервиса Remote SDK.

> Далее будет описана информация только для нейронных сетей для извлечения биометрических шаблонов.

> Можно удалить какую-либо нейронную сеть из контейнера сервиса Remote SDK, если выключается использование некоторых эстиматоров или детекторов.

В текущей сборке LUNA PLATFORM поддержаны модели нейросетей для извлечения биометрических шаблонов:

Объект, из которого извлекается БШ	Модели нейронных сетей	Модель по умолчанию
Лицо	59, 60, 62	62
Тело	116	116

Размеры всех моделей нейронных сетей для извлечения биометрических шаблонов лиц приведены ниже:

Модель нейронной сети	Размер данных в формате Raw (байты)	Размер метаданных (байты)	Размер данных в формате SDK (Raw + метаданные)
54	512	8	520
56	512	8	520

Модель нейронной сети	Размер данных в формате Raw (байты)	Размер метаданных (байты)	Размер данных в формате SDK (Raw + метаданные)
57	512	8	520
58	512	8	520
59	512	8	520
60	512	8	520
62	512	8	520

Размеры всех моделей нейронных сетей для извлечения биометрических шаблонов тел приведены ниже:

Модель нейронной сети	Размер данных в формате Raw	Размер метаданных (байты)	Размер данных в формате SDK (Raw + метаданные)
102	2048	8	2056
103	2048	8	2056
104	2048	8	2056
105	512	8	520
106	512	8	520
107	512	8	520

> См. подробную информацию о форматах БШ в разделе «Форматы биометрических шаблонов».

> Нейросеть 105 является более новой версией нейросети 102.

Биометрические шаблоны, полученные с использованием разных моделей нейронной сети, не сопоставимы друг с другом. Поэтому необходимо повторно извлекать все БШ из существующих БО при использовании новой модели нейронной сети (см. раздел «Изменение используемой модели нейросети»).

Можно хранить для одного лица или тела несколько БШ полученных для одного изображения с помощью разных нейронных сетей.

> См. параметры «DEFAULT_FACE_DESCRIPTOR_VERSION» и «DEFAULT_HUMAN_DESCRIPTOR_-VERSION» в сервисе Configurator для проверки текущих нейронных сетей для извлечения.

> Перед любыми действиями с нейронными сетями необходимо проконсультироваться со специалистами VisionLabs.

7.8.1 Изменение используемой модели нейросети

Изменение используемой модели нейросети требуется когда необходимо повысить качество распознавания лиц/тел или когда старые модели объявляются устаревшими и удаляются из контейнера сервиса Remote SDK.

При изменении используемой модели нейросети необходимо:

- **для сохранения возможности использования старых БШ:** повторно выполнить извлечение существующих биометрических шаблонов, чтобы БШ были извлечены в помощью новой модели нейросети. Повторное извлечение БШ реализовано в виде длительной задачи Additional extraction (см. «Запуск задачи Additional extraction»);
- задать новую модель нейросети в настройках LP (см. «Изменение модели нейросети в настройках»).

> Не следует менять модель нейросети по умолчанию в настройках LP до того, как операция повторного извлечения БШ будет завершена.

> Можно как повышать модель нейросети, так и понижать. Для понижения модели нейросети необходимо выполнить аналогичные повышению действия.

7.8.1.1 Запуск задачи Additional extraction

Задачу Additional extraction можно запустить одним из следующих способов:

- с помощью запроса «additional extract» к сервису Admin;
- с помощью пользовательского интерфейса сервиса Admin, по умолчанию расположенного по адресу `http://<admin_server_ip>:5010`.

В зависимости от способа нужно указать следующую информацию:

- тип объекта: лица или события
- цель извлечения: БШ лица, БШ тела или базовые атрибуты
- новая версия нейронной сети (не применимо для базовых атрибутов)

См. подробную информацию в разделе «Задача Additional extraction».

7.8.1.2 Изменение модели нейросети в настройках

Новая модель нейросети задается в настройках сервиса Remote SDK в сервисе Configurator:

- Открыть пользовательский интерфейс Configurator `http://<configurator_server_ip>:5070`.
- Установить требуемую нейросеть в параметре «DEFAULT_FACE_DESCRIPTOR_VERSION» (для лиц) или «DEFAULT_HUMAN_DESCRIPTOR_VERSION» (для тел).
- Сохранить изменения, нажав кнопку «Save».
- Подождать, пока настройка не будет применена во всех сервисах LP.

7.8.2 Использование модели нейросети не из поставки

В данном разделе описывается процесс переноса нейросети не из поставки в контейнер Remote SDK. Это необходимо, если пользователь использует старую нейросеть из предыдущей версии LP и не хочет её менять при обновлении на новую версию LP.

Необходимо запросить архив с файлами нейросети от представителя VisionLabs. В архиве содержатся следующие файлы:

- файл(ы) нейронной сети `cnn<model>_<architecture>.plan`, где `<model>` — модель нейронной сети, `<architecture>` используемая архитектура (`cpu-avx2` и/или `gpu`).
- конфигурационный файл `cnndescriptor_<model>.conf`, где `<model>` — модель нейронной сети.

После скачивания архива с нейронной сетью и архива с её конфигурацией необходимо выполнить следующие действия:

- распаковать архив
- присвоить права нейросетям
- скопировать нейросети и их конфигурацию в запущенный контейнер Remote SDK
- убедиться, что в конфигурациях сервисов используется необходимая модель нейросети (см. раздел «Изменение модели нейросети в настройках»)

Ниже приведен пример команд для переноса нейронных сетей в контейнер Remote SDK.

7.8.2.1 Распаковка нейросетей

Откройте директорию с архивами и распакуйте их.

```
unzip <archive_name>.zip
```

7.8.2.2 Присвойте права нейросетям

```
chown -R 1001:0 <archive_name>/cnn<model>_<architecture>.plan
```

7.8.2.3 Копирование нейросети и конфигурационного файла в контейнер Remote SDK

Скопируйте требуемую нейросеть и ее конфигурационный файл в контейнер Remote SDK с помощью следующих команд.

```
docker cp <archive_name>/cnn<model>_<architecture>.plan luna-remote-sdk:/srv
    /fsdk/data/
```

```
docker cp <archive_name>/cnndescriptor_<model>.conf luna-remote-sdk:/srv/
    fsdk/data/
```

`luna-remote-sdk` — имя запущенного контейнера Remote SDK. Это имя может отличаться в разных инсталляциях.

Проверьте, что требуемая модель требуемого устройства (CPU и/или GPU) была успешно загружена:

```
docker exec -t luna-remote-sdk ls /srv/fsdk/data/
```

7.9 Логирование информации

В LUNA PLATFORM существует два способа вывода логов:

- стандартный вывод логов (stdout);
- вывод логов в файл.

Настройки вывода логов задаются в настройках каждого сервиса в секции `<SERVICE_NAME>_LOGGER`.

По умолчанию логи выводятся только в стандартный вывод.

> Посмотреть логи сервиса через стандартный вывод можно с помощью команды `docker logs <service_name>`.

> Рекомендуется настроить внешнюю систему для сбора и хранения логов. В данном руководстве не приводится пример настройки внешней системы для настройки логов.

При необходимости можно использовать оба способа вывода логов.

При включении сохранения логов в файл в контейнере, в качестве пути по умолчанию используется путь `srv/logs` для каждого контейнера. Логи сохраняются в формате `<service_name>_<type_of_logs>.txt`, где:

- `<service_name>` — имя сервиса, например, `luna-faces`
- `<types_of_logs>` — тип выводимых логов — «ERROR», «INFO», «WARNING», «DEBUG»

Возможно создать до 6 файлов каждого типа. Для каждого типа выводимых логов в настройках сервиса Configurator можно назначить максимальный размер (параметр «max_log_file_size» в секции `<SERVICE_NAME>_LOGGER`). Так, например, для типа «INFO» может быть создано 6 файлов по 1024 Мб. Для остальных типов принцип работы аналогичен. Таким образом, если максимальное значение «max_log_file_size» равно 1024 Мб, то общее количество памяти, занимаемой логами в контейнере не может превышать 6*4*1024=24576 Мб. После окончания оставшегося места в последнем файле, будет осуществлена перезапись первого файла. Если необходимо снизить количество занимаемой логами памяти, то необходимо уменьшить значение параметра «max_log_file_size».

Можно синхронизировать папку с логами сервиса с локальной папкой на сервере с помощью команды `-v /tmp/logs/<service_name>:/srv/logs \` при запуске контейнера. Для этого предварительно нужно создать соответствующую директорию на сервере и присвоить ей необходимые права.

> Без присвоения прав монтируемой папке сервис будет выдавать соответствующую ошибку.

> При включении сохранения логов в файле необходимо помнить о том, что логи занимают определенное место в хранилище (см. выше), а процесс логирования в файл негативно вли-

> яет на производительность системы.

7.10 Проверка изображений

LUNA PLATFORM позволяет произвести различные проверки изображений фронтального типа. Проверка может быть выполнена как с помощью порогов, соответствующих стандарту ISO/IEC 19794-5:2011, так и с помощью ручного ввода порогов и выбора необходимых проверок.

> Результаты проверок не сохраняются в БД, они возвращаются только в ответе.

Проверки на соответствие ISO/IEC 19794-5:2011 выполняются с помощью ресурса «/iso» (см. подробное описание в разделе «Проверка изображений на соответствие стандарту ISO/IEC 19794-5:2011» ниже).

Проверки с ручным указанием условий выполняются с помощью группы проверок «face_quality», политики «detect_policy», ресурсов «/handlers» и «/verifiers» (см. подробное описание в разделе «Проверка изображений по заданным условиям» ниже).

> Возможность выполнения таких проверок регулируется специальным параметром в файле лицензии.

Результат выполнения всех проверок определяется параметром «status», где:

- «0» — какая-либо из проверок не была пройдена
- «1» — все проверки были пройдены

Напротив каждой проверки также отображается её результат (параметр «result»).

Можно включить проверку для нескольких лиц на фотоизображении с помощью параметра «multiface_policy». Проверки выполняются для каждой детекции лица, найденного на фото. Результаты проверок не агрегируются.

Для некоторых проверок должны быть выполнены определенные требования. Например, чтобы получить корректные результаты статуса бровей, необходимо чтобы углы наклона головы находились в определенном диапазоне и размер лица по ширине был не менее 80 пикс. Требования для проверок перечислены в разделе «Оцениваемые данные». В случае невыполнения каких-либо требований при проверке определенного параметра, результаты проверки этого параметра могут быть некорректными.

Для следующих проверок **недоступно использование биометрического образца** в качестве исходного изображения:

- Проверка положения головы по вертикали/горизонтали (параметры `head_horizontal_center`, `head_vertical_center`)
- Проверка вертикального/горизонтального размеров головы (параметры `head_width`, `head_height`)
- Проверка расстояния между центрами глаз (параметр `eye_distance`)
- Проверки ширины/высоты лица (параметры `indent_upper`, `intent_lower`, `intent_right`, `intent_left`)

- Проверки отступов от краёв фото (параметры `indent_upper`, `intent_lower`, `intent_right`, `intent_left`)
- Проверка равномерности освещения по стандарту ICAO (параметр `illumination_uniformity`)
- Проверка динамического диапазона (параметр `dynamic_range`)
- Проверка фона (параметры `background_lightness`, `background_uniformity`)

Набор проверок для «face_quality» и ресурса «/iso» отличается (см. различие проверок в разделе «Таблица сравнения доступных проверок»).

7.10.1 Проверка изображений на соответствие стандарту ISO/IEC 19794-5:2011

Такая проверка выполняется с помощью ресурса «/iso».

По умолчанию проверяются фотоизображения, на которых присутствует одно лицо. Для каждого из найденных лиц вернутся оценки и координаты найденного лица. Следует учитывать, что многие проверки по стандарту ISO предполагают наличие одного лица в кадре, поэтому не все проверки для нескольких лиц будут выполнены успешно.

Порядок возвращаемых ответов после обработки соответствует порядку передаваемых фотоизображений.

В запросе можно дополнительно включить извлечение EXIF данных фотоизображения.

Для каждой проверки заданы пороги, соответствующие требованиям ISO. Значение порогов для каждой проверки приведено в примере ответа на запрос «/iso» в документации OpenAPI.

Некоторые проверки объединяются под одно требование ISO. Например, для успешного прохождения проверки статуса глаз необходимо, чтобы статусы левого и правого глаза принимали значение «open».

В ответе на запрос ресурс возвращает:

- Вердикт о прохождении проверок, который равен 1, если все проверки успешны.
- Результаты выполнения каждой из проверок. Это позволяет определить, какая именно проверка не была пройдена. Возвращаются следующие значения:
  - Название проверки.
  - Полученное после выполнения проверки значение.
  - Заданный по умолчанию порог. Пороги заданы в системе в соответствием с требованиями стандарта ISO/IEC 19794-5:2011 и не изменяются пользователем.
  - Результат этой проверки. При прохождении порогов возвращается 1.
- Координаты найденного лица.

При возникновении ошибки для одного из обрабатываемых фотоизображений, например, если изображение повреждено, в ответе будет выведена ошибка. Обработка остальных фотоизображений будет продолжаться в обычном режиме.

> Кроме ресурса «/iso» доступна возможность выполнить проверку на соответствие стандартам ISO/IEC 19794-5:2011 и ICAO в ресурсе «/detector» (параметр «estimate_face_quality»). Принцип выполнения проверок аналогичен вышеописанному, однако в данном ресурсе доступны дополнительные проверки из группы проверок «face_quality».

7.10.2 Проверка изображений по заданным условиям

Такая проверка выполняется с помощью группы проверок «face_quality», политики «detect_policy», ресурсов «/handlers» и «/verifiers».

Принцип работы схож с проверкой на соответствие стандарту ISO, однако пользователь вправе сам решать какие проверки необходимо выполнять и какие пороги задавать.

Для включения проверок необходимо указать значение «1» в поле «estimate» для «face_quality». По умолчанию проверка изображений отключена. Для отключения определённой проверки требуется выставить «0» в поле «estimate» для этой проверки. По умолчанию все проверки включены и будут выполнены при включении «face_quality».

В зависимости от типа проверки можно указать минимальное и максимальное значения порога или допустимые значения для этой проверки. Для этого используется поле «threshold». Если минимальный или максимальный порог не задан, то в качестве незаданного порога автоматически будет выбрано минимальное или максимальное допустимое значение. Если максимальное значение неограниченно (например, «>=0»), то в теле ответа события в поле «max» вернется значение «null». Если оба порога не заданы — проверка выполнится по стандартным порогам, установленным в группе параметров «FACE_QUALITY_SETTINGS» в сервисе Configurator или в теле запроса «face_quality». Если порог указан в теле запроса «face_quality», то это переопределяет стандартные пороги, заданные в группе параметров «FACE_QUALITY_SETTINGS».

> Стандартные пороги подобраны специалистами VisionLabs для получения оптимальных результатов. Данные пороги могут отличаться в зависимости от условий съемки, оборудования и т.д.

При задании порогов для проверок «Качество изображения» и «Положение головы» рекомендуется принимать во внимание стандартные пороги, предустановленные в настройках системы. Например, для проверки изображения на размытость (параметр «blurriness_quality»), рекомендуется задавать порог в диапазоне [0.57…0.65]. При задании порога вне данного диапазона результаты могут быть непредсказуемыми. При выборе углов положения головы нужно обращать внимание на рекомендуемые максимальные пороги для проведения оценки в кооперативном и некооперативном режимах. Информация об этих порогах приведена в соответствующих главах руко-

водства администратора.

Рекомендуется рассматривать результаты проверок состояния рта («mouth_smiling», «mouth_occluded», «mouth_open», «smile_properties») совместно друг с другом. Так, например, если проверка выявила, что лицо чем-то перекрыто, то остальные результаты проверки рта не будут полезны.

Доступна возможность включения фильтрации по результатам проверок (параметр «filter»). Если одна из проверок «face_quality» для определённой детекции завершится неудачей, то будут возвращены результаты и причина фильтрации. Для данной детекции не будут выполняться дальнейшие политики.

Кроме того, в группе проверок «face_quality» доступны некоторые проверки, которые отсутствуют в проверке изображений на соответствие стандарту (см. ниже).

7.10.3 Таблица сравнения доступных проверок

Для ресурса «/iso» и группы проверок «face_quality» ресурсов «/handlers» и «/verifiers» доступны следующие проверки:

Описание проверок	Наименование проверок	Ресурсы «/iso»	«face_quality»
Проверка качества изображения	illumination_quality, specularity_quality, blurriness_quality, dark_quality, light_quality	+	+
Проверка фона	background_lightness, background_uniformity	+	+
Проверка равномерности освещения по стандарту ICAO	illumination_uniformity	-	+
Проверка положения головы	head_yaw, head_pitch, head_roll	+	+
Проверка направления взгляда	gaze_yaw, gaze_pitch	+	+
Проверка атрибутов рта	mouth_smiling, mouth_occluded, mouth_open	+	+

Описание проверок	Наименование проверок	Ресурсы «/iso»	«face_quality»
Проверка состояния улыбки	smile_properties (none, smile_lips, smile_teeth)	+	+*
Проверка состояния очков	glasses	+	+
Проверка атрибутов глаз	left_eye (open, occluded, closed), right_eye (open, occluded, closed)	+	+
Проверка расстояния между центрами глаз	eye_distance	+	+
Проверка естественности освещения	natural_light (0, 1)	+	+
Проверка бочкообразной дисторсии (эффект Fisheye)	radial_distortion (0, 1)	+	+
Проверка эффекта красных глаз	red_eyes (0, 1)	+	+
Проверка состояния бровей	eyebrows_state (neutral, raised, squinting, frowning)	+	+*
Проверка наличия и типа головного убора	headwear_type (none, baseball_cap, beanie, peaked_cap, shawl, hat_with_ear_flaps, helmet, hood, hat, other)	+	+*
Проверка положения головы по вертикали/горизонтали	head_horizontal_-center, head_vertical_center	+	+
Проверка вертикального/горизонтального размеров головы	head_width, head_height	+	+
Проверка формата изображения	image_format	+	+
Проверка типа цвета по лицу	face_color_type (color, grayscale, infrared)	+	+

Описание проверок	Наименование проверок	Ресурсы «/iso»	«face_quality»
Проверка положения плеч	shoulders_position	+	+
Проверка размера изображения	image_size	-	+
Проверки отступов от краёв фото	indent_upper, intent_lower, intent_right, intent_left	-	+
Проверки ширины/высоты изображения	image_width, image_height	-	+
Проверка соотношения сторон изображения	aspect_ratio	-	+
Проверки ширины/высоты лица	face_width, face_height	-	+
Проверка динамического диапазона	dynamic_range	-	+
Перекрытие лица	face_occlusion, lower_face_-occlusion, forehead_occlusion, nose_occlusion	-	+

* Для данных проверок можно указывать несколько параметров.

7.11 Проверки работоспособности сервисов (health checks)

Для проверок работоспособности сервисов предназначен ресурс «/healthcheck». Ресурс может использоваться для активной проверки состояния сервиса, а именно, может ли сервис выполнять свои функции в полном объёме или нет. Проверяется возможность подключения данного сервиса к сервисам LP и БД, от которых он зависит.

Возможно настроить периодическую проверку ресурса с использованием HAProxy, NGINX или иной системы. Это позволит определить, что сервис недоступен, и принять решение о его отключении из контура или перезапуске.

С помощью опции «include_luna_services» можно включать и отключать проверку health check для сервисов LUNA PLATFORM, от которых зависит данный сервис. Если опция включена, то отправляются дополнительные запросы на ресурсы «/healthcheck» этих сервисов. Опция «include_luna_services» отключается, чтобы не выполнять рекурсивную проверку одних и тех же сервисов. Например, когда сразу несколько сервисов, от которых зависит данный сервис, будут отправлять запросы в сервис Faces и тем самым увеличивать нагрузку на него.

При успешном выполнении проверки health check возвращается только время выполнения подключения в поле «execution_time».

При недоступности одного или нескольких сервисов возвращается ошибка с кодом 502 «Unhealthy». В теле ответа перечисляются компоненты, статусы проверок и возникшие ошибки. Код ошибки 500 в ответе не обязательно означает проблему в работе сервиса. Долгий запрос может завершиться с ошибкой из-за превышения таймаутов, повышенной нагрузки на сервер, проблем в сети или по иным причинам.

При выполнении запроса на ресурс «/healthcheck» рекомендуется выставлять таймаут в несколько секунд. Если запрос не успевает отработать, это признак того, что при работе системы возникли проблемы.

См. описание ресурса «/healthcheck» в спецификации OpenAPI требуемого сервиса LUNA PLATFORM.

> Для проверки работоспособности сервисов также доступен запрос «get health (redirect)», позволяющий не указывать версию API в запросе.

7.12 Загрузка изображений из папки

Скрипт «folder_uploader.py» загружает изображения из указанной папки и обрабатывает загруженные изображения в соответствии с заранее прописанными параметрами.

7.12.1 Основная информация о скрипте

Скрипт «folder_uploader.py» можно использовать для загрузки изображений только с помощью сервиса API.

Нельзя задать адрес и порт Backport 4 для использования этого скрипта. Можно использовать данные, загруженные в LP 5 API в запросах Backport 4.

Нельзя использовать скрипт «folder_uploader.py» для загрузки данных в сервис Backport 3, т.к. созданные для Backport 3 объекты отличаются (например, объект «person» не создается скриптом).

7.12.2 Использование скрипта

Порядок работы скрипта:

1. Поиск изображений допустимого типа (форматы: «.jpg», «.jpeg», «.png», «.bmp», «.ppm», «.tif», «.tiff»; цветовая модель: RGB, CMYK) в указанной папке (источнике).
2. Запуск асинхронной обработки изображения в соответствии с заданными параметрами (см. раздел «Запуск скрипта»).

Процесс обработки изображения:

1. Обнаружение лиц и создание биометрических образцов.
2. Извлечение атрибутов.
3. Создание лиц и прикрепление их к списку.
4. Добавление записи в файл логов.

Если изображение было загружено успешно, будет добавлена следующая запись в {start_upload_time}_success_log.txt: success load logfile. Структура записи следующая:

```
{
"image name": ...,
"face id": [...]
}.
```

При возникновении ошибок на каком-либо этапе обработки скрипта обработка изображения прерывается и добавляется запись в лог-файл с ошибками {start_upload_time}_error_log.txt: error. Структура записи следующая:

```
{
"image name": ...,
"error": ...
}
```

7.12.3 Установка зависимостей

Перед запуском скрипта необходимо установить все требуемые для его запуска зависимости.

> Крайне рекомендуется создать виртуальное окружение для установки зависимостей python.

Небходимо установить пакеты Python (версия 3.7 и более поздние) перед запуском установки. Эти пакеты не поставляются в пакете дистрибутива, их установка не описана в данном руководстве:

- python3.7
- python3.7-devel

Установите gcc.

```
yum -y install gcc
```

Откройте директорию со скриптом.

```
cd /var/lib/luna/current/extras/utils/folder_uploader
```

Создайте виртуальное окружение.

```
python3.7 -m venv venv
```

Активируйте виртуальное окружение.

```
source venv/bin/activate
```

Установите библиотеку tqdm.

```
pip3.7 install tqdm
```

Установите библиотеки luna3.

```
pip3.7 install ../luna3*.whl
```

Отключите виртуальное окружение.

```
deactivate
```

7.12.4 Запуск скрипта

Для запуска скрипта необходимо использовать следующую команду (виртуальное окружение должно быть активировано):

```
python3.7 folder_uploader.py --account_id 6d071cca-fda5-4a03-84d5-5
    bea65904480 --source "Images/" --warped 0 --descriptor 1 --origin http
    ://127.0.0.1:5000 --avatar 1  --list_id 0dde5158-e643-45a6-8a4d-
    ad42448a913b --name_as_userdata 1
```

> Убедитесь в том, что параметр `--descriptor` установлен на `1`, чтобы биометрические шаблоны создавались.

> Версия API установлена в значение «6» по умолчанию, её нельзя изменить с помощью аргументов командной строки.

`--source "Images/"` — «Images/» — папка с изображениями, расположенными рядом со скриптом «folder_uploader.py». Можно также задать полный путь к директории

`--list_id 0dde5158-e643-45a6-8a4d-ad42448a913b` — задание существующего списка

`--account_id 6d071cca-fda5-4a03-84d5-5bea65904480` — задание требуемого ID аккаунта

`--origin http://127.0.0.1:5000` — задание текущего адреса и порта сервиса API

См. help для более подробной информации о доступных аргументах скрипта:

```
python3.7 folder_uploader.py --help
```

Аргументы командной строки:

- account_id: ID аккаунта, используемый в запросах в LUNA PLATFORM (**требуется**)
- source: директория с изображениями для загрузки (**требуется**)
- warped: является ли изображение биометрическим образцом (0,1) (**требуется**)
- descriptor: извлекать ли биометрический шаблон (0,1); по умолчанию — 0
- origin: адрес API; по умолчанию — «http://127.0.0.1:5000»
- avatar: использовать ли биометрический образец как аватар (0,1); по умолчанию — 0

- list_id: ID списка, к которому прикрепляются лица (новый список LUNA будет создан, если list_id не задан и list_linked=1); по умолчанию — None
- list_linked: прикреплять ли лица к списку (0,1); по умолчанию — 1
- list_userdata: пользовательские данные для списка, к которому прикрепляются лица (для нового списка); по умолчанию — None
- pitch_threshold: максимальный угол головы вверх/вниз [0..180];
- roll_threshold: максимальный угол отклонения головы вправо/влево [0..180];
- yaw_threshold: максимальный угол поворота головы вправо/влево [0..180];
- multi_face_allowed: разрешено ли обнаружение нескольких лиц на одном изображении (0,1); по умолчанию — 0
- get_major_detection: выбрать ли основное обнаружение лица посредством «манхеттенского расстояния» на одном изображении (0,1); по умолчанию — 0
- basic_attr: извлекать ли базовые атрибуты (0,1); по умолчанию — 1
- score_threshold: пороговое значение качества биометрического шаблона (0..1); по умолчанию — 0
- name_as_userdata: использовать ли имя изображения в качестве пользовательских данных (0,1); по умолчанию — 0
- concurrency: число параллельной обработки изображений; по умолчанию — 10

7.13 Клиентская библиотека

7.13.1 Основная информация

Архив с клиентской библиотекой для LUNA PLATFORM 5 поставляется в пакете дистрибутива: `/var/lib/luna/current/extras/utils/luna3-*.whl`

Данная библиотека Python является HTTP клиентом для всех сервисов LUNA PLATFORM.

В документе `/var/lib/luna/current/docs/ReferenceManuals/APIReferenceManual.html` можно найти примеры использования библиотеки.

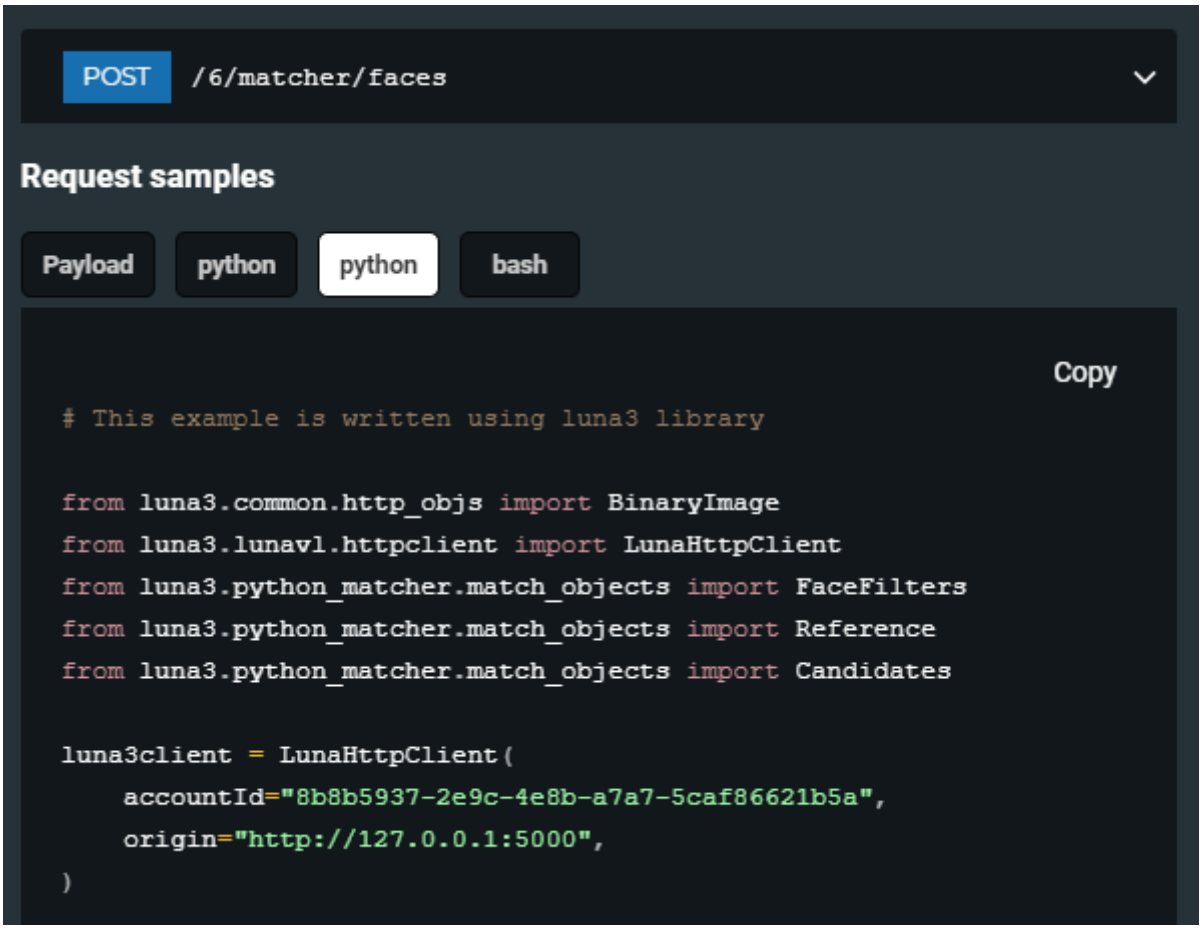


Рис. 58: Пример использования библиотеки Luna3

В примере показан запрос на сравнение лиц. Библиотека luna3 используется для создания запроса. См. «matcher» > «matching faces» в «APIReferenceManual.html»:

```
    # This example is written using luna3 library {#this-example-is-written-
      using-luna3-library}
```

```
from luna3.common.http_objs import BinaryImage
from luna3.lunavl.httpclient import LunaHttpClient
from luna3.python_matcher.match_objects import FaceFilters
from luna3.python_matcher.match_objects import Reference
from luna3.python_matcher.match_objects import Candidates

luna3client = LunaHttpClient(
    accountId="8b8b5937-2e9c-4e8b-a7a7-5caf86621b5a",
    origin="http://127.0.0.1:5000",
)

# create sample {#create-sample}
sampleId = luna3client.saveSample(
    image=BinaryImage("image.jpg"),
    raiseError=True,
).json["sample_id"]

attributeId = luna3client.extractAttrFromSample(
    sampleIds=[
        sampleId,
    ],
    raiseError=True,
).json[0]["attribute_id"]

# create face {#create-face}
faceId = luna3client.createFace(
    attributeId=attributeId,
    raiseError=True,
).json["face_id"]

# match {#match}
candidates = Candidates(
    FaceFilters(
        faceIds=[
            faceId,
        ]
    ),
    limit=3,
    threshold=0.5,
)
reference = Reference("face", faceId)

response = luna3client.matchFaces(
    candidates=[candidates], references=[reference],
    raiseError=True,
```

```
)

print(response.statusCode)
print(response.json)
```

7.13.2 Пример установки библиотеки

В данном примере создается виртуальное окружение для установки luna3.

Данную библиотеку Python можно использовать на Windows, Linux, MacOS.

Установите пакеты Python (версия 3.7 и позже) перед началом установки. Эти пакеты не поставляются в пакете дистрибутива, их установка не описана в данном руководстве:

- python3.7
- python3.7-devel

Установите gcc.

```
yum -y install gcc
```

Перейдите в директорию со скриптом, например, `folder_uploader.py`

```
cd /var/lib/luna/current/extras/utils/folder_uploader
```

Создайте виртуальное окружение.

```
python3.7 -m venv venv
```

Активируйте виртуальное окружение.

```
source venv/bin/activate
```

Установите библиотеки luna3.

```
pip3.7 install ../luna3*.whl
```

Деактивируйте виртуальное окружение.

```
deactivate
```

7.14 Плагины

Плагины используются для выполнения второстепенных действий для различных нужд пользователя. Например, можно создать свой собственный ресурс на основе абстрактного класса или описать, что нужно сделать в каком-либо ресурсе в дополнение к стандартной функциональности.

> Файлы с базовыми абстрактными классами находятся в папке `.plugins/plugins_meta` конкретного сервиса.

Плагины должны быть написаны на языке программирования Python.

Доступно три типа плагинов:

- Плагины событий
- Фоновые плагины
- Плагины сравнения

7.14.1 Плагины событий

Первый тип запускается при выполнении события. Плагин должен реализовывать функцию обратного вызова. Эта функция вызывается для каждого события соответствующего типа. Набор типов событий определяется разработчиками сервиса. Для сервиса Remote SDK доступны два подтипа плагинов событий:

- Мониторинг события
- Отправка события

> Для других сервисов доступен только мониторинг события.

Примеры плагина события приведены в документе «PythonMatcherDevelopmentManual».

7.14.2 Фоновые плагины

Второй тип плагинов предназначен для фоновой работы. Фоновый плагин может реализовывать:

- индивидуальный запрос на конкретный ресурс (route),
- фоновый мониторинг ресурсов сервисов,
- совместную работу плагина событий и фонового плагина (группирование точек мониторинга),
- подключение к другим источникам данных (Redis, RabbitMQ) и обработка их данных.

Примеры фонового плагина приведены в документе «PythonMatcherDevelopmentManual».

7.14.3 Плагины сравнения

Третий тип плагинов позволяет значительно ускорить выполнение запросов на сравнение.

> Важно, что плагины не предоставляются как готовое решение для матчинга. Требуется дополнительно реализовать логику, необходимую для решения конкретных бизнес-задач.

> Плагины сравнения активируются в сервисе Python Matcher Proxy. Данный сервис не устанавливается по умолчанию. Следует запустить его для работы плагинов. См. команду запуска Python Matcher Proxy в разделе «Использование Python Matcher с Python Matcher Proxy» руководства по установке LUNA PLATFORM.

При обычном сценарии работы LUNA PLATFORM с использованием Python Matcher Proxy все запросы на сравнение, обрабатываемые сервисом Python Matcher Proxy, перенаправляются на сервис Python Matcher. В этом случае обработка запросов на сравнение может идти медленнее по нескольким причинам:

- из-за большого объема данных и невозможности ускорить запрос любыми изменениями конфигурации базы данных;
- из-за способа хранения данных — биометрический шаблон и идентификатор объекта (face_id/event_id) хранятся в разных таблицах базы данных. Фильтры, указанные в запросе на сравнение, также могут быть представлены в отдельной таблице базы данных, что замедляет обработку запроса;
- из-за внутренних ограничений базы данных.

С помощью плагинов сравнения можно отделить некоторые группы запросов и ускорить их обработку, в том числе путем передачи данных в другое хранилище с опредленным способом хранения данных, которое делает возможным выполнять сравнение быстрее, чем со способом по умолчанию (см. «Источники данных плагина»).

Примеры:

- запросы на сравнение, в которых все лица (для случая, когда все подходящие кандидаты являются лицами) связаны с одним списком, а любые другие фильтры не указываются в запросе.

  В этом случае можно скопировать необходимых данные для кандидатов матчинга в другое хранилище данных, отличное от хранилища данных по умолчанию, и создать плагин сравнения, который будет сравнивать только указанные эталоны с этими кандидатами, а не с любыми другими объектами.

  Обработка запроса на сравнение будет выполняться быстрее, чем при способе по умолчанию, потому что плагин не будет тратить время на выбор лиц, связанных со списком, от всех лиц, хранящихся в базе данных.

- запросы на сравнение, где все кандидаты являются событиями, задаётся только один фильтр по «event_ids», и требуется использовать только биометрические шаблоны тел.

  В этом случае можно дублировать все event_id и их биометрические шаблоны тел в отдельное хранилище данных, отличное от хранилища данных по умолчанию, и создать плагин сравнения, который будет сравнивать указанный эталон с этими кандидатами, а не с другими объектами.

  Обработка запроса на сравнение будет быстрее по сравнению со способом по умолчанию, потому что плагин не будет тратить время на отделение событий с телами от всех событий и использовать фильтры.

Можно использовать **встроенные плагины сравнения** или создать **пользовательские плагины сравнения**.

Доступно три примера встроенных плагина сравнения:

- плагин «Thin event», использующийся для быстрого сравнения лиц с упрощенными событиями;
- плагин «Thin face», использующийся для быстрого сравнения лиц с упрощенными лицами;
- плагин «Cached Matcher», использующийся для быстрого сравнения лиц по большим спискам.

Thin event

Плагин использует собственную таблицу в базе данных «luna_events».

Ниже приведены несколько особенностей, которые ускоряют сравнение с помощью плагина по сравнению со способом по умолчанию:

- база данных «Thin event» содержит меньше полей данных;
- база данных «Thin event» хранит «event_id» и БШ лица в одной таблице;
- база данных «Thin event» хранит возраст, пол и некоторые другие фильтры в одной таблице.

Thin face

Плагин использует собственную таблицу в базе данных «luna_faces» с тремя обязательными столбцами («face_id», «descriptor», «descriptor_version») и рядом дополнительных, которые можно настроить: «account_id», «lists», «create_time», «external_id», «user_data», «event_id», «avatar».

Cached Matcher

Сопоставление лиц по списку — трудоемкий процесс, поэтому для повышения производительности с помощью плагина реализованы следующие методы:

- для сравнения кандидатов и ссылок используется отдельный сервис «LUNA-CACHED-MATCHER»;
- данные для кандидата («list_id», «face_id», ”descriptor:) хранятся в кеше памяти. Это обеспечивает быстрый доступ к данным;

- данные разбиты по горизонтали на сегменты (в качестве сегмента используется сервис «LUNA-CACHED-MATCHER»), что обеспечивает быстрый поиск совпадающих результатов.

См. подробное описание встроенных плагинов и инструкцию по написанию пользовательских плагинов в документе «PythonMatcherDevelopmentManual» в комплекте поставки.

7.14.3.1 Общее описание работы плагина

Каждый запрос на сравнение представлен в виде всех возможных комбинаций кандидатов и эталонов, затем каждая такая комбинация обрабатывается как отдельный подзапрос следующим образом (дополнительный подзапрос означает комбинацию эталонов и кандидатов):

- Получение сложности подзапроса (см. «Затраты ресурсов на сравнение»).
- Выбор способа обработки подзапроса: с помощью плагина сравнения или сервиса Python Matcher.
  - Если на предыдущем шаге был выбран сервис Python Matcher, он обработает подзапрос и вернет ответ сервису Python Matcher Proxy.
  - Если на предыдущем шаге был выбран плагин сравнения, то он обработает подзапрос. Если подзапрос успешно обработан, ответ возвращается в сервис Python Matcher Proxy. Если подзапрос не был успешно обработан, будет совершена попытка обработки в сервисе Python Matcher.
- Если запрос был успешно обработан плагином сравнения, но у него нет доступа ко всем полям объекта, указанным в позапросе в поле «target» для сравнения, то сервис Python Matcher Proxy получит эти данные перед следующим шагом.
- Сервис Python Matcher Proxy собирает результаты от всех подзапросов, сортирует их в правильном порядке, и выдает ответ пользователю.

Рабочий процесс плагина сравнения показан ниже:

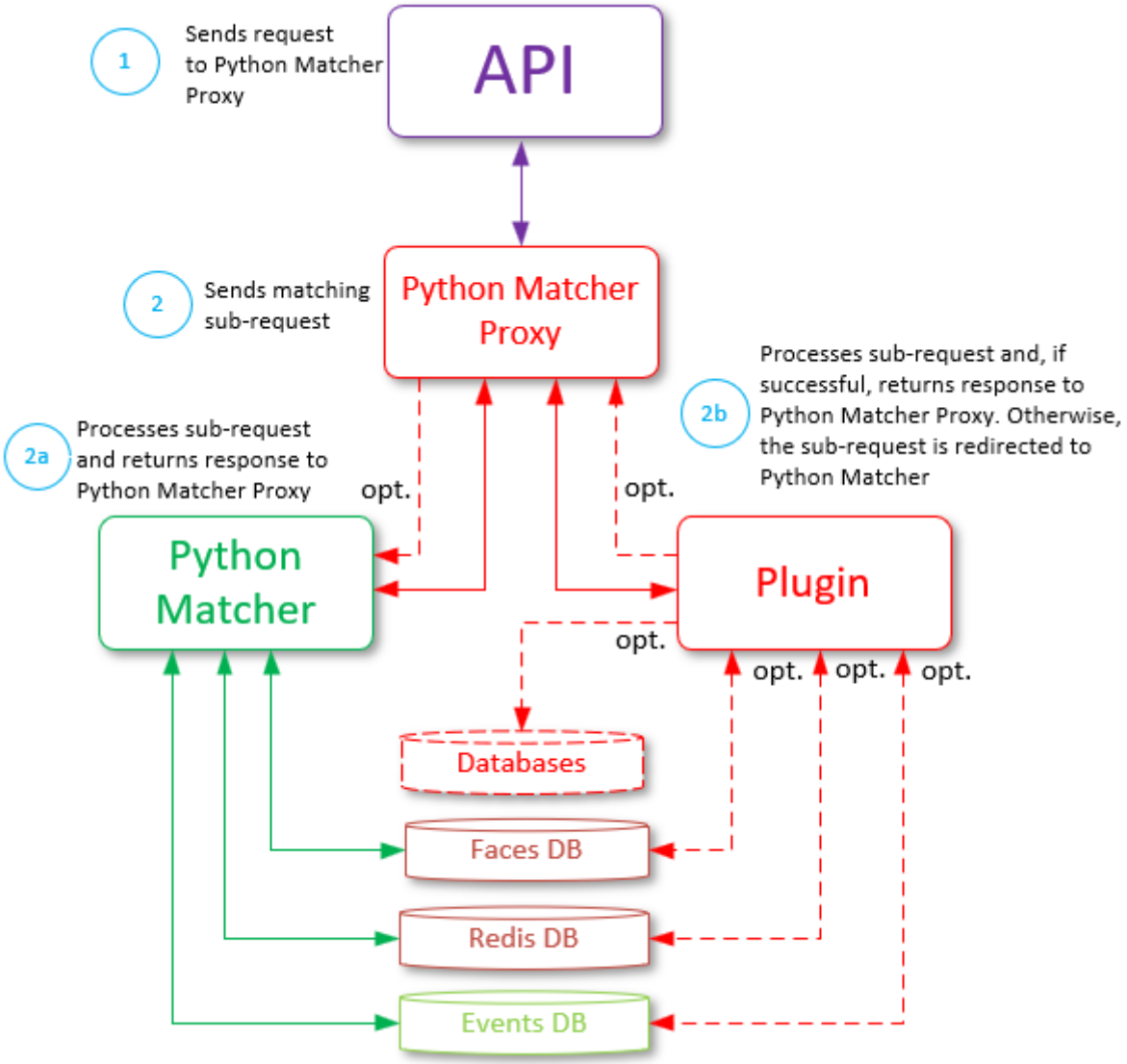


Рис. 59: Рабочий процесс плагина сравнения

7.14.3.2 Сложность запроса на сравнение

Matching cost — это число с плавающей запятой, определяющее сложность запроса на сравнение с использованием плагина. Определение сложности сравнения необходимо для выбора наилучшего способа обработки запроса на сравнение: с помощью сервиса Python Matcher или с помощью одного или нескольких плагинов.

Значение сложности запроса на сравнение для сервиса Python Matcher равно 100. Если существует несколько плагинов, то соответствующее значение сложности будет рассчитано для каждого плагина. Будет использоваться плагин сравнения с наименьшим значением сложности, если его сложность ниже, чем сложность запроса для сервиса Python Matcher. Все запросы со сложностью более 100 будут обрабатываться в сервисе Python Matcher. Если плагинов сравнения нет, то для

обработки запроса будет использоваться сервис Python Matcher.

7.14.3.3 Поля target, выступающие в качестве сравнения

Сервис Python Matcher имеет доступ ко всем данным сущностей сравнения, поэтому он может обрабатывать запросы на сравнение со всеми полями target. В свою очередь, плагины сравнения могут не иметь доступа к данным, указанным в поле target запроса. В этом случае сервис Python Matcher Proxy дополнит ответ плагина сравнения отсутствующими данными о полях target, например:

- ответ на сравнение содержит следующие поля target: face_id , user_data и similarity, а выбранный плагин сравнения не имеет доступа к полю user_data, тогда:
  - плагин сравнения сравнивает эталон с указанными face_id и возвращает результат сравнения сервису Python Matcher Proxy, который содержит только пары face_id и similarity.
  - для каждого кандидата на сравнение в результате сервис Python Matcher Proxy получит user_data из основной базы данных по face_id и объединит face_id и similarity с user_data.
  - плагин сравнения вернет пользователю расширенный ответ с указанными полями target и ключевым полем face_id в качестве target. Эта механика требует, чтобы плагин поддерживал соответствующий идентификатор объекта в качестве target. Если плагин не поддерживает идентификатор сущности в качестве target, такой запрос не будет отправлен этому плагину.
- ответ на сравнение содержит следующие поля target: age и gender, а выбранный плагин сравнения имеет доступ только к полям event_id , descriptor и age.
  - плагин сравнения сравнивает эталон и возвращает соответствующий ответ в сервис Python Matcher Proxy, который содержит только пары event_id , age и similarity.
  - для каждого кандидата на сравнение в результате сервис Python Matcher Proxy получит gender из основной базы данных по event_id и объединит event_id с age, а также после этого удалит ненужные event_id и similarity из ответа.
  - плагин сравнения вернет пользователю расширенный ответ с указанными полями target и ключевым полем event_id в качестве target. Эта механика требует, чтобы плагин поддерживал соответствующий идентификатор объекта в качестве target. Если плагин не поддерживает идентификатор сущности в качестве target, такой запрос не будет отправлен этому плагину.

7.14.3.4 Источники данных плагина

Для ускорения обработки запроса каждый плагин сравнения может использовать отдельный источник данных вместо источника по умолчанию (базы данных Events, Faces или Attributes (см. раздел «Описание баз данных»)), например использовать отдельную базу данных, новую таблицу в существующей базе данных, кеш в памяти и т.д.

Более подробную информацию о плагинах сравнения см. в документе «PythonMatcherDevelopmentManual».

7.14.4 Использование плагинов

7.14.4.1 Ручное добавление плагинов в директорию

Добавлять плагины в директорию вручную можно, когда плагин не требует дополнительных зависимостей, которые не предусмотрены в Docker-контейнере сервиса.

Чтобы использовать плагин вместе с сервисом в Docker-контейнере, необходимо выполнить два шага:

- Добавить файл плагина в контейнер.
- Указать использование плагина в настройках контейнера.

При запуске контейнера необходимо переслать файл плагина в папку с плагинами конкретного сервиса. Например, для сервиса Remote SDK это будет папка `/srv/luna_remote_sdk/plugins`.

Сделать это можно любым удобным способом. Например, при запуске сервиса можно смонтировать папку с плагинами в каталог нужного сервиса (см. команды запуска сервиса в руководстве по установке):

Необходимо добавить следующий том, если все необходимые плагины для сервиса хранятся в директории «/var/lib/luna/remote_sdk/plugins»:

```
  -v /var/lib/luna/remote_sdk/plugins:/srv/luna_remote_sdk/plugins/ \
```

Данная команда приведена для случая ручного запуска сервиса.

Затем нужно добавить имя/имена файла в настройку «LUNA_REMOTE_SDK_ACTIVE_PLUGINS» в сервисе Configurator.

```
[
   "plugin_1",
   "plugin_2",
   "plugin_3"
]
```

> Список должен содержать имена файлов без расширения (.py).

После выполнения этих шагов LP будет автоматически использовать плагин(ы).

> Дополнительную информацию о настройках для конкретного сервиса можно найти в руководстве разработчика сервиса API.

7.14.4.2 Создание нового Docker-контейнера с помощью плагина

Создавать новый Docker-контейнер с помощью плагина можно, когда для использования плагина требуются дополнительные зависимости или когда для производственного использования требуется контейнер с плагином.

Необходимо создать свой Docker-контейнер на основе базового контейнера сервиса.

Нужно добавить «Dockerfile» со следующей структурой в CI:

```
FROM dockerhub.visionlabs.ru/luna/luna-remote-sdk:v.0.15.6
USER root
...
USER luna
```

`FROM` должен включать адрес базового контейнера сервиса, который будет использоваться.

`USER root` — изменение прав пользователя на root для выполнения следующих действий. После изменения прав должны быть перечислены команды для добавления плагина и его зависимости. Они не приводятся в данном руководстве. См. документацию Docker.

`USER luna` — после выполнения всех команд пользователь должен быть снова изменен на «luna».

Нужно добавить имя файла плагина в настройку «LUNA_REMOTE_SDK_ACTIVE_PLUGINS» в сервисе Configurator.

Доступны следующие возможности:

- Обновление настройки вручную в сервисе Configurator в соответствии с вышеописанным способом.
- Создание файла дампа с настройками плагина LP и добавление его в сервис Configurator после запуска.

Пример файла дампа с настройками плагина Remote SDK приведен ниже.

```
{
    "settings":[
        {
            "value": [
```

```
                            "plugin_1",
                            "plugin_2",
                            "plugin_3"
                        ],
                "description": "list active plugins",
                "name": "LUNA_REMOTE_SDK_ACTIVE_PLUGINS",
                "tags": []
            },
        ]
    }
```

Затем файл применяется с помощью нижеописанной команды. Например, файл хранится в «/var/lib/luna/». Имя файла дампа — «luna_remote_sdk_plugin.json».

```
    docker run \
    -v /var/lib/luna/luna_remote_sdk_plugin.json:/srv/luna_configurator/
       used_limitations/luna_remote_sdk_plugin.json \
    --network=host \
    --rm \
    --entrypoint=python3 \
    dockerhub.visionlabs.ru/luna/luna-configurator:v.2.2.82 \
    ./base_scripts/db_create.py --dump-file /srv/luna_configurator/
       used_limitations/luna_remote_sdk_plugin.json
```

7.15 Мониторинг

В LUNA PLATFORM есть несколько методов мониторинга:

- Отправка данных в InfluxDB (включен по умолчанию)
- Экспорт метрик в формате Prometheus через ресурс `/metrics` (отключен по умолчанию)

7.15.1 Отправка данных в InfluxDB

Начиная с версии 5.5.0, в LUNA PLATFORM предоставляется возможность использовать InfluxDB версии 2.

> При необходимости можно выполнить миграцию с версии 1 на версию 2 с помощью встроенных инструментов. См. документацию InfluxDB

Для работы с InfluxDB необходимо зарегистрироваться с помощью логина и пароля и указать имя бакета, имя организации и токен. Все эти данные задаются при запуске контейнера InfluxDB с помощью переменных окружения.

Для использования мониторинга необходимо в настройках каждого сервиса задать для полей `bucket`, `organization`, `token` точно такие же данные, которые указывались при запуске контейнера InfluxDB. Так, например, если при запуске контейнера InfluxDB использовались следующие настройки…:

```
  -e DOCKER_INFLUXDB_INIT_BUCKET=luna_monitoring \
  -e DOCKER_INFLUXDB_INIT_USERNAME=luna \
  -e DOCKER_INFLUXDB_INIT_PASSWORD=password \
  -e DOCKER_INFLUXDB_INIT_ORG=luna \
  -e DOCKER_INFLUXDB_INIT_ADMIN_TOKEN=kofqt4Pfqjn6o \
```

… то в настройках каждого сервиса должны быть указаны следующие параметры:

```
  "influxdb": {
      "organization": "luna",
      "token": "kofqt4Pfqjn6o",
      "bucket": "luna_monitoring",
```

> По умолчанию значения для переменных окружения контейнера и значения настроек сервисов совпадают. Это означает, что если LUNA PLATFORM запускается с помощью руководства по установке или скрипта Docker Compose, то мониторинг будет автоматически включен.

> Логин и пароль используются для доступа к пользовательскому интерфейсу InfluxDB.

Также доступна возможность запуска InfluxDB на отдельном сервере. Адрес сервера с InfluxDB необходимо указать в параметрах `host` и `port` в настройках сервисов.

> См. остальные настройки для InfluxDB на примере сервиса API в разделе «LUNA_MONITORING».

7.15.1.1 Отправляемые данные

Мониторинг возможен для двух типов событий: запрос (все запросы) и ошибка (только неудавшиеся запросы).

Каждое событие — это точка временного ряда. Для сервиса API точка представлена с использованием следующих данных:

- тип события (запросы или ошибки)
- отметка времени начала запроса
- теги
- поля

> Для других сервисов набор типов событий может отличаться. Например, сервис Remote SDK также собирает данные об использовании SDK, выполненных оценках и лицензировании.

Тег — это индексированные данные в хранилище. Он представлен в виде словаря, где

- keys — имена тегов (строка),
- values — строка, целое число или число с плавающей запятой.

Поле представляет собой неиндексированные данные в хранилище. Оно представлено в виде словаря, где

- keys — имена полей (строка),
- values — строка, целое число или число с плавающей запятой.

Ниже приведен пример тегов и полей для сервиса API. Нижеперечисленные теги уникальны для каждого сервиса. Информацию о мониторинге конкретного сервиса можно найти в соответствующей документации разработчика:

- «API»
- «Licenses»
- «Configurator»
- «Image Store»
- «Faces»
- «Admin»
- «Backport 3»
- «Backport 4»
- «Events»
- «Remote SDK»

- «Handlers»
- «Python Matcher»
- «Sender»
- «Tasks»
- «Lambda»

Сохранение данных для запросов запускается при каждом запросе. Каждая точка содержит данные о соответствующем запросе (время выполнения и т.д.).

- теги

Имя тега	Описание
service	сервис, всегда «luna-api»
account_id	идентификатор аккаунта или пусто
route	объединение метода запроса и ресурса запроса (POST:/extractor)
status_code	HTTP статус код ответа

- поля

Имя поля	Описание
request_id	идентификатор запроса
execution_time	время выполнения запроса

Сохранение данных для ошибок запускается при сбое запроса. Каждая точка содержит *error_code* ошибки LUNA.

- теги

Имя тега	Описание
request_id	идентификатор запроса,всегда «luna-api»
account_id	идентификатор аккаунта или пусто
route	объединение метода запроса и ресурса запроса (POST:/extractor)
status_code	HTTP статус код ответа
error_code	код ошибки LUNA PLATFORM

- поля

Имя поля	Описание
request_id	идентификатор запроса

Каждый обработчик может добавлять дополнительные теги или поля. Например, обработчик ресурса «/handlers/{handler_id}/events» добавляет тег `handler_id`.

7.15.1.2 Просмотр данных мониторинга

Для просмотра данных мониторинга можно использовать пользовательский интерфейс InfluxDB.

- перейдите в пользовательский интерфейс InfluxDB `<server_ip>:<influx_port>`. Порт по умолчанию — 8086. Данные для входа по умолчанию — luna/password.
- выберите вкладку Explore
- выберите способ отображения информации в выпадающем списке (график, гистограмма, таблица и пр.)
- выберите бакет внизу страницы. По умолчанию — `luna_monitoring`
- отфильтруйте необходимые данные
- нажмите «Submit».

Также InfluxDB версии 2 позволяет визуализировать данные мониторинга с помощью инструмента «LUNA Dashboards (Grafana)».

7.15.1.3 Подсчет статистики выполненных запросов и оценок

LUNA PLATFORM подсчитывает количество выполненных запросов и оценок за месяц по данным мониторинга если включен сбор статистики. Сбор статистики работает только при включенном мониторинге и установленной InfluxDB версии 2.0.8 и более поздних версий.

Для получения статистики следует использовать запрос на ресурс «/luna_sys_info» или перейти в GUI сервиса Admin на вкладку «help» и нажать «Get LUNA PLATFORM system info». Необходимая информация содержится в секции «stats».

В этой секции содержатся два поля — «estimators_stats» и «routes_stats».

Первое поле содержит список выполненных оценок. Для каждой оценки отображается три поля:

- name — наименование выполненной оценки (например, `estimate_emotions`)
- count — суммарное количество выполненных оценок одного типа
- month — месяц, за который был произведен подсчет (например, `2021-09`)

Второе поле содержит список сервисов, к которым выполнялись запросы. Для каждого сервиса отображается пять полей:

- service — наименование сервиса (например, `luna-api`)
- route — метод и запрос (например, `GET:/version`)
- month — месяц, за который был произведен подсчет
- errors — количество запросов, выполненных с конкретной ошибкой (например, `[ { "count": 1, "error_code": "12012"} ]`)
- request_stats — количество успешно выполненных запросов (например, `[ { "count": 1, "status_code": "200"} ]`)

> Информация является обезличенной и содержит только количественные данные.

Статистика подсчитывается в InfluxDB на основе данных из бакета «luna_monitoring» и хранится в бакете «luna_monitoring_aggregated». Бакеты создаются в InfluxDB. Не следует удалять данные из данного бакета, иначе будет невозможно получить статистику.

Статистика подсчитывается раз в сутки, поэтому не отображается сразу после запуска LP.

Задачи для подсчёта статистики можно найти в GUI InfluxDB на вкладке «Tasks». Там можно вручную запустить их выполнение.

Для включения данного функционала следует выполнить команду `python3 ./base_scripts/influx2_cli.py create_usage_task --luna-config http://127.0.0.1:5070/1` после запуска сервиса Admin (см. руководство по установке). Команда автоматически создаёт необходимый бакет «luna_monitoring_aggregated». Если данная команда не выполнена, то в ответе «/luna_sys_info» не будет отображаться статистика.

При необходимости можно отключить сбор статистики, удалив или отключив соответствующие задачи на вкладке «Tasks» в GUI InfluxDB.

7.15.2 Экспорт метрик в формате Prometheus

Каждый сервис LUNA PLATFORM может собирать и сохранять метрики в формате Prometheus в виде данных временных рядов, которые можно использовать для отслеживания поведения сервиса. Метрики могут быть интегрированы в систему мониторинга Prometheus для отслеживания производительности. См. официальную документацию Prometheus для подробной информации.

По умолчанию сбор метрик отключен. Сбор метрик включается в группе параметров «LUNA_SERVICE_METRICS».

> Обратите внимание, что все данные метрик сбрасываются при завершении работы сервиса.

7.15.2.1 Тип метрик

Доступно два типа метрик:

- **счетчики**, которые увеличиваются с каждым событием

- **кумулятивные гистограммы**, которые используются для измерения распределения продолжительности или размера событий

> Кумулятивная гистограмма — это отображение, которое подсчитывает совокупное количество наблюдений во всех интервалах до указанного интервала. См. описание в Wikipedia.

Доступны следующие метрики типа **счетчики**:

- `request_count_total` — общее количество запросов
- `errors_count_total` — общее количество ошибок

Каждый из них имеет как минимум два лейбла для сортировки:

- `status_code` (или `error_code` для метрик ошибок)
- `path` — путь, состоящий из метода запроса и маршрута конечной точки

> Лейблы представляют собой ключевые пары, состоящие из имени и значения, которые присваиваются метрикам.

При необходимости можно добавить пользовательские типы лейблов, указав пару `label_name=label_value` в параметре «extra_labels».

> Обратите внимание, что пара `label_name=label_value` будет добавлена к каждому сервису LUNA PLATFORM.

Специальный менеджер распределяет все запросы, проходящие через сервис, среди счетчиков, используя эти метки. Это обеспечивает, что два успешных запроса, отправленных на разные конечные точки или на одну конечную точку, но с разными кодами состояния, будут доставлены к различным метрикам.

> Неудачные запросы распределяются по метрикам `request_count_total` и `request_errors_total` .

Метрика `requests` типа **кумулятивные гистограммы** отслеживает длительность запросов к сервису. Для гистограммы определяются следующие интервалы (бакеты), в которые попадают измерения:

- 0.0001
- 0.00025
- 0.0005
- 0.001
- 0.0025
- 0.005
- 0.01
- 0.025
- 0.05
- 0.075

- 0.1
- 0.25
- 0.5
- 0.75
- 1.0
- 2.5
- 5.0
- 7.5
- 10.0
- Inf

Таким образом диапазон времени запроса может быть разбит на несколько интервалов, начиная от очень быстрых запросов (0.0001 секунды) до очень долгих (Inf - бесконечность). У гистограмм также есть метки для классификации данных, такие как `status_code` для статуса запроса или `route` для указания маршрута запроса.

Примеры

Если отправить один запрос в ресурс `/healthcheck`, за которым последуют три запроса в ресурс `/docs/spec`, один из которых будет перенаправлен (код состояния ответа 301), то при выполнении запроса к ресурсу `/metrics`, в теле ответа будет выдан следующий результат:

```
# HELP request_count_total Counter of requests
# TYPE request_count_total counter
request_count_total{path="GET:/docs/spec",status_code="200"} 2.0
request_count_total{path="GET:/docs/spec",status_code="301"} 1.0
request_count_total{path="GET:/healthcheck",status_code="200"} 1.0
```

Если отправить один неверный POST-запрос к ресурсу `/handlers`, то при выполнении запроса к ресурсу `/metrics`, в теле ответа будет выдан следующий результат:

```
# HELP request_count_total Counter of requests
# TYPE request_count_total counter
request_count_total{path="POST:/handlers",status_code="401"} 1.0
# HELP request_errors_total Counter of request errors
# TYPE request_errors_total counter
request_errors_total{error_code="12010",path="POST:/handlers"} 1.0
# HELP requests Histogram of request time metrics
# TYPE requests histogram
requests_sum{route="GET:/docs/spec",status_code="200"} 0.003174567842297907
requests_bucket{le="0.0001",route="GET:/docs/spec",status_code="200"} 0.0
requests_bucket{le="0.00025",route="GET:/docs/spec",status_code="200"} 0.0
requests_bucket{le="0.0005",route="GET:/docs/spec",status_code="200"} 0.0
```

```
requests_bucket{le="0.001",route="GET:/docs/spec",status_code="200"} 1.0
...
requests_count{route="GET:/docs/spec",status_code="200"} 2.0
requests_sum{route="GET:/docs/spec",status_code="301"} 0.002381476051209132
```

7.15.3 Настройка сбора метрик для Prometheus

Prometheus необходимо настроить на сбор метрик LUNA PLATFORM.

Пример конфигурации Prometheus для сбора метрик сервисов LP:

```
  - job_name: "luna-api"
    static_configs:
      - targets: ["127.0.0.1:5000"]
  ...

  - job_name: "luna-configurator"
    static_configs:
      - targets: ["127.0.0.1:5070"]
```

См. пример запуска Prometheus в официальной документации.

В сервисе LUNA Dashboards уже созданы дашборды Prometheus.

7.15.4 LUNA Dashboards

Инструмент LUNA Dashboards предназначен для визуализации данных мониторинга. LUNA Dashboards на основе веб-приложения Grafana создает набор дашбордов для анализа состояния отдельных сервисов, а также пару обобщенных дашбордов, которые можно использовать для оценки состояния системы.

> Для использования веб-интерфейса Grafana нужно перейти по адресу «http://IP_ADDRESS:3000».

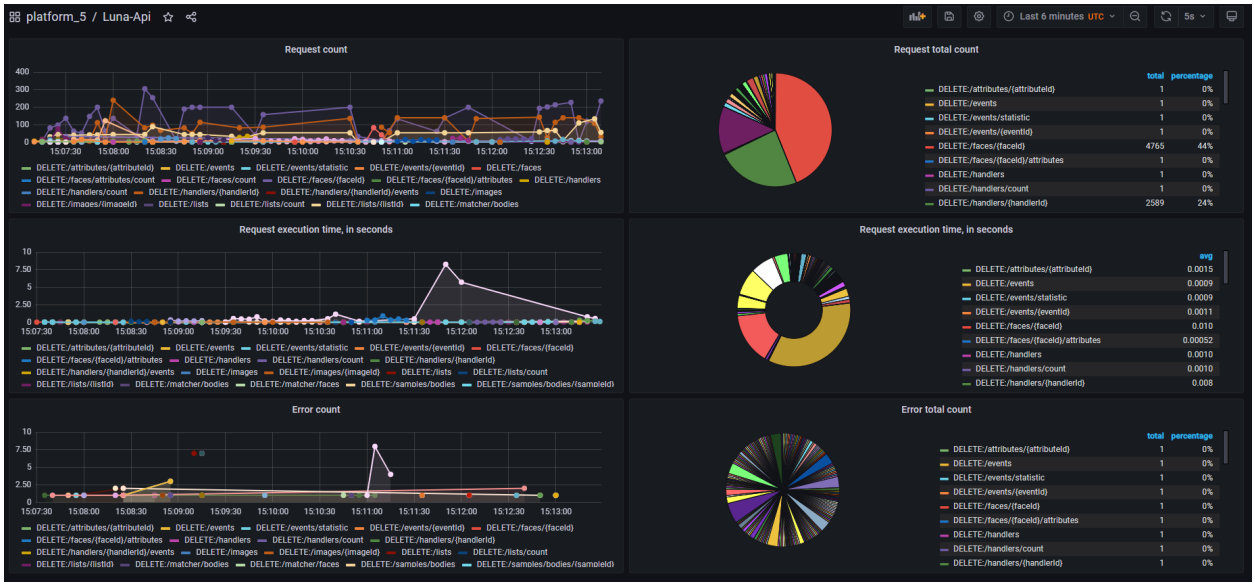


Рис. 60: Дашборд сервиса API при запуске тестирования

В Grafana конфигурирует источник данных, благодаря которому веб-приложение может связываться с базой данных Influx, в которую попадают данные мониторинга.

Рис. 61: Работа LUNA Dashboards

Инструмент LUNA Dashboard может быть полезен:

- для контроля состояния системы;
- для анализа ошибок;
- для получения статистики по ошибкам;
- для анализа нагрузки на отдельные сервисы и LP в целом, нагрузки по дням недели, времени суток и т.д;
- для анализа статистики выполнения запросов, т.е. на какие ресурсы приходится какая доля запросов приходящихся на всю LP;
- для анализа динамики времени выполнения запросов;
- для оценки среднего значения времени исполнения запросов на конкретный ресурс;
- для анализа метрик Prometheus;
- для анализа изменений показателей во времени.

После установки дашбордов (см. ниже) в веб-интерфейсе Grafana становится доступен каталог «luna_platform_5», который содержит следующие дашборды:

- Luna Platform Heatmap,
- Luna Platform Summary,
- Дашборды отдельных сервисов.

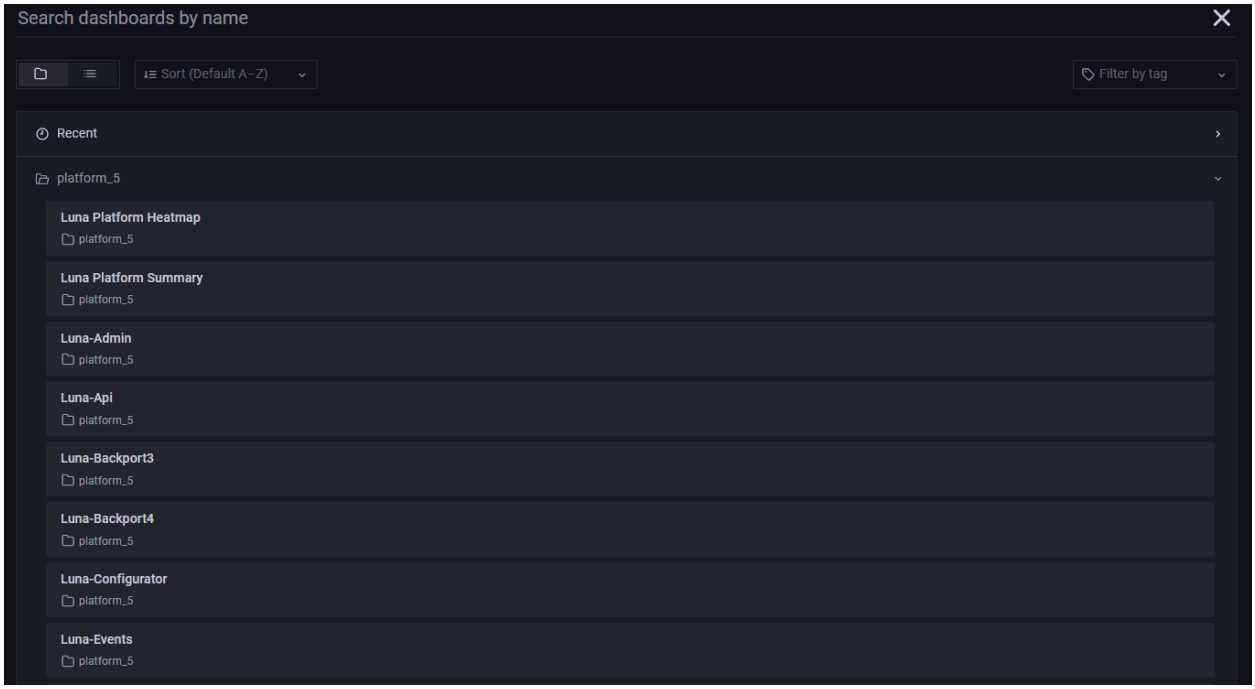


Рис. 62: Структура LUNA Dashboards

Luna Platform Heatmap позволяет оценить нагрузку на систему без конкретного ресурса. В статистике вы можете оценить время активности системы в определенное время.

Luna Platform Summary позволяет получать статистику по запросам для всех сервисов в одном месте, а также оценивать графики по RPS (Request Per Seconds).

Dashboards for individual services позволяет получать информацию о запросах по отдельным ресурсам, ошибкам и статус кодам для каждого сервиса. В таком дашборде будут отображаться не данные о загрузке, а искусственно сгенерированные данные в выбранном временном интервале.

Для использования дашбордов Grafana требуется следующее ПО:

- InfluxDB 2.0 (в настоящее время используется версия 2.0.8-alpine)
- Grafana (в настоящее время используется версия 8.5.20)

InfluxDB и Grafana уже включены в состав дистрибутива. При необходимости можно использовать собственную инсталляцию Grafana или установить ее вручную.

7.15.4.1 Ручная установка LUNA Dashboards

Примечание. Ниже описаны действия для ручной установки LUNA Dashboards. Дашборды можно также автоматически запустить с помощью специальной команды в разделе «Визуализация мониторинга и логов с помощью Grafana» в руководстве по установке.

Установка плагина

Помимо встроенных плагинов Grafana, дашборды также используют плагин piechart. Использовать инструмент grafana-cli для установки piechart-panel можно с помощью следующей команды:

```
grafana-cli plugins install grafana-piechart-panel
```

> При необходимости можно использовать архив «grafana-piechart-panel.zip» в «/var/lib/luna/current/extras/uti

Для применения плагина требуется перезапуск:

```
sudo service grafana-server restart
```

Запуск дашбордов

Дашборды можно запускать вручную или с помощью специального контейнера Grafana с дашбордами — `luna-dashboards`:

- Запуск с использованием специального контейнера Grafana с дашбордами описан в руководстве по установке.
- Запуск вручную описан далее в этом документе.

Установите Grafana. Пример команды приведен ниже:

```
docker run \
--restart=always \
--detach=true \
--network=host \
--name=grafana \
-v /etc/localtime:/etc/localtime:ro \
-e "GF_INSTALL_PLUGINS=grafana-piechart-panel" \
dockerhub.visionlabs.ru/luna/grafana:8.5.20
```

> При необходимости в переменной окружения «GF_INSTALL_PLUGINS» можно указать путь до архива «/var/lib/luna/current/extras/utils/grafana-piechart-panel.zip».

Скрипты для установки плагинов Grafana можно найти в «/var/lib/luna/current/extras/utils/».

> Перед запуском следующего скрипта необходимо установить Python версии 3.7 или новее. Данный пакет не входит в состав дистрибутива, и его установка в данном руководстве не описывается.

Перейдите в каталог luna dashboards.

```
cd /var/lib/luna/current/extras/utils/luna-dashboards_linux_rel_v.*
```

Создайте виртуальную среду.

```
python3.7 -m venv venv
```

Активируйте виртуальную среду.

```
source venv/bin/activate
```

Установите файл luna dashboards.

```
pip install luna_dashboards-*-py3-none-any.whl
```

Перейдите в следующий каталог.

```
cd luna_dashboards
```

> Папка «luna_dashboards» содержит файл конфигурации «config.conf», который включает настройки для Grafana, InfluxDB и периодов мониторинга. По умолчанию файл уже включает настройки по умолчанию, но при необходимости их можно изменить с помощью команды «vi config.conf».

Запустите следующий скрипт для создания дашбордов.

```
python create_dashboards.py
```

Отключите виртуальную среду.

```
deactivate
```

Для использования веб-интерфейса Grafana нужно перейти по адресу «http://IP_ADDRESS:3000», при условии, что контейнеры Grafana и InfluxDB были запущены.

В левом верхнем углу нужно нажать кнопку «General», затем развернуть папку «luna_platform_5» и выбрать необходимый дашборд.

7.15.5 Grafana Loki

Grafana Loki — система агрегации логов, позволяющая гибко работать с логами LUNA PLATFORM в Grafana.

С помощью Grafana Loki можно выполнять следующие задачи:

- сбор логов LUNA PLATFORM
- поиск по логам LUNA PLATFORM
- визуализация логов
- извлечение числовых метрик из логов
- другие

> См. подробную информацию о возможностях Grafana Loki в официальной документации: https://grafana.com/oss/loki/.

Grafana Loki включена в состав дистрибутива LUNA PLATFORM.

Для запуска Grafana Loki требуется следующее ПО:

- запущенная Grafana с настроенным источник данных Loki в Grafana (см. раздел «LUNA Dashboards»)
- запущенный агент доставки логов Promtail (см. раздел «Promtail» ниже)

Таким образом, для работы с логами LUNA PLATFORM в Grafana выполняется следующая цепочка действий:

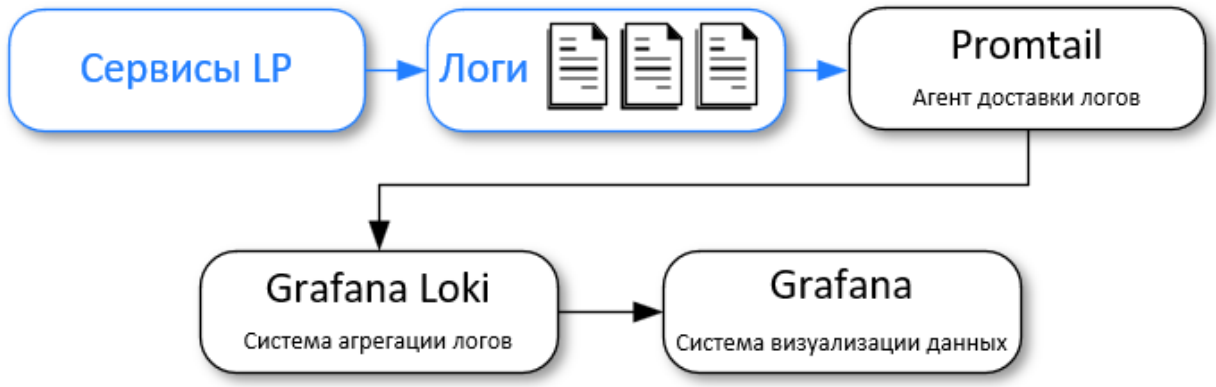


Рис. 63: Схема работы Grafana Loki

> Команды запуска Grafana и Promtail приведены в руководстве по установке. В контейнере Grafana из дистрибутива LUNA PLATFORM уже настроен источник данных Loki.

> Также можно запустить Grafana и Promtail с помощью выполнения дополнительного скрипта Docker Compose после выполнения основного скрипта Docker Compose (см. документ «Развертывание с помощью Docker Compose»).

7.15.5.1 Promtail

Для доставки логов LUNA PLATFORM в Grafana Loki используется специально настроенный агент Promtail. Как и Grafana Loki, агент Promtail включен в состав дистрибутива LUNA PLATFORM.

В комплекте поставки LUNA PLATFORM доступен настроенный файл конфигурации Promtail, предоставляющий возможность фильтрации по следующим меткам:

- уровень логирования LP
- сервисы LP
- URI
- статус коды LP

Некоторые дополнительные метки (например, версия сервиса LP) также могут быть указаны с помощью аргумента `client.external-labels` в команде запуска Promtail.

В Grafana Loki также настроено вторичное поле (см. derived field в официальной документации) для поиска по определенному идентификатору запроса (Request ID) в логах.

7.16 Базы данных

7.16.1 Ручное создание баз данных сервисов в PostgreSQL

В этом разделе описаны команды, необходимые для настройки внешнего PostgreSQL для работы с сервисами LP. Внешний означает, что уже есть рабочая БД и её необходимо использовать с LP.

> Команды для Oracle не приведены в данной документации.

Необходимо указать внешнюю базу данных в конфигурациях сервисов LP.

Для сервисов Faces и Events необходимо добавить дополнительные функции VLMatch в используемую базу данных. Сначала библиотеку VLMatch нужно скомпилировать и перенести в PostgreSQL, а затем добавить функцию VLMatch в базы данных Events и Faces. См. разделы «Компиляция VLMatch для PostgreSQL», «Добавление функции VLMatch для БД Faces в PostgreSQL» и «Добавление функции VLMatch для БД Events в PostgreSQL» для подробной информации.

7.16.1.1 Создание пользователя PostgreSQL

Создайте пользователя базы данных.

```
runuser -u postgres -- psql -c 'create role luna;'
```

Присвойте пользователю пароль.

```
runuser -u postgres -- psql -c "ALTER USER luna WITH PASSWORD 'luna';"
```

7.16.1.2 Создание базы данных Configurator

Предполагается, что пользователь базы данных уже создан.

Создайте базу данных для сервиса Configurator.

```
runuser -u postgres -- psql -c 'CREATE DATABASE luna_configurator;'
```

Присвойте привилегии пользователю базы данных.

```
runuser -u postgres -- psql -c 'GRANT ALL PRIVILEGES ON DATABASE
   luna_configurator TO luna;'
```

Разрешите пользователю авторизацию в базе данных.

```
runuser -u postgres -- psql -c 'ALTER ROLE luna WITH LOGIN;'
```

7.16.1.3 Создание базы данных Accounts

Предполагается, что пользователь базы данных уже создан.

Создайте базу данных для сервиса Accounts.

```
  runuser -u postgres -- psql -c 'CREATE DATABASE luna_accounts;'
```

Присвойте привилегии пользователю базы данных.

```
  runuser -u postgres -- psql -c 'GRANT ALL PRIVILEGES ON DATABASE
     luna_accounts TO luna;'
```

Разрешите пользователю авторизацию в базе данных.

```
  runuser -u postgres -- psql -c 'ALTER ROLE luna WITH LOGIN;'
```

7.16.1.4 Создание базы данных Handlers

Предполагается, что пользователь базы данных уже создан.

Создайте базу данных для сервиса Handlers.

```
  runuser -u postgres -- psql -c 'CREATE DATABASE luna_handlers;'
```

Присвойте привилегии пользователю базы данных.

```
  runuser -u postgres -- psql -c 'GRANT ALL PRIVILEGES ON DATABASE
     luna_handlers TO luna;'
```

Разрешите пользователю авторизацию в базе данных.

```
  runuser -u postgres -- psql -c 'ALTER ROLE luna WITH LOGIN;'
```

7.16.1.5 Создание базы данных Backport 3

Предполагается, что пользователь базы данных уже создан.

Создайте базу данных для сервиса Backport 3.

```
  runuser -u postgres -- psql -c 'CREATE DATABASE luna_backport3;'
```

Присвойте привилегии пользователю базы данных.

```
runuser -u postgres -- psql -c 'GRANT ALL PRIVILEGES ON DATABASE
   luna_backport3 TO luna;'
```

Разрешите пользователю авторизацию в базе данных.

```
runuser -u postgres -- psql -c 'ALTER ROLE luna WITH LOGIN;'
```

7.16.1.6 Создание базы данных Faces

Предполагается, что пользователь базы данных уже создан.

Создайте базу данных для сервиса Faces.

```
runuser -u postgres -- psql -c 'CREATE DATABASE luna_faces;'
```

Присвойте привилегии пользователю базы данных.

```
runuser -u postgres -- psql -c 'GRANT ALL PRIVILEGES ON DATABASE luna_faces
   TO luna;'
```

Разрешите пользователю авторизацию в базе данных.

```
runuser -u postgres -- psql -c 'ALTER ROLE luna WITH LOGIN;'
```

7.16.1.7 Создание базы данных Events

Предполагается, что пользователь базы данных уже создан.

Создайте базу данных для сервиса Events.

```
runuser -u postgres -- psql -c 'CREATE DATABASE luna_events;'
```

Присвойте привилегии пользователю базы данных.

```
runuser -u postgres -- psql -c 'GRANT ALL PRIVILEGES ON DATABASE luna_events
    TO luna;'
```

Разрешите пользователю авторизацию в базе данных.

```
runuser -u postgres -- psql -c 'ALTER ROLE luna WITH LOGIN;'
```

7.16.1.8 Создание базы данных Tasks

Предполагается, что пользователь базы данных уже создан.

Создайте базу данных для сервиса Tasks.

```
runuser -u postgres -- psql -c 'CREATE DATABASE luna_tasks;'
```

Присвойте привилегии пользователю базы данных.

```
runuser -u postgres -- psql -c 'GRANT ALL PRIVILEGES ON DATABASE luna_tasks
    TO luna;'
```

Разрешите пользователю авторизацию в базе данных.

```
runuser -u postgres -- psql -c 'ALTER ROLE luna WITH LOGIN;'
```

7.16.1.9 Создание базы данных Lambda

Предполагается, что пользователь базы данных уже создан.

Создайте базу данных для сервиса Lambda.

```
runuser -u postgres -- psql -c 'CREATE DATABASE luna_lambda;'
```

Присвойте привилегии пользователю базы данных.

```
runuser -u postgres -- psql -c 'GRANT ALL PRIVILEGES ON DATABASE luna_lambda
    TO luna;'
```

Разрешите пользователю авторизацию в базе данных.

```
runuser -u postgres -- psql -c 'ALTER ROLE luna WITH LOGIN;'
```

7.16.1.10 Создание базы данных Video Manager

Предполагается, что пользователь базы данных уже создан.

Создайте базу данных для сервиса Video Manager.

```
runuser -u postgres -- psql -c 'CREATE DATABASE luna_video_manager;'
```

Присвойте привилегии пользователю базы данных.

```
runuser -u postgres -- psql -c 'GRANT ALL PRIVILEGES ON DATABASE
   luna_video_manager TO luna;'
```

Разрешите пользователю авторизацию в базе данных.

```
runuser -u postgres -- psql -c 'ALTER ROLE luna WITH LOGIN;'
```

7.16.2 Компиляция VLMatch в PostgreSQL

Примечание. В следующей инструкции описана установка для PostgreSQL 16.

Все файлы, требуемые для компиляции расширения, заданного пользователем (UDx), в VLMatch, можно найти в следующей директории:

```
/var/lib/luna/current/extras/VLMatch/postgres/
```

Для компиляции функции VLMatch UDx необходимо:

- установить репозиторий RPM:

```
dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-8-
   x86_64/pgdg-redhat-repo-latest.noarch.rpm
```

- установить PostgreSQL:

```
dnf install postgresql16-server
```

- установить окружение для разработки:

```
dnf install postgresql16-devel
```

- установить пакет gcc:

```
dnf install gcc-c++
```

- установить CMAKE. Необходима версия 3.5 или выше.
- открыть скрипт make.sh в текстовом редакторе. Он включает в себя пути к используемой на данный момент версии PostgreSQL. Измените следующие значения (при необходимости):

  `SDK_HOME` задает путь к домашней директории PostgreSQL. По умолчанию это `/usr/pgsql-16/include/server`;

  `LIB_ROOT` задает путь к библиотечной корневой директории PostgreSQL. По умолчанию это `/usr/pgsql-16/lib`.

  Откройте директорию скрипта make.sh и запустите его:

```
cd /var/lib/luna/current/extras/VLMatch/postgres/
```

```
chmod +x make.sh
```

```
./make.sh
```

Перенесите сгенерированный файл `VLMatchSource.so` в любое удобное место если PostgreSQL работает вне контейнера или в директорию `/srv` в контейнер PostgreSQL.

Путь до библиотеки указывается во время создания функции в БД (см. ниже).

7.16.3 Добавление функции VLMatch для БД Faces в PostgreSQL

Сервис Faces требует добавления дополнительной функции VLMatch к используемой базе данных. LUNA PLATFORM не может выполнять вычисления по сравнению биометрических шаблонов без этой функции.

Библиотека VLMatch компилируется для конкретной версии базы данных.

> Не используйте библиотеку, созданную для другой версии базы данных. Например, библиотеку, созданную для PostgreSQL версии 12 нельзя использовать для PostgreSQL версии 16.

В данном разделе описывается создание функции для PostgreSQL. Библиотека VLMatch должна быть скомпилирована и перенесена в PostgreSQL. См. раздел «Компиляция VLMatch для PostgreSQL».

7.16.3.1 Добавление функции VLMatch в базу данных Faces

Функцию VLMatch необходимо применить в базу данных PostgreSQL.

Определите функцию в базе данных Faces:

```
sudo -u postgres -h 127.0.0.1 -- psql -d luna_faces -c "CREATE FUNCTION
    VLMatch(bytea, bytea, int) RETURNS float8 AS '/srv/VLMatchSource.so', '
    VLMatch' LANGUAGE C PARALLEL SAFE;"
```

Важно! Здесь `/srv/VLMatchSource.so` - полный путь до скомпилированной библиотеки. Необходимо заменить путь на актуальный.

Протестируйте функцию, отправив следующий запрос в базу данных сервиса:

```
sudo -u postgres -h 127.0.0.1 -- psql -d luna_faces -c "SELECT VLMatch('\
    x12345678901234567890123456789012345678901234567890123456789012345678901234'::bytea
    , '\x01234567890123456789012345678901234567890123456789012345678901234567890123'::
    bytea, 32);"
```

База данных должна вернуть результат «0.4765625».

7.16.4 Добавление функции VLMatch для БД Events в PostgreSQL

Сервис Events требует добавления дополнительной функции VLMatch к используемой базе данных. LUNA PLATFORM не может выполнять вычисления по сравнению биометрических шаблонов без этой функции.

Библиотека VLMatch компилируется для конкретной версии базы данных.

> Не используйте библиотеку, созданную для другой версии базы данных. Например, библиотеку, созданную для PostgreSQL версии 12 нельзя использовать для PostgreSQL версии 16.

В данном разделе описывается создание функции для PostgreSQL. Библиотека VLMatch должна быть скомпилирована и перенесена в PostgreSQL. См. раздел «Компиляция VLMatch для PostgreSQL».

7.16.4.1 Добавление функции VLMatch в базу данных Events

Функцию VLMatch необходимо применить в базе данных PostgreSQL.

Определите функцию в базе данных Events.

```
sudo -u postgres -h 127.0.0.1 -- psql -d luna_events -c "CREATE FUNCTION
    VLMatch(bytea, bytea, int) RETURNS float8 AS 'VLMatchSource.so', 'VLMatch
    ' LANGUAGE C PARALLEL SAFE;"
```

Протестируйте функцию.

```
sudo -u postgres -h 127.0.0.1 -- psql -d luna_events -c "SELECT VLMatch('\
    x12345678901234567890123456789012345678901234567890123456789012345678901234'::bytea
    , '\x01234567890123456789012345678901234567890123456789012345678901234567890123'::
    bytea, 32);"
```

База данных должна вернуть результат «0.4765625».

7.16.5 VLMatch для Oracle

Примечание. В следующей инструкции описана установка для Oracle 21c.

Все файлы, требуемые для компиляции расширения, заданного пользователем (UDx), в VLMatch, можно найти в следующей директории:

```
/var/lib/luna/current/extras/VLMatch/oracle
```

Для компиляции функции VLMatch UDx необходимо:

- Установить требуемое окружение, см. требования:

```
sudo yum install gcc g++
```

2. Поменяйте переменную SDK_HOME — oracle sdk root (по умолчанию $ORACLE_HOME/bin, проверьте, что переменная окружения $ORACLE_HOME задана) в makefile.

```
vi /var/lib/luna/current/extras/VLMatch/oracle/make.sh
```

3. Откройте директорию и запустите файл «make.sh».

```
cd /var/lib/luna/current/extras/VLMatch/oracle
```

```
chmod +x make.sh
```

```
./make.sh
```

4. Определите библиотеку и функцию внутри базы данных (из консоли базы данных):

```
CREATE OR REPLACE LIBRARY VLMatchSource AS '$ORACLE_HOME/bin/VLMatchSource.
   so';
CREATE OR REPLACE FUNCTION VLMatch(descriptorFst IN RAW, descriptorSnd IN
   RAW, length IN BINARY_INTEGER)
   RETURN BINARY_FLOAT
AS
   LANGUAGE C
   LIBRARY VLMatchSource
   NAME "VLMatch"
   PARAMETERS (descriptorFst BY REFERENCE, descriptorSnd BY REFERENCE,
      length UNSIGNED SHORT, RETURN FLOAT);
```

5. Протестируйте функцию посредством вызова (из консоли базы данных):

```
SELECT VLMatch(HEXTORAW('
   12345678901234567890123456789012345678901234567890123456789012345678901234'),
   HEXTORAW('
```

```
        01234567890123456789012345678901234567890123456789012345678901234567890123'), 32)
        FROM DUAL;
```

Результат должен быть равен «0.4765625».

Перенесите сгенерированный файл `VLMatchSource.so` в любое удобное место если Oracle работает вне контейнера или в директорию `/srv` в контейнер Oracle.

Путь до библиотеки указывается во время создания функции в БД (см. ниже).

7.17 Сбор информации для технической поддержки

Для эффективного и оперативного решения проблем, технической поддержке VisionLabs необходимо предоставить логи сервисов LUNA PLATFORM и дополнительную информацию о статусе сторонних сервисов, состоянии лицензии, настройках LUNA PLATFORM и пр.

Соберите нижеописанные данные и отправьте их специалистам VisionLabs.

7.17.1 Сбор логов сервисов

В LUNA PLATFORM существует два способа вывода логов:

- стандартный вывод логов (stdout);
- вывод логов в файл.

Настройки вывода логов задаются в настройках каждого сервиса в секции `<SERVICE_NAME>_LOGGER`.

По умолчанию логи выводятся только в стандартный вывод.

> Для более подробной информации о системе логирования LUNA PLATFORM см. раздел «Логирование информации» в руководстве администратора.

Соберите логи всех сервисов. Например, можно собрать логи за последние 10 минут для всех сервисов с помощью нижеописанных команд.

```
docker logs --since 10m luna-licenses > luna-licenses_log.txt
docker logs --since 10m luna-faces > luna-faces_log.txt
docker logs --since 10m luna-image-store > luna-image-store_log.txt
docker logs --since 10m luna-accounts > luna-accounts_log.txt
docker logs --since 10m luna-tasks > luna-tasks_log.txt
docker logs --since 10m luna-events > luna-events_log.txt
docker logs --since 10m luna-sender > luna-sender_log.txt
docker logs --since 10m luna-admin > luna-admin_log.txt
docker logs --since 10m luna-remote-sdk > luna-remote-sdk_log.txt
docker logs --since 10m luna-handlers > luna-handlers_log.txt
docker logs --since 10m luna-lambda > luna-lambda_log.txt
docker logs --since 10m luna-python-matcher > luna-python-matcher_log.txt
docker logs --since 10m luna-backport3 > luna-backport3_log.txt
docker logs --since 10m luna-backport4 > luna-backport4_log.txt
```

7.17.2 Сбор дополнительной информации

Необходимо собрать следующую дополнительную информацию:

- Версия LUNA PLATFORM.

  Версию LUNA PLATFORM содержится в названии архива с комплектом поставки. Также можно узнать актуальную версию перейдя на страницу `http://your_server_ip_adress:5000/version` в браузере.

- Статус лицензии в зависимости от вендора:

  - HASP — информация со страниц `http://your_server_ip_adress:1947/_int_/features.html` и `http://your_server_ip_adress:1947/_int/devices.html`
  - Guardant — информация со страниц `http://your_server_ip_adress:3189/#/dongles/list` и `http://your_server_ip_adress:3189/#/sessions`

- Актуальные настройки LUNA PLATFORM.

  Актуальные настройки можно получить перейдя на страницу `http://your_server_ip_adress:5010/4/luna_sys_info`, указав логин и пароль от аккаунта. Логин и пароль по умолчанию — `root@visionlabs.ai/root`. После ввода пароля будет скачан файл в формате `json`, который нужно передать в техническую поддержку.

- Статус сторонних сервисов:

  - Docker: `systemctl status docker`
  - aksusbd: `systemctl status aksusbd`
  - grdcontrol: `systemctl status grdcontrol`

- Статус контейнеров LUNA PLATFORM.

  Можно получить список всех контейнеров с помощью команды `docker ps -a`.

- Список открытых портов.

  Можно вывести список открытых портов с помощью команды `ss -ltn`.

- Список реестров в конфигурации Docker, к которым можно подключаться без использования защищенного соединения.

  Список реестров можно получить с помощью команды `cat /etc/docker/*`.

- Правила файрволла.

  Правила файрволла можно получить с помощью команды `iptables-save`.

- Общая информация о системе:

  - Информация о процессоре: `lscpu`
  - Использование памяти: `free -h; lsmem`
  - Использование дискового пространства: `df -h`

- Окружение и тип сервера:

Укажите, в каком окружении работает система (тестовое, продуктивное) и является ли сервер виртуальным или физическим.

8 Рекомендации

В данном разделе приводятся рекомендации для оптимальной работы с LUNA PLATFORM.

8.1 Оптимизация ресурсов

В данном разделе представлены советы по оптимизации ресурсов системы при работе с LUNA PLATFORM.

1. Убирать ненужные политики в обработчиках, включенные по умолчанию. Например, сохранение биометрических образцов и событий включено в обработчиках по умолчанию.

   Если сохранение не требуется, то необходимо их отключить. В противном случае необходимо следить и периодически удалять устаревшие данные, чтобы память сервера не заполнилась.

   Если включены эстимации, которые не требуются, то их выполнение будет увеличивать время выполнения каждого запроса.

2. Отключить использование ненужных сервисов.

3. Вместо увеличения количества экземпляров сервисов, увеличивать количество «рабочих процессов» (за исключением сервиса Remote SDK на GPU и Python Matcher).

4. Регулировать количество экземпляров/«рабочих процессов» Remote SDK и параметр «num_threads» (количество потоков «рабочего процесса») в настройках сервиса Remote SDK при работе на CPU.

   Как правило, значение параметра «num_threads» и количество экземпляров/«рабочих процессов» Remote SDK кратно числу физических ядер. Т.е. если есть 8 физических ядер, то рекомендуется использовать 2 экземпляра Remote SDK и «num_threads» = 4 для каждого экземпляра (2x4=8). Аналогично, если 24 ядра, то нужно 4 экземпляра Remote SDK и «num_threads» = 6 для каждого экземпляра (4x6=24).

   Следует помнить о том, что слишком большое количество экземпляров/«рабочих процессов» может негативно сказаться на производительности, поэтому не следует запускать 8 экземпляров/воркеров с «num_threads» = 1 при 8 доступных ядрах. Можно разделить количество ядер на 6 или 8 для определения количества экземпляров/«рабочих процессов».

   Параметр «num_threads» не влияет на работу на GPU.

5. При выполнении сравнения явно указывать необходимые поля «target».

   Например, если от результата сравнения не требуется ничего кроме «face_id», то нет смысла тратить ресурсы на использование остальных полей (поведение по умолчанию если не задать поля в «target»).

В рамках обработчиков в политике «match_policy» также указываются «targets», которые тоже необходимо настроить.

6. При небольших списках можно запускать больше экземпляров Python Matcher, уменьшая параметр «thread_count».
7. При больших списках более 1-2M не делать больше одного сервиса Python Matcher на одной NUMA-node.
8. Грамотно указывать лимит кандидатов и эталонов и не задавать больше чем нужно (см. настройку «PLATFORM_LIMITS» в настройках сервиса Python Matcher).

   Например, если нужно получить в ответе только одного кандидата с самой высокой схожестью, то не нужно указывать, что должно возвращаться top-3 кандидата.
9. Установить уровень логирования на «WARNING».

   Обратите внимание, что уменьшение уровня логирования может уменьшить потребление ресурсов, однако в случае возникновения проблем, данного уровня логов может быть недостаточно.
10. Использовать расписание задач для управления запуском задачи Garbage collection.
11. Использовать заголовки «Accept-Encoding» со значением «gzip, deflate» для оптимизации сетевого трафика при запросах к API при получении данных в формате JSON.
12. Уменьшать значение параметра «optimal_batch_size» в настройках сервиса Remote SDK при работе на CPU (значения разные для каждого эстиматора).

    При работе на CPU, параметр «optimal_batch_size» должен быть меньше, чем значение «num_threads». Например, если «num_threads» = 4, то рекомендуется выставлять «optimal_batch_size» <= 4.

    Как и в случае с «num_threads», оптимальное значение «optimal_batch_size» зависит от конкретной системы и характеристик задачи, поэтому его следует настраивать экспериментальным путем.
13. Не забывать указывать «image_type» равным «1» или 2”при отправке биометрических образцов лица или тела. По умолчанию значение равно «0» (необработанные изображения).

    Если оставить значение «0» , то выполняется повторное детектирование лица на изображении, что увеличивает время обработки запроса и может приводить к непредвиденным проблемам. Например, когда во frontend нашлось одно лицо, а в backend нашлось два лица, т. к. используются разные версии детекторов или разные настройки для детекции лиц.
14. Запускать Remote SDK только с определенными эстиматорами.

    Если какие-то эстиматоры не требуются для реализации бизнес-логики, то их можно отключить и удалить из контейнера, чтобы уменьшить потребление памяти.

15. Ознакомиться со разделом «Потребление ресурсов сервисами», чтобы подбирать под сервисы оптимальные серверы.

8.2 Продвинутая настройка PostrgeSQL

PostgreSQL можно настроить на эффективное взаимодействие с LUNA PLATFORM 5. Для этого необходимо задать определенные значения настройкам PostgreSQL в файле `postgresql.conf`.

> В данном разделе не приводится полный список всех настроек с подробным описанием. См. полный перечень настроек с их описанием на официальном сайте PostgreSQL.

> Полезные советы для расчета конфигурации PostgreSQL описаны здесь: https://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server.

> Доступна возможность рассчитать конфигурацию для PostgreSQL исходя из максимальной производительности для заданной аппаратной конфигурации (см. https://pgtune.leopard.in.ua/).

8.2.1 Рекомендуемые значения для настроек

Примечание. Следует изменять нижеприведенные настройки с осторожностью, т.к. ручное изменение настроек PostgreSQL требует должный опыт.

Ниже приведены рекомендуемые значения настроек и их описание.

max_connections = **200** — задаёт максимальное количество одновременных подключений к серверу БД. По умолчанию равно 100.

Значения по умолчанию может хватать для тестовых демонстраций LUNA PLATFORM, однако для реальных целей, стандартного значения может быть недостаточно и его необходимо будет высчитать.

В сервисе Configurator можно задать количество подключений к БД с помощью настройки `connection_pool_size`, расположенной в секциях `LUNA_<SERVICE_NAME>_DB`, где `<SERVICE_NAME>` — имя сервиса, имеющего базу данных. Реальное количество подключений может быть больше значения данной настройки на 1.

Если соединений слишком много, но мало активных, можно использовать сторонние сервисы для балансировки нагрузки, например, `haproxy` или `pgbouncer`. При использовании сервисов балансировки необходимо учитывать некоторые нюансы, описанные здесь: https://magicstack.github.io/asyncpg/current/faq.html#why-am-i-getting-prepared-statement-errors.

maintenance_work_mem = **2GB** — задаёт максимальный объём памяти для операций обслуживания БД.

shared_buffers = **0.25...0.5 * RAM (MB)** — определяет, сколько памяти будет выделяться Postgres для кеширования данных. Зависит от того, как часто выполняется сравнение по БД, какие индексы и т.п.

effective_io_concurrency = **100** — задаёт допустимое число параллельных операций ввода/вывода, которое говорит Postgres о том, сколько операций ввода/вывода могут быть выполнены одновременно. Чем больше это число, тем больше операций ввода/вывода будет пытаться выполнить параллельно PostgreSQL в отдельном сеансе.

max_worker_processes = **CPU_COUNT (количество ядер процессора)** — задаёт максимальное число фоновых процессов, которые можно запустить в текущей системе.

max_parallel_maintenance_workers = **4** — задаёт максимальное число параллельных рабочих процессов, выполняющих команду создания индекса (`CREATE INDEX`).

max_parallel_workers_per_gather = **4** — задаёт максимальное число рабочих процессов, на которые может быть распараллелен запрос или подзапрос.

max_parallel_workers = **CPU_COUNT (количество ядер процессора)** — задаёт максимальное число рабочих процессов, которое система сможет поддерживать для параллельных операций.

Примечание. Следующие значения настроек относятся к функции сравнения по базе данных для больших таблиц.

enable_bitmapscan = **off** — включает или отключает использование bitmapscan. Иногда может понадобится, когда PostgreSQL по ошибке определяет, что bitmapscan лучше индекса. Изменять рекомендуется только при необходимости, когда предполагается, что запрос будет использовать индекс, но по неизвестным причинам не использует.

seq_page_cost = **1** — задаёт приблизительную стоимость чтения одной страницы с диска, которое выполняется в серии последовательных чтений.

random_page_cost = **1.5** — задаёт приблизительную стоимость чтения одной произвольной страницы с диска.

parallel_tuple_cost = **0.1** — задаёт приблизительную стоимость передачи одного кортежа (строки) от параллельного рабочего процесса другому процессу.

parallel_setup_cost = **5000.0** — задаёт приблизительную стоимость запуска параллельных рабочих процессов.

max_parallel_workers_per_gather = **CPU_COUNT (количество ядер процессора) / 2** — задаёт максимальное число рабочих процессов, на которые может быть распараллелен запрос или подзапрос.

min_parallel_table_scan_size = **1MB** — задаёт минимальный объем данных таблицы, подлежащий сканированию, при котором может применяться параллельное сканирование.

min_parallel_index_scan_size = **8k** — задаёт минимальный объем индексных данных для параллельного сканирования.

effective_cache_size = **75% от RAM** — определяет ожидаемый объем оперативной памяти, который может быть использован для кэширования данных в базе данных.

9 Диаграммы последовательностей

В этой главе описаны общие запросы к LP и показано взаимодействие между сервисами при обработке запроса.

> Более подробную информацию о запросах можно найти в справочном руководстве сервиса API.

9.1 Диаграммы создания биометрических образцов

9.1.1 Диаграмма создания биометрического образца

С помощью этого запроса можно определять лица на исходных фотографиях. Фотографии можно отправлять в стандартных форматах изображений (JPEG, PNG и т.д.) или в формате BASE64.

Более подробная информация приведена в запросе «detect faces» в справочном руководстве сервиса API.

Запрос	Описание	Тип
detect faces	Обнаруживать лица на входных изображениях, оценивать свойства лиц, извлекать биометрические образцы и сохранять их в Image Store.	POST

Результат запроса содержит параметры обнаруженных лиц и идентификаторы всех созданных биометрических образцов.

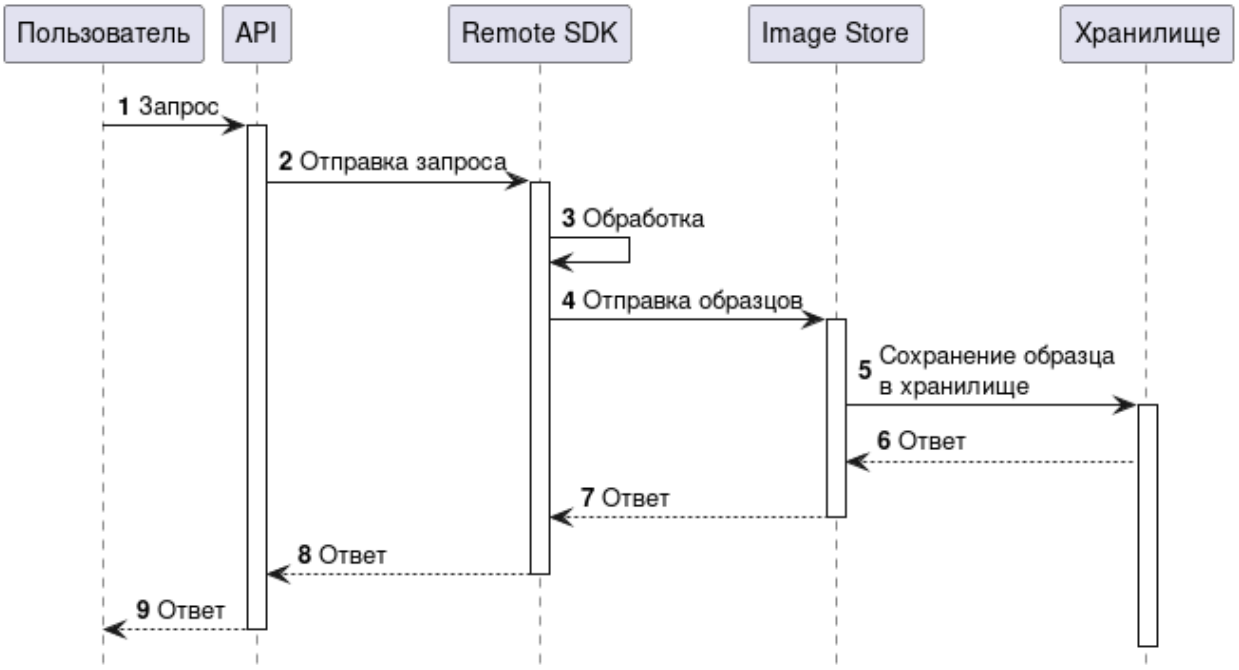


Рис. 64: Диаграмма создания биометрического образца

Общая схема обработки запроса:

1. Запрос на обнаружение лиц отправляется в API.
2. API получает запрос, обрабатывает его и отправляет задачу в сервис Remote SDK.
3. Сервис Remote SDK обрабатывает задачу в соответствии с заданными параметрами.
4. Сервис Remote SDK отправляет полученные биометрические образцы в Image Store.
5. С помощью Image Store биометрические образцы сохраняются в хранилище.
6. Из хранилища выдается результат сохранения биометрического образца.
7. Image Store выдает идентификаторы биометрических образцов и URL-адреса.
8. Сервис Remote SDK отправляет идентификаторы биометрических образцов и полученные параметры лиц в сервис API.
9. API генерирует и отправляет ответ.

9.1.2 Получение информации о биометрических образцах и их сохранение

С помощью запроса, приведенного ниже, внешние биометрические образцы добавляются в Image Store.

Более подробная информация приведена в запросе «save samples» в справочном руководстве сервиса API.

Запрос	Описание	Тип
save samples	Сохранить внешний биометрический образец в Image Store	POST

С помощью следующих запросов можно управлять уже существующими биометрическими образцами.

Более подробная информация приведена в разделе «samples» в справочном руководстве сервиса API.

Запрос	Описание	Тип
get sample	Получить биометрический образец по его sample_id	GET
remove sample	Удалить биометрический образец по его sample_id	DELETE
check to exist sample	Проверить наличие биометрического образца по его sample_id	HEAD

Все вышеперечисленные запросы обрабатываются с помощью сервиса Image Store.

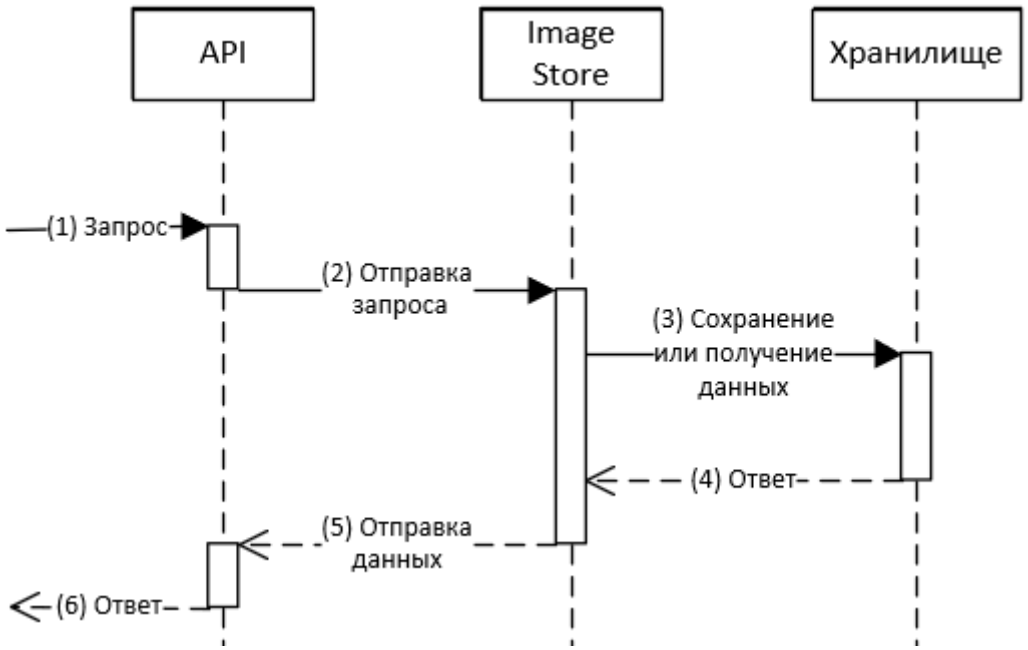


Рис. 65: Диаграмма получения информации о биометрических образцах и их сохранения

Схема обработки запросов:

1. Запрос отправляется в сервис API.

2. API отправляет запрос в Image Store.
3. Image Store выполняет необходимые действия с хранилищем.
4. Из хранилища выдаются необходимые данные.
5. Сервис Image Store выдает результат в сервис API.
6. API генерирует и отправляет ответ.

> Более подробную информацию о запросах можно найти в документе «APIReferenceManual.html» в разделе «Samples».

9.2 Диаграммы атрибутов

9.2.1 Диаграмма извлечения временных атрибутов

Сервис Handlers получает биометрические образцы из Image Store и извлекает из них информацию.

Более подробная информация приведена в запросе«extract attributes» в справочном руководстве сервиса API.

Необходимо указать массив идентификаторов биометрических образцов.

Запрос	Описание	Тип
extract attributes	Извлечь атрибуты: БШ и базовые атрибуты (возраст, пол, этническая принадлежность)	POST

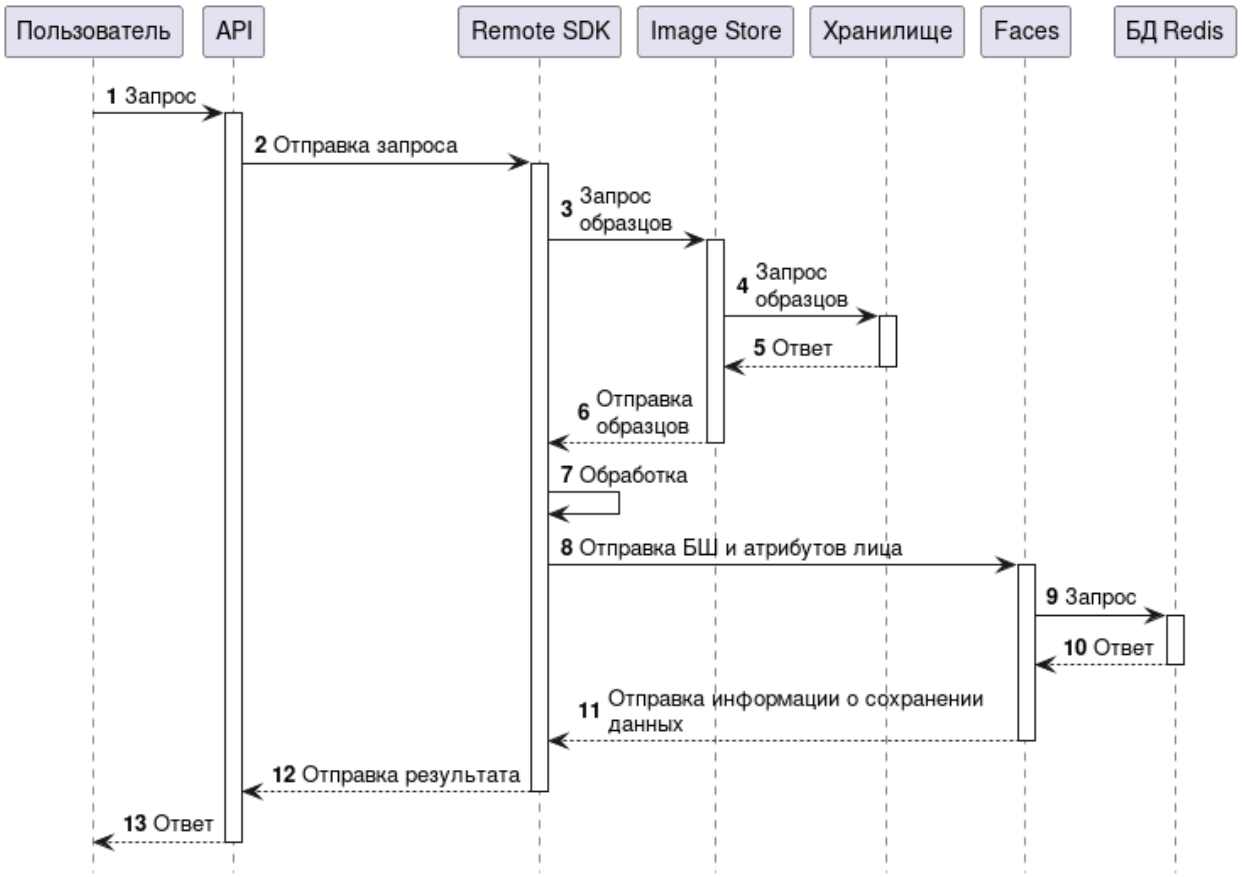


Рис. 66: Диаграмма извлечения временных атрибутов

Общая схема обработки запроса:

1. Запрос на извлечение отправляется в API.
2. Сервис API получает запрос, обрабатывает его и отправляет запрос в сервис Remote SDK.
3. Сервис Remote SDK запрашивает биометрические образцы из Image Store.
4. Сервис Image Store запрашивает биометрические образцы из хранилища.
5. Из хранилища отправляются биометрические образцы.
6. Сервис Image Store отправляет биометрические образцы в сервис Remote SDK.
7. Сервис Remote SDK обрабатывает задачу в соответствии с заданными параметрами.
8. Сервис Remote SDK отправляет БШ и базовые атрибуты в сервис Faces.
9. Сервис Faces отправляет запросы на хранение временных атрибутов в базе данных Redis.
10. Из базы данных Redis отправляется ответ в Faces.
11. Сервис Faces отправляет ответ в сервис Remote SDK.

12. Сервис Remote SDK отправляет полученные идентификаторы атрибутов, базовые атрибуты, URL-адреса атрибутов, результаты фильтрации и оценку по шкале GS в сервис API.
13. Сервис API отправляет ответ.

9.2.2 Диаграмма создания атрибута по внешним данным

На диаграмме показано создание атрибутов с использованием данных из внешней базы данных.

Более подробная информация приведена в разделе «attributes» в справочном руководстве сервиса API.

Запрос	Описание	Тип
create temporary attribute	Создать новые временные атрибуты.	POST

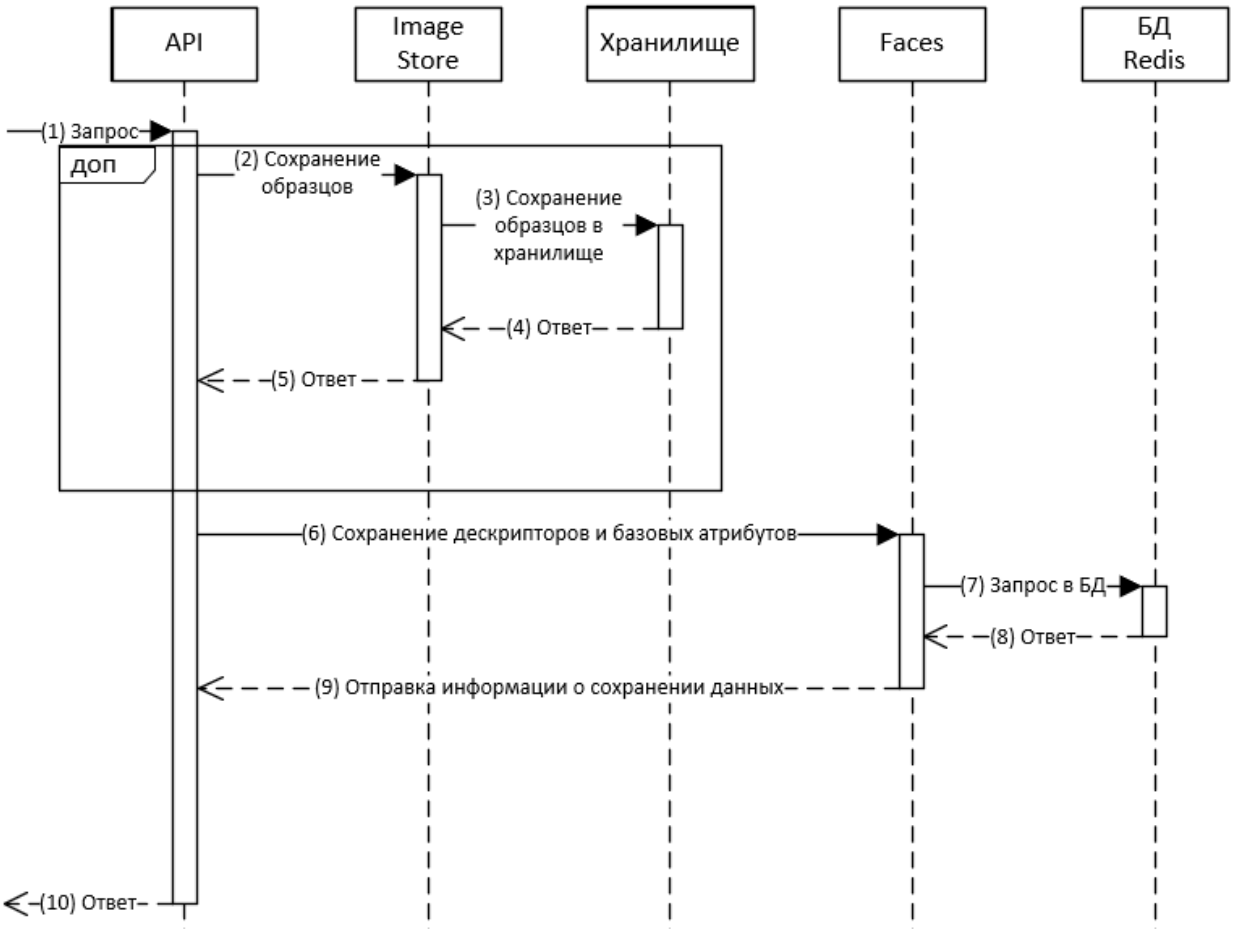


Рис. 67: Диаграмма сохранения внешних временных атрибутов

1. Запрос на добавление новых атрибутов (биометрических шаблонов и/или базовых атрибу-

тов) отправляется в сервис API. Все необходимые данные для создания атрибутов отправляются вместе с запросом. **Дополнительно.** В сервис Image Store отправляется запрос, если биометрические образцы предоставляются с БШ и/или атрибутами;

2. Сервис API отправляет запрос в сервис Image Store для сохранения полученных биометрических образцов.
3. Сервис Image Store запрашивает биометрические образцы из хранилища.
4. Из хранилища отправляются биометрические образцы.
5. Сервис Image Store отправляет биометрические образцы в сервис API.
6. Сервис API отправляет БШ и базовые атрибуты в сервис Faces.
7. Сервис Faces отправляет запросы на хранение временных атрибутов в базе данных Redis.
8. Из базы данных Redis отправляется ответ в Faces.
9. Сервис Faces отправляет ответ в сервис API.
10. Сервис API отправляет ответ.

9.2.3 Диаграммы получения информации об атрибутах

С помощью следующих запросов можно получить данные об уже существующих атрибутах или удалить их.

Запрос	Описание	Тип
get temporary attributes	Получить все идентификаторы атрибутов и время их создания в соответствии с целевыми полями	GET
get temporary attribute	Получить информацию о временном атрибуте по его attribute_id	GET
check temporary attribute	Проверить наличие атрибута по его attribute_id	HEAD
delete attributes	Удалить атрибут по его attribute_id	DELETE
get temporary attribute samples	Получить все биометрические образцы временных атрибутов по attribute_id	GET

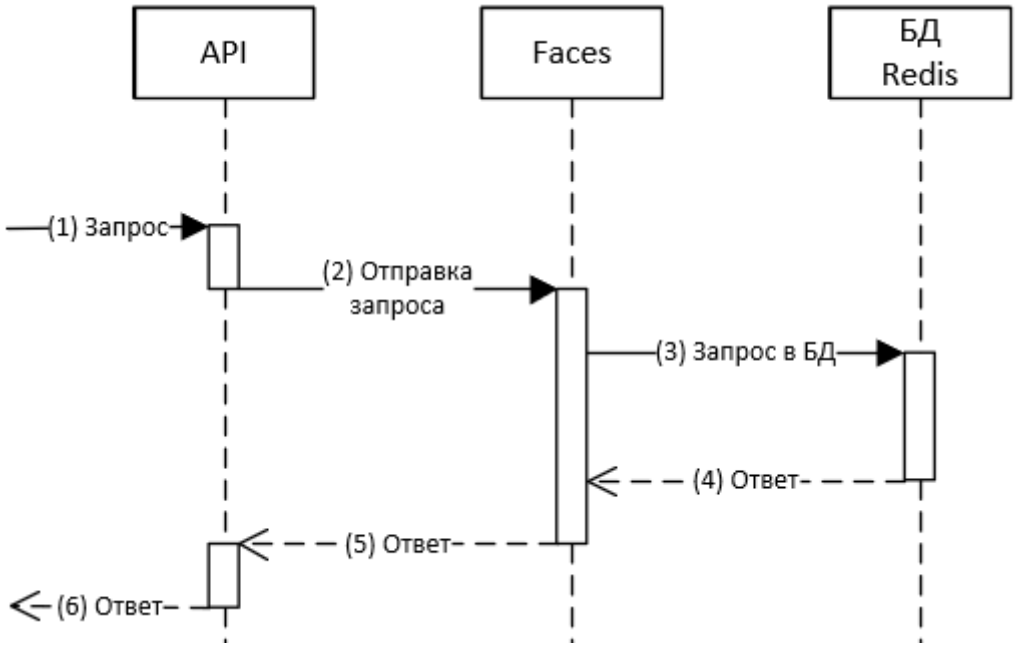


Рис. 68: Диаграмма получения информации о временных атрибутах

Общая схема обработки запроса:

1. Запрос отправляется в сервис API.
2. API отправляет запрос в сервис Faces.
3. Сервис Faces отправляет запрос в базу данных Redis на получение информации о временных атрибутах.
4. Из базы данных Redis выдается запрошенная информация.
5. Сервис Faces выдает результат.
6. Сервис API выдает информацию пользователю. Если TTL атрибута истек, выдается ошибка.

9.3 Диаграммы лиц и списков

Все запросы в этом разделе обрабатываются с помощью сервиса Faces.

Более подробная информация приведена в разделе «faces» в справочном руководстве сервиса API.

9.3.1 Диаграмма создания лица

Запрос	Описание	Тип
create face	Создать новое лицо с указанным идентификатором атрибута, данными пользователя, аватаром и списками	POST

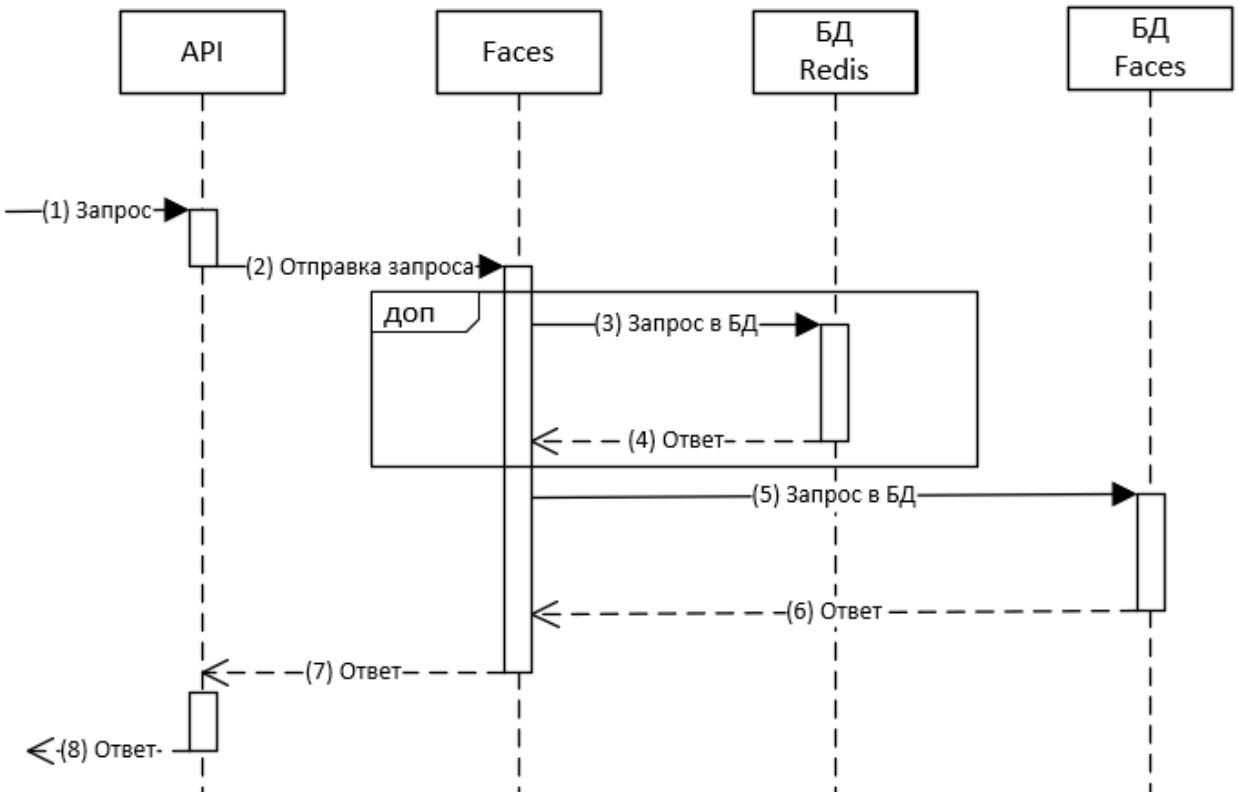


Рис. 69: Диаграммы создания нового лица

Общая схема обработки запроса:

1. Запрос отправляется в сервис API.
2. API отправляет запрос в сервис Faces.
3. Сервис Faces отправляет запрос в базу данных Redis на получение временных атрибутов; **Дополнительно.** Запрос в базу данных Redis не отправляется, если для лица указаны внешние атрибуты или не установлены атрибуты..
4. Из базы данных Redis выдается запрошенная информация.
5. Сервис Faces выдает результат.
6. Сервис Faces отправляет запрос в базу данных Faces для создания нового лица с использованием указанных данных.

7. В базе данных Faces сохраняются данные.
8. Сервис Faces выдает информацию о созданном лице.
9. Сервис API выдает информацию пользователю.

9.3.2 Информация о лицах и списках

Все последующие запросы имеют похожие диаграммы последовательности.

С помощью следующих запросов можно создать лицо, получить информацию об уже существующих лицах или удалить их.

Запрос	Описание	Тип
get faces	Получить массив всех существующих лиц и их данных: face_id, external_id, user_data, create_time, avatar, account_id и списки list_id, к которым прикреплено лицо	GET
delete faces	Удалить несколько лиц по их face_id	DELETE
get face count	Получить количество существующих лиц по заданному фильтру	GET
get count of faces with attributes	Получить количество лиц с атрибутами	GET
get face	Получить данные о лице (face_id, external_id, user_data, create_time, avatar, account_id и списки list_id, к которым прикреплено лицо) по указанному face_id	GET
patch face	Обновить лицо указанными данными: user_data, event_id, external_id, avatar	PATCH
remove face	Удалить указанное лицо	DELETE
check to exist a face	Проверить наличие лица по его face_id	HEAD
put face attribute	Установить атрибут лица, изменив все данные атрибута, соответствующие указанному лицу	PUT
get face attribute	Получить атрибуты указанного лица: пол, возраст, этническую принадлежность, время создания	GET
delete face attribute	Удалить атрибут лица по его face_id	DELETE

Запрос	Описание	Тип
get face attribute samples	Получить информацию о биометрических образцах (по sample_id), соответствующих указанному лицу	GET

С помощью следующих запросов можно создать списки, получить информацию об уже существующих списках или удалить их.

Запрос	Описание	Тип
create list	Создать новый пустой список. Можно указать для него user_data	POST
get lists	Получить массив всех существующих списков со следующими данными: list_id, user_data, account_id, create_time, last_update_time	GET
delete lists	Удалить несколько списков по их list_id	DELETE
get list count	Получить количество существующих списков	GET
get list	Получить информацию (list_id, user_data, account_id, create_time, last_update_time) о списке по list_id	GET
check list existence	Проверить наличие списка с list_id	HEAD
update list	Обновить поле user_data списка	PATCH
delete list	Удалить список с указанным list_id	DELETE
attach/detach faces to the list	Обновить список, прикрепив или открепив от него указанные лица	PATCH

Ниже на диаграмме представлен процесс работы для всех перечисленных выше запросов к сервису Faces.

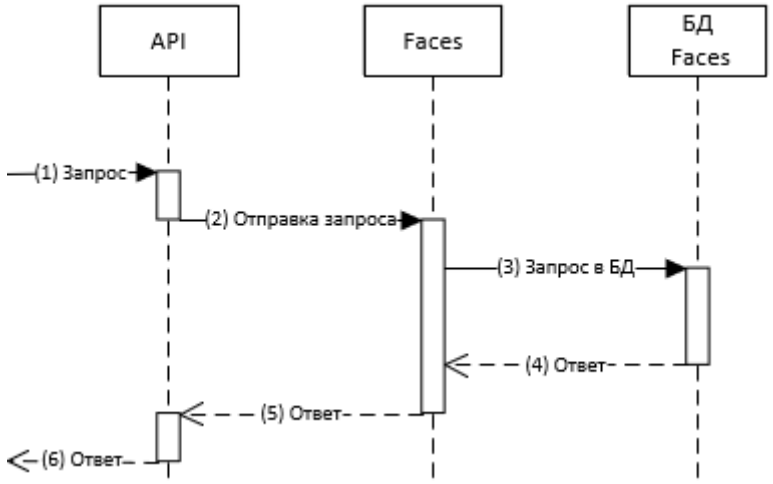


Рис. 70: Диаграммы обработки запросов к сервису Faces

Общая схема обработки запроса:

1. Запрос отправляется в сервис API.
2. API отправляет запрос в сервис Faces.
3. Сервис Faces отправляет запрос в базу данных Faces для получения информации или управления имеющимися данными.
4. Из базы данных Faces выдается запрошенная информация или информация об изменениях в базе данных.
5. Сервис Faces выдает результат.
6. Сервис API выдает результат пользователю.

9.4 Диаграммы сравнения

Необходимо указать эталоны и кандидаты для сравнения. Можно ограничить количество кандидатов с наибольшим значением схожести.

Запрос	Описание	Тип
matcher/faces	Позволяет сравнить указанные эталоны с указанными кандидатами. В результате будет получен уровень схожести для каждого кандидата и дополнительная информация о кандидатах	POST
human body matching	Задачи могут отправляться в сервис, который путем сравнения ищет тела, похожие на заданные эталоны	POST

Запрос	Описание	Тип
raw matching	Можно производить расчеты схожести для входных БШ	POST

Подробная информация приведена в разделах «matching faces», «human body matching» и «raw matching» в «APIReferenceManual.html».

9.4.1 Сравнение с помощью Python Matcher

9.4.1.1 Сравнение по базе данных

Ниже приведен пример сравнения событий (эталонов) с лицами (кандидатами).

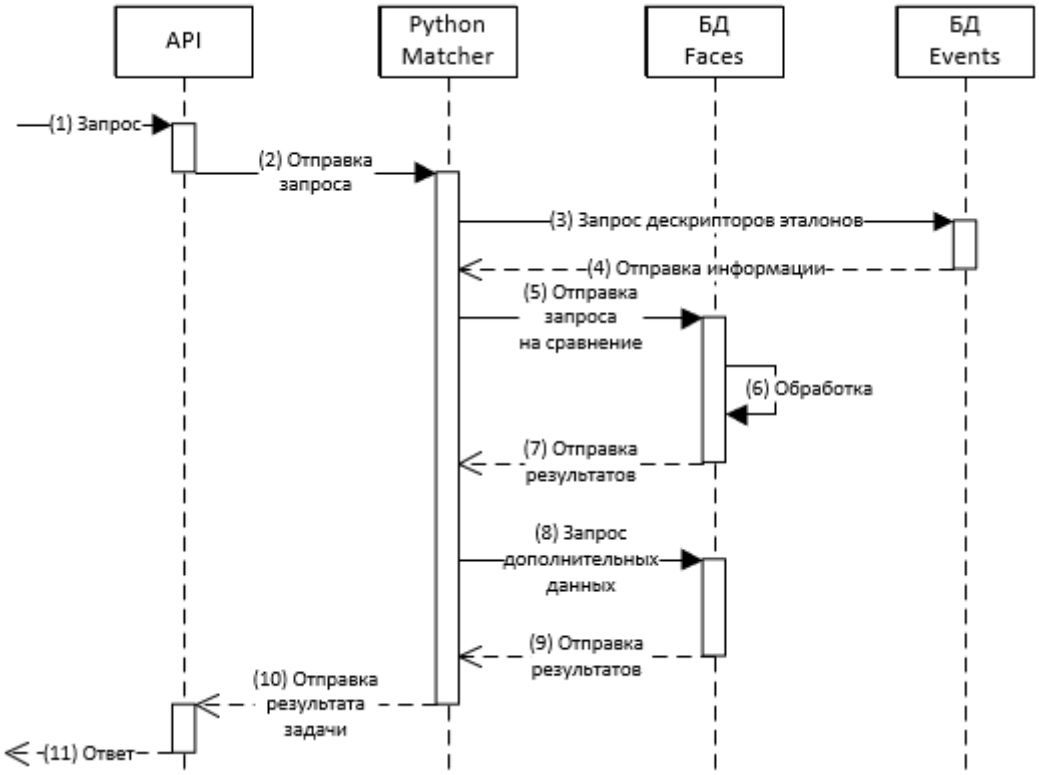


Рис. 71: Диаграмма сравнения событий и лиц

1. Запрос на сравнение отправляется в сервис API.
2. Сервис API отправляет запрос в сервис Python Matcher.
3. Сервис Python Matcher запрашивает эталоны из базы данных Events.
4. Из базы данных Events выдаются данные.

5. Сервис Python Matcher отправляет запросы на сравнение базы данных Faces.
6. Сравнение выполнено.
7. Из базы данных Faces выдаются соответствующие результаты.
8. Сервис Python Matcher запрашивает дополнительные данные для кандидатов.
9. Из базы данных Faces выдаются данные.
10. Сервис Python Matcher выдает результаты в сервис API.
11. Сервис API отправляет ответ.

9.4.1.2 Сравнение по списку

Ниже приведен пример сравнения лиц (эталонов) со списком лиц (кандидатов).

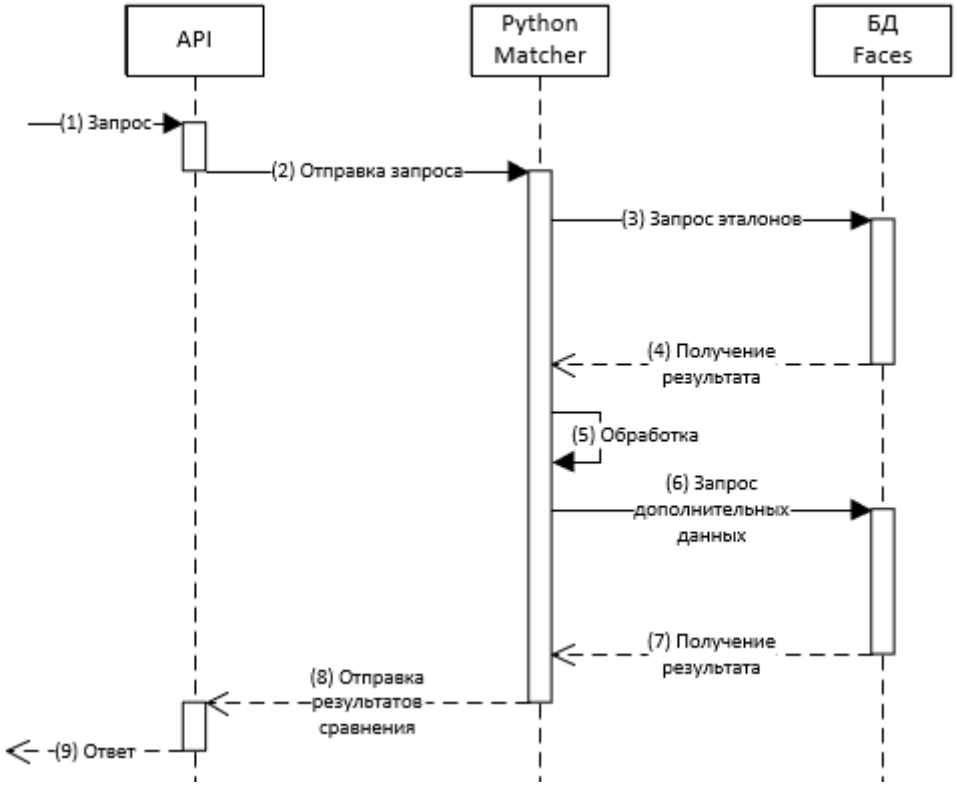


Рис. 72: Диаграмма сравнения лиц по списку

1. Запрос на сравнение отправляется в сервис API.
2. Сервис API отправляет запрос в сервис Python Matcher.
3. Сервис Python Matcher запрашивает эталоны из базы данных Faces.
4. Из базы данных Faces выдаются данные.

5. Сравнение выполняется в сервисе Python Matcher. Используются кешированные БШ.
6. Сервис Python Matcher запрашивает дополнительные данные для кандидатов.
7. Из базы данных Faces выдаются данные.
8. Сервис Python Matcher выдает результаты в сервис API.
9. Сервис API отправляет ответ.

9.5 Диаграммы обработчиков

9.5.1 Запросы на управление обработчиками

Обработчик определяет логику обработки входного изображения. Необходимо указать обработчик при создании нового события.

Запросы, приведенные ниже, относятся к обработчику.

Запрос	Описание	Тип
create handler	Создать обработчик	POST
get handlers	Получить обработчики по заданным фильтрам	GET
get handler count	Получить количество существующих обработчиков	GET
get handler	Получить политики обработчика по handler_id	GET
replace handler	Обновить поля обработчика. Необходимо указать handler_id. Необходимо указать более детальную информацию обо всех соответствующих политиках в теле запроса. Обновление индивидуальных параметров обработчика не допускается	PUT
check to exist a handler	Проверить наличие обработчика с указанным handler_id	HEAD
remove handler	Удалить обработчик по его handler_id	DELETE

Общая диаграмма создания обработчика представлена на рисунке ниже. Все запросы обработчика имеют похожие диаграммы последовательности.

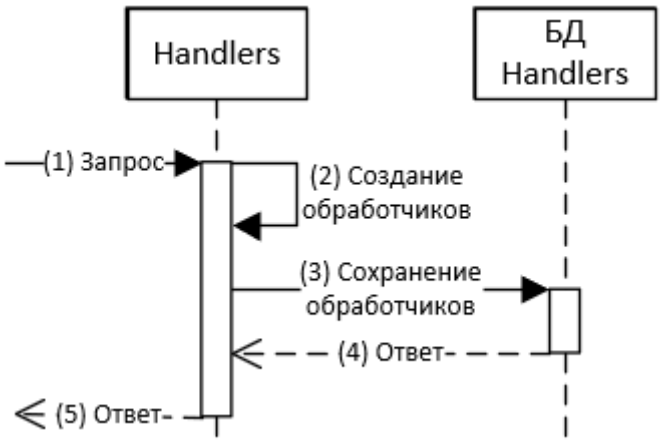


Рис. 73: Диаграмма создания обработчика

1. Запрос на создание нового обработчика отправляется из сервиса API в сервис Handlers.
2. Сервис Handlers обрабатывает запрос и создает обработчик.
3. Сервис Handlers сохраняет обработчика в базе данных API.
4. Из базы данных сервиса Handlers выдается результат.
5. Сервис Handlers выдает идентификатор созданного обработчика.

Обработчик используется при создании события. Пример его использования описан в разделе «Events». Все результаты хранятся в базе данных событий Events.

9.6 Диаграммы событий

9.6.1 Общая диаграмма создания события

Событие создается после обработки изображения в соответствии с обработчиком.

Запрос	Описание	Сервис
generate events	Создать событие. Необходимо указать соответствующий ID обработчика (handler_id) или указать политики для динамического обработчика и указать обрабатываемые изображения. Необходимо установить дополнительные параметры и данные для созданного события.	POST

Диаграмма последовательности для создания нового события будет отличаться в зависимости от указанных политик обработчика.

На приведенной ниже диаграмме последовательности показан общий процесс создания нового

события и приведены только общие точки входа для выполнения политик.

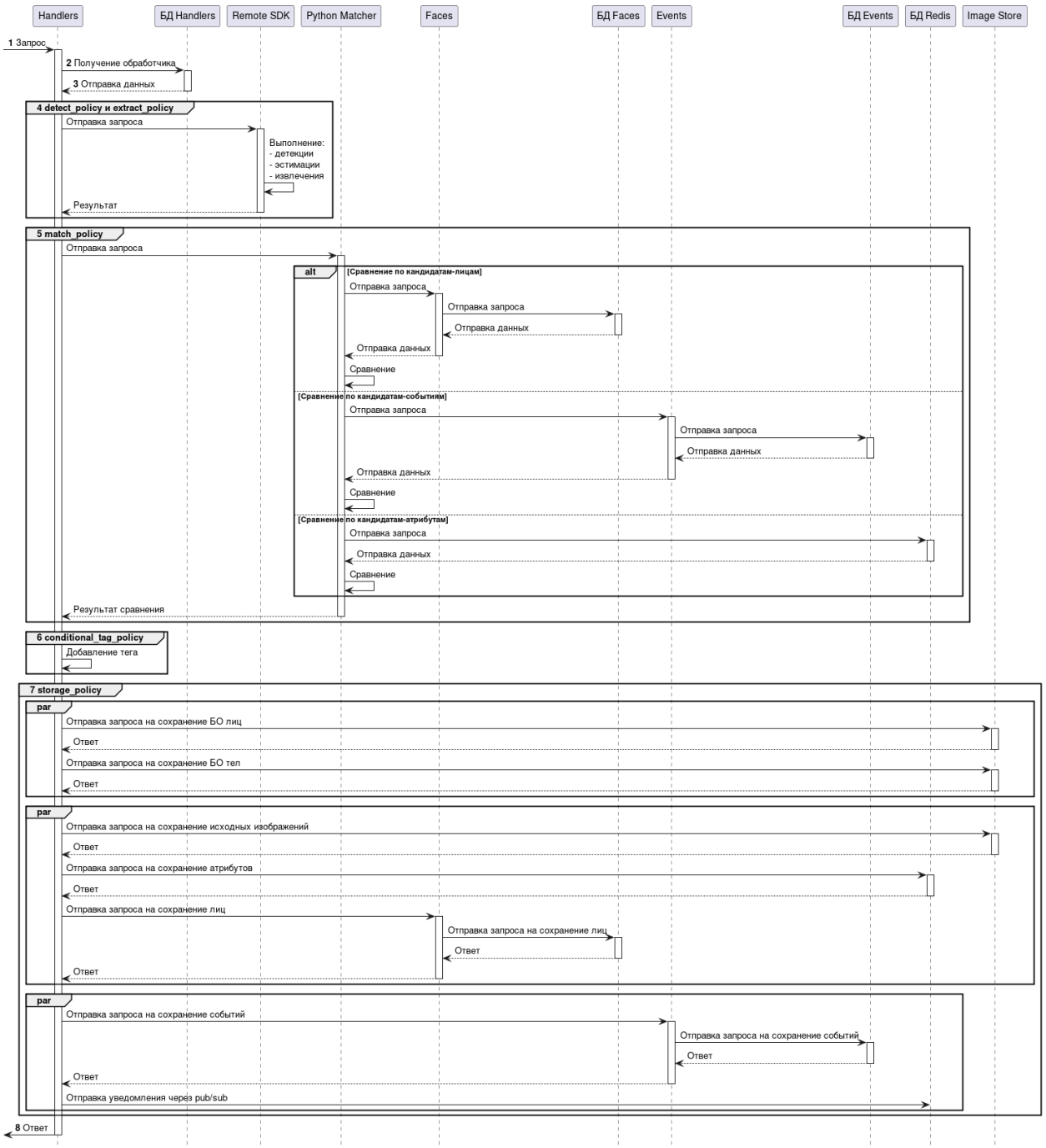


Рис. 74: Диаграмма создания события

1. Сервис API отправляет запрос на создание нового события в сервис Handlers.
2. Сервис Handlers получает соответствующий обработчик из базы данных Handlers.
3. База данных Handlers отправляет обработчик в сервис Handlers.

4. «detect_policy» и «extract_policy» обрабатываются сервисом Remote SDK. Полученные биометрические образцы и атрибуты хранятся в ОЗУ.
5. «match_policy» обрабатывается в соответствии с установленными фильтрами. Биометрические шаблоны, полученные при выполнении «extract_policy», предоставляются в виде эталонов для сравнения. Сравнение можно выполнить несколькими способами:
    - Лица установлены в качестве кандидатов. Сравнение выполняется с использованием БД Faces. В фильтрах можно указать необходимые списки с лицами.
    - События устанавливаются в качестве кандидатов. Сравнение выполняется с использованием БД Events.
    - Атрибуты устанавливаются в качестве кандидатов. Сравнение выполняется с использованием БД Redis.
6. «conditional_tags_policy» обрабатывается сервисом Handlers.
7. «storage_policy» обрабатывается в соответствии с указанными данными:
    - Биометрические образцы лиц, тел и исходные изображения хранятся в сервисе Image Store;
    - Атрибуты создаются и сохраняются в базе данных Redis сервисом Faces;
    - Лица создаются и сохраняются в базе данных Faces сервиса Faces. Их также можно привязать к спискам с помощью политики «link_to_lists_policy»;
    - События сохраняются в базе данных Events сервисом Events;
    - Уведомления отправляются через pub/sub в Redis (см. раздел «Сервис Sender»).
8. Сервис Handlers выдает результаты в сервис API.

9.6.2 Получение статистики по событиям и информации о событиях

Политика	Описание	Сервис
get statistics on events	Получить статистику по событиям. С помощью группы целевых полей (target) можно агрегировать события в соответствии с указанными агрегаторами: количество элементов, максимальное или минимальное значение, среднее значение, группировка. Также можно применять фильтры и периоды для выбора событий, которые следует агрегировать. Также возможна группировка по частоте или по временным интервалам.	POST

Политика	Описание	Сервис
get events	Получить все события, удовлетворяющие указанным фильтрам. В фильтрах вы можете установить значения одного или нескольких полей событий. С помощью параметра *target* можно выбрать поля, которые необходимо получить в ответ. Можно оставить пустым поле запроса *target*. В этом случае будут показаны все данные о событиях. Также можно установить параметры сортировки, количество событий на странице и количество страниц. Если `create_time__gte` не задано, по умолчанию устанавливается один месяц	GET
get event	Получить все поля для события. Для запроса необходимо указать *event_id*.	GET
check event existence	Проверить наличие указанного события. Для запроса необходимо указать *event_id*.	HEAD

На диаграмме последовательности показана обработка запроса.

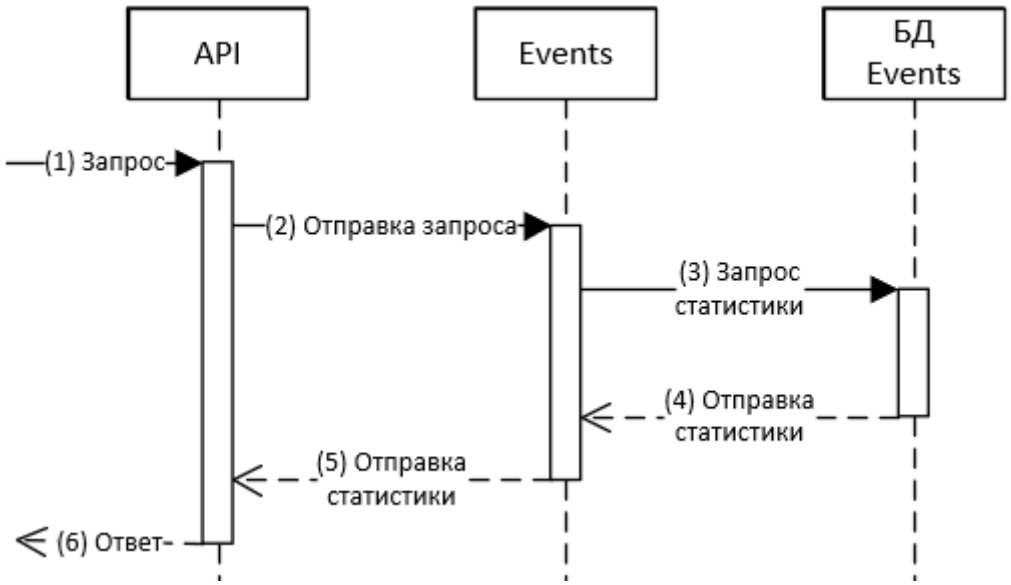


Рис. 75: Диаграмма получения информации о событиях

1. API получает запрос к сервису Events.
2. API отправляет запрос к сервису Events.

3. Сервис Events получает необходимые данные из базы данных.
4. Из базы данных выдается результат.
5. Сервис Events выдает результаты.
6. Сервис API выдает результаты.

9.7 Диаграммы задач

В данном разделе приведены диаграммы последовательности выполнения задач.

В разделе «Общая диаграмма создания и выполнения задачи» приведен общий процесс обработки задач. Для некоторых задач процесс может отличаться. Например, для задачи Clustering создается только одна подзадача и используется только один «рабочий процесс». Для таких задач приведены отдельные комментарии.

9.7.1 Общая диаграмма создания и выполнения задачи

Диаграмма общего процесса создания и выполнения задачи представлена ниже.

Обратите внимание, что «рабочий процесс» Tasks постоянно находится в ожидании получения каких-либо данных. У сервиса есть определенный приоритет выполнения работ, а именно:

1. Проверка не нужно ли отменить текущую задачу/подзадачу (не отражено в диаграмме).
2. Запросы на разделение на подзадачи.
3. Запросы на объединение результатов.
4. Запросы на обработку подзадач.

Это означает, что «рабочий процесс» Tasks не будет обрабатывать запрос на подзадачу, пока не выполнит проверку отмены текущей задачи.

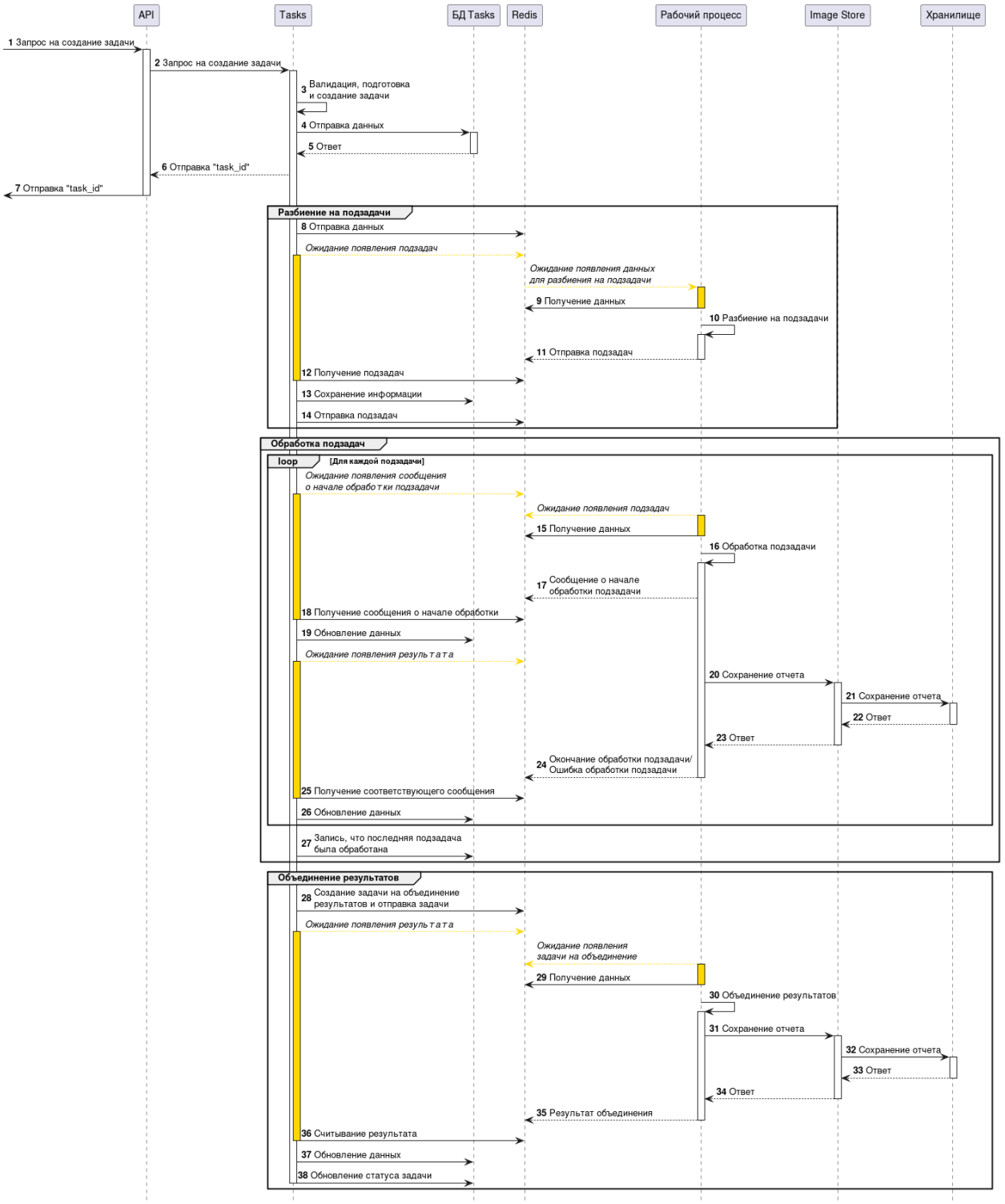


Рис. 76: Диаграмма создания задачи

9.7.1.1 Начало создания задачи

В данном разделе описан процесс начала создания задачи.

1. Сервис API получает запрос на создание задачи.
2. Сервис API отправляет запрос в сервис Tasks.
3. Сервис Tasks выполняет валидацию задачи, создает её и готовится к её обработке. На данном этапе сервис Tasks может выполнить дополнительные действия, например, получить «descriptor_id» для выполнения задачи «Additional extraction».
4. Сервис Tasks отправляет данные о создаваемой задаче в БД Tasks.
5. Сервис Tasks получает ответ.
6. Сервис Tasks отправляет идентификатор задачи «task_id» в сервис API.
7. Пользователь получает идентификатор задачи, т.е. сервис API выдает идентификатор созданной задачи и не дожидается завершения выполнения задачи. По данному идентификатору пользователь может оценить статус выполнения задачи.

> Для получения отчета со статусом выполнения задачи и выполнения других действий с задачей необходимо использовать идентификатор задачи. Соответствующие запросы перечислены в разделе «Информация о задачах».

9.7.1.2 Разбиение задачи на подзадачи

В данном разделе описан процесс разбиения задачи на подзадачи. Если для какой-то задачи предполагается только одна подзадача (например, задача Clustering), то задача разобьется на одну подзадачу согласно нижеописанному процессу.

8. Сервис Tasks отправляет данные в Redis.

> Сервис Tasks начинает ожидать появления подзадач в Redis.

> «Рабочий процесс» Tasks ожидает появления данных в Redis для разбиения их на подзадачи.

9. «Рабочий процесс» Tasks получает данные из Redis.
10. «Рабочий процесс» Tasks выполняет процесс разделения на подзадачи.
11. «Рабочий процесс» Tasks отправляет подзадачи обратно в Redis.
12. Сервис Tasks получает подзадачи.
13. Сервис Tasks сохраняет информацию о подзадачах в БД Tasks.
14. Сервис Tasks отправляет подзадачи в Redis.

9.7.1.3 Обработка каждой подзадачи

В данном разделе описан общий процесс обработки подзадачи. Для каждой задачи свой процесс обработки, описанный в соответствующем разделе ниже.

> Сервис Tasks начинает ожидать появления сообщения о начале обработки подзадачи в Redis.

> «Рабочий процесс» Tasks ожидает появления данных в Redis для обработки подзадачи.

15. «Рабочий процесс» Tasks получает данные из Redis.
16. «Рабочий процесс» Tasks начинает обрабатывать подзадачу.
17. «Рабочий процесс» Tasks отправляет сообщение в Redis о том, что обработка началась.
18. Сервис Tasks получает сообщение о начале обработки.
19. Сервис Tasks обновляет информацию о задаче БД Tasks.

> Сервис Tasks начинает ожидать появления результата.

20. «Рабочий процесс» Tasks сохраняет отчет о задаче в сервис Image Store.
21. Сервис Image Store отправляет запрос в хранилище на сохранение отчета.

> Если сервис Image Store отключен, то отчет сохраняться не будет.

22. Сервис Image Store получает ответ от сохранении.
23. Сервис Image Store возвращает ответ «рабочему процессу».
24. «Рабочий процесс» Tasks отправляет сообщение в Redis о том, что подзадача была обработана или о том, что во время обработки произошла какая-то ошибка.
25. Сервис Tasks считывает соответствующее сообщение из Redis.
26. Сервис Tasks обновляет информацию о задаче БД Tasks.
27. После обработки последней подзадачи, сервис Tasks обновляет информацию о задаче БД Tasks.

9.7.1.4 Объединение результатов и завершение обработки

В данном разделе описан общий процесс объединения результатов и завершения обработки. Если для какой-то задачи предполагается только одна подзадача (например, задача Clustering), то объединение результатов выполняться не будет. См. раздел «Завершение обработки задачи» для определенной задачи.

28. Сервис Tasks создает внутреннюю задачу на объединение всех результатов и отправляет эту задачу в Redis.

> Сервис Tasks начинает ожидать появления результата.

> «Рабочий процесс» Tasks ожидает появления задачи на объединение.

29. «Рабочий процесс» Tasks получает данные задачи на объединение.
30. «Рабочий процесс» Tasks объединяет результаты всех подзадач.
31. «Рабочий процесс» сохраняет отчет о задаче в сервис Image Store.
32. Сервис Image Store отправляет запрос в хранилище на сохранение отчета.

> Если сервис Image Store отключен, то отчет сохраняться не будет.

33. Сервис Image Store получает ответ от сохранении.
34. Сервис Image Store возвращает ответ сервису «рабочему процессу».
35. «Рабочий процесс» Tasks отправляет результат объединения в Redis.
36. Сервис Tasks считывает результат объединения из Redis.
37. Сервис Tasks обновляет данные о задаче в БД Tasks.
38. Сервис Tasks обновляет статус задачи в БД Tasks.

9.7.2 Общая диаграмма отмены задач

Диаграмма общего процесса отмены задачи представлена ниже.

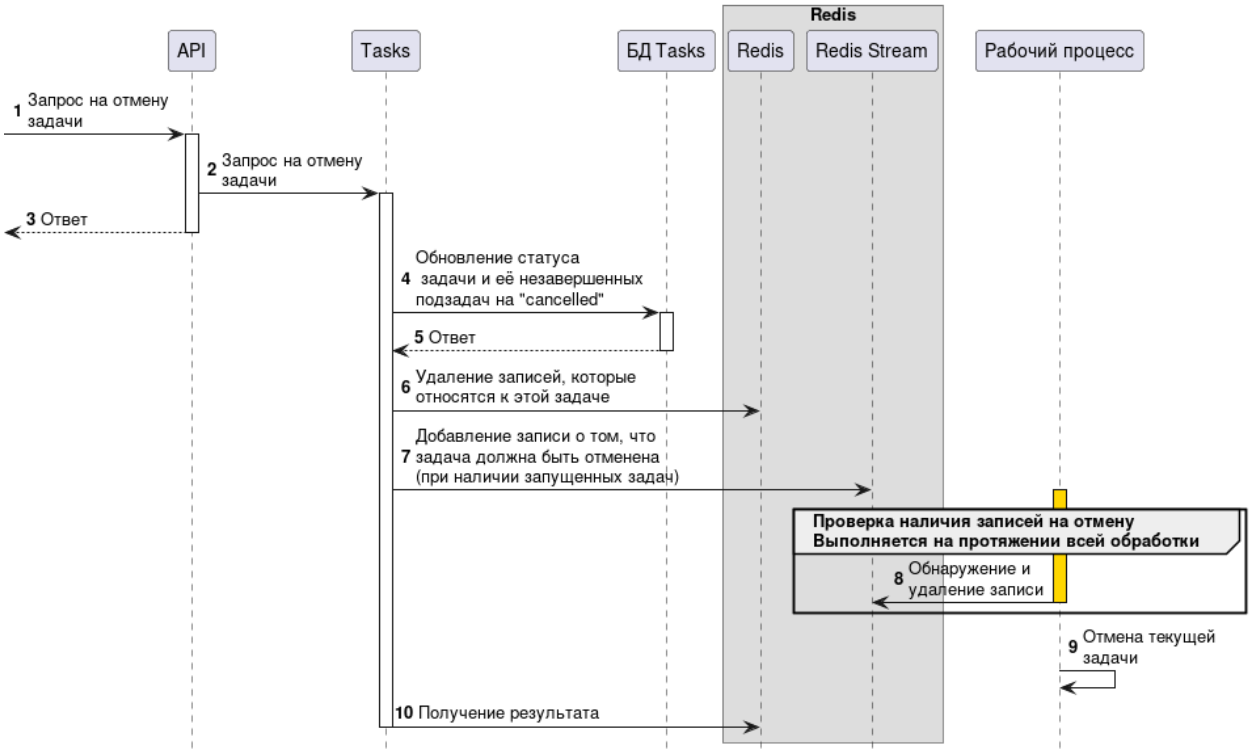


Рис. 77: Диаграмма отмены задачи

1. Пользователь отправляет запрос на отмену задачи в сервис API.
2. Сервис API направляет запрос на отмену задачи в сервис Tasks.
3. Сервис API возвращает ответ, что был активирован процесс отмены задачи.
4. Сервис Tasks обновляет статус задачи и её незавершенных подзадач на «cancelled» в БД Tasks.
5. Сервис Tasks получает ответ.
6. Сервис Tasks удаляет записи в Redis, которые относятся к отменяемой задаче.
7. Сервис Tasks добавляет запись в Redis Stream о том, что задача должна быть отменена, если есть задачи, статус которых находится в «in_progress».
8. «Рабочий процесс» Tasks обнаруживает запись на отмену в Redis Streams и удаляет её.

> «Рабочий процесс» Tasks выполняет проверку наличия записей на протяжении всей обработки.

9. «Рабочий процесс» Tasks отменяет обработку задачи.
10. Сервис Tasks получает результат отмены из Redis.

> Redis используется для хранения данных, включая записи, которые представлены как потоки событий в Redis Streams.

На данном этапе задача считается полностью отмененной, т.е. в ответах на запросы типа «get task» будет возвращен статус «2» («cancelled»).

9.7.3 Диаграмма задачи Clustering

Запрос	Описание	Метод
clustering task	Создать задачу Clustering для лиц или событий в соответствии с заданными фильтрами.	POST

9.7.3.1 Создание задачи Clustering

Создание задачи выполняется стандартным способ, описанным в разделе «Начало создания задачи» в разделе с общей диаграммой работы сервиса Tasks.

9.7.3.2 Разбиение задачи Clustering на подзадачу

Для этого типа задач создается одна подзадача. Подзадача содержит фильтры, в соответствии с которыми выбираются лица или события. Подзадача обрабатывается одним «рабочим процессом».

См. «Разделение задачи на подзадачи» в разделе с общей диаграммой работы сервиса Tasks.

9.7.3.3 Обработка подзадачи Clustering

Обработка подзадачи Clustering зависит от объектов (лиц или событий), указанных в запросе.

Общий процесс работы для обработки подзадачи показан ниже.

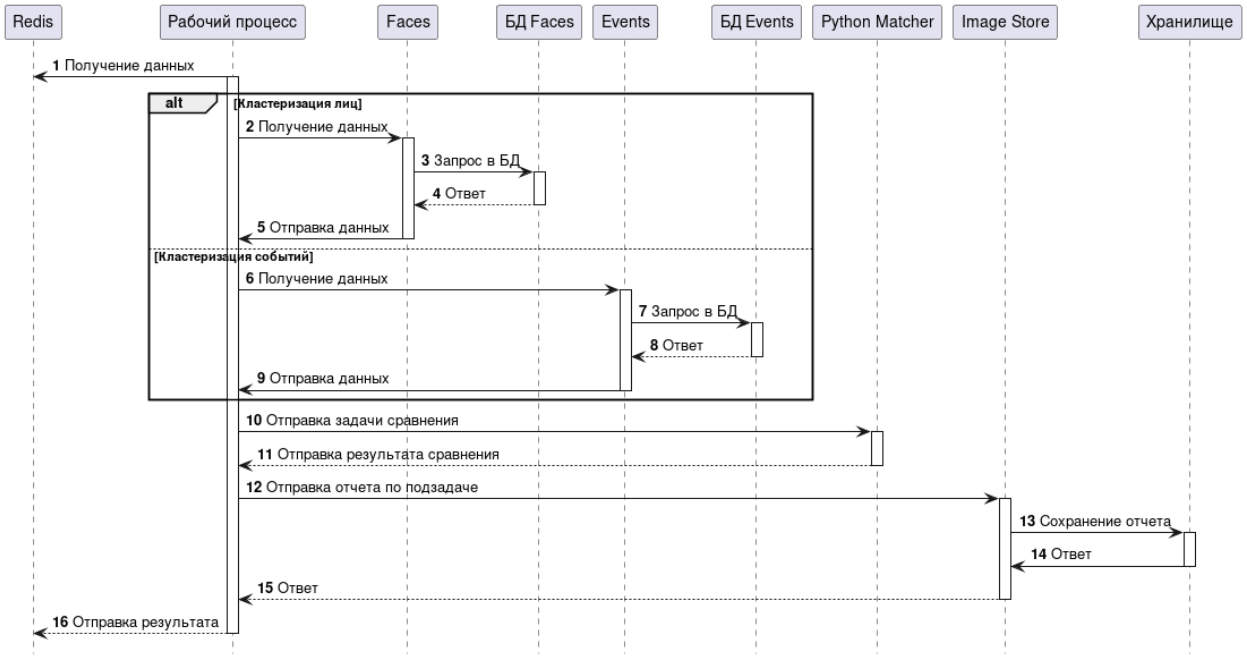


Рис. 78: Диаграмма обработки задачи Clustering

1. «Рабочий процесс» Tasks получает данные из Redis.
2. **Лица:** «Рабочий процесс» Tasks отправляет запрос в сервис Faces. «Рабочий процесс» запрашивает все необходимые идентификаторы атрибутов в соответствии с фильтрами, указанными в подзадаче.
3. **Лица:** Сервис Faces отправляет запрос в БД Faces.
4. **Лица:** Сервис Faces получает ответ.
5. **Лица:** Сервис Faces отправляет идентификаторы «рабочему процессу» Tasks.
6. **События:** «Рабочий процесс» Tasks отправляет запрос в сервис Events. «Рабочий процесс» запрашивает все необходимые идентификаторы атрибутов в соответствии с фильтрами, указанными в подзадаче.
7. **События:** Сервис Events отправляет запрос в БД Events.
8. **События:** Сервис Events получает ответ.
9. **События:** Сервис Events отправляет данные «рабочему процессу» Tasks.
10. Запрос на сравнение «рабочего процесса» Tasks. Обработка запросов выполняется по одной из схем, описанных в разделе «Диаграммы сравнения».
11. Сервис Python Matcher отправляет результаты «Рабочему процессу» Tasks.
12. Сервис Tasks сохраняет отчет о задаче в сервис Image Store.
13. Сервис Image Store отправляет запрос в хранилище на сохранение отчета.

14. Сервис Image Store получает ответ от сохранении.
15. Сервис Image Store возвращает ответ сервису Tasks.
16. «Рабочий процесс» Tasks отправляет результат в Redis.

9.7.3.4 Завершение обработки задачи Clustering

Далее сервис Tasks получает результат из Redis и обновляет статус задачи в БД Tasks.

9.7.4 Диаграммы задачи Linker

Запрос	Описание	Метод
linker task	Создать задачу Linker.	POST

9.7.4.1 Создание задачи Linker

Задачу Linker можно создать для объектов лиц и событий. Процесс создания задачи Linker зависит от типа объекта.

Прикрепление лиц к списку

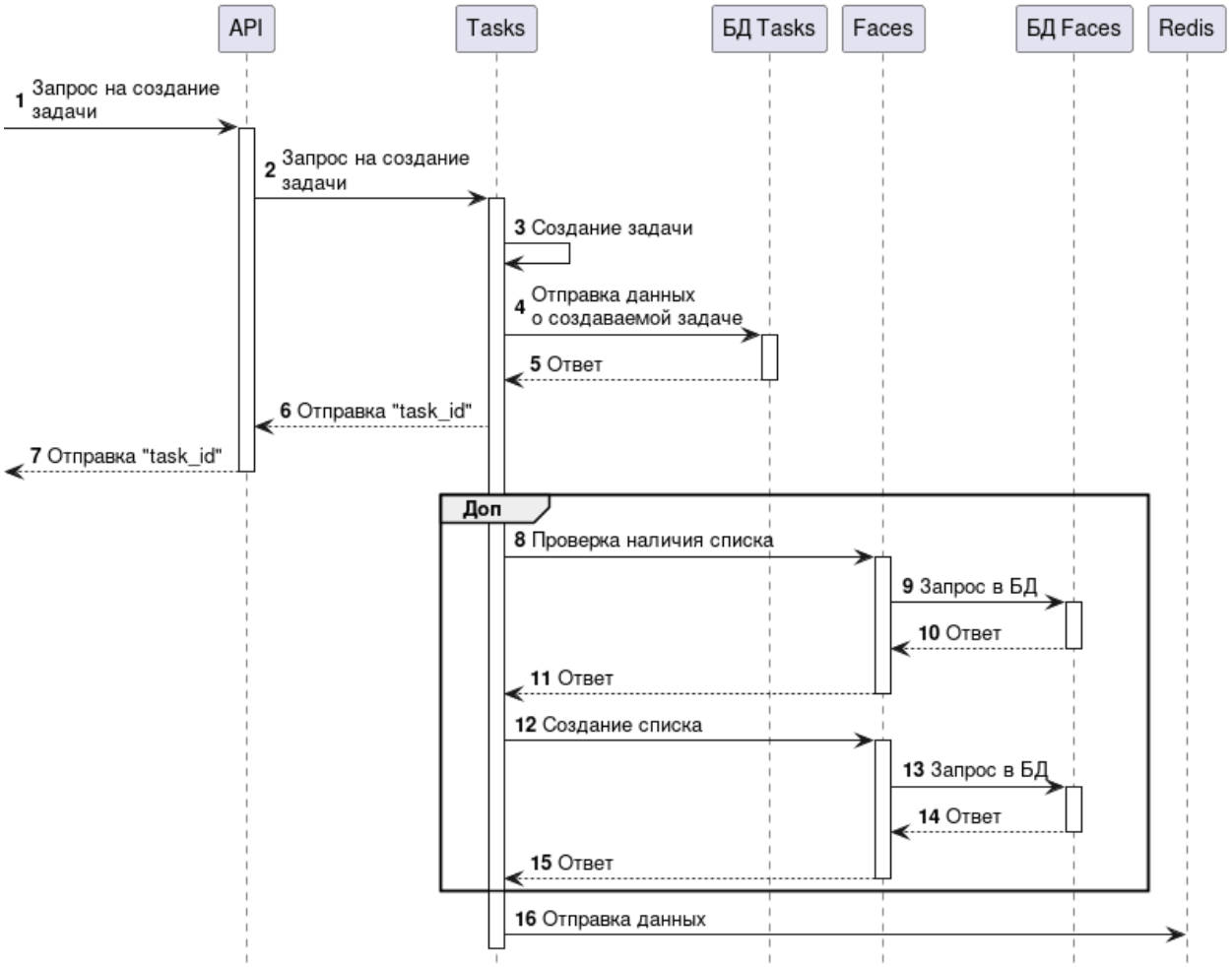


Рис. 79: Диаграмма создания задачи Linker для лиц

1. Сервис API получает запрос на создание задачи.
2. Сервис API отправляет запрос в сервис Tasks.
3. Сервис Tasks создает задачу.
4. Сервис Tasks отправляет информацию в базу данных Tasks.
5. Из базы данных Tasks выдается идентификатор задачи.
6. Идентификатор задачи отправляется в сервис API.
7. Сервис API отправляет в ответ идентификатор задачи.
8. **Дополнительно.** При указании в запросе идентификатора списка, сервис Tasks проверяет наличие списка.
9. **Дополнительно.** Сервис Faces проверяет наличие списка в базе данных Faces.
10. **Дополнительно.** Из базы данных Faces отправляется ответ.

11. **Дополнительно.** Сервис Faces отправляет ответ в сервис Tasks.
12. **Дополнительно.** Если указанный список не существует или в запросе указано создание нового списка, сервис Tasks отправляет запрос на создание нового списка.
13. **Дополнительно.** Faces создает список в базе данных Faces.
14. **Дополнительно.** Из базы данных Faces отправляется ответ.
15. **Дополнительно.** Сервис Faces отправляет ответ в сервис Tasks.
16. Сервис Tasks отправляет данные в Redis.

Прикрепление лиц, созданных на основе событий, к списку

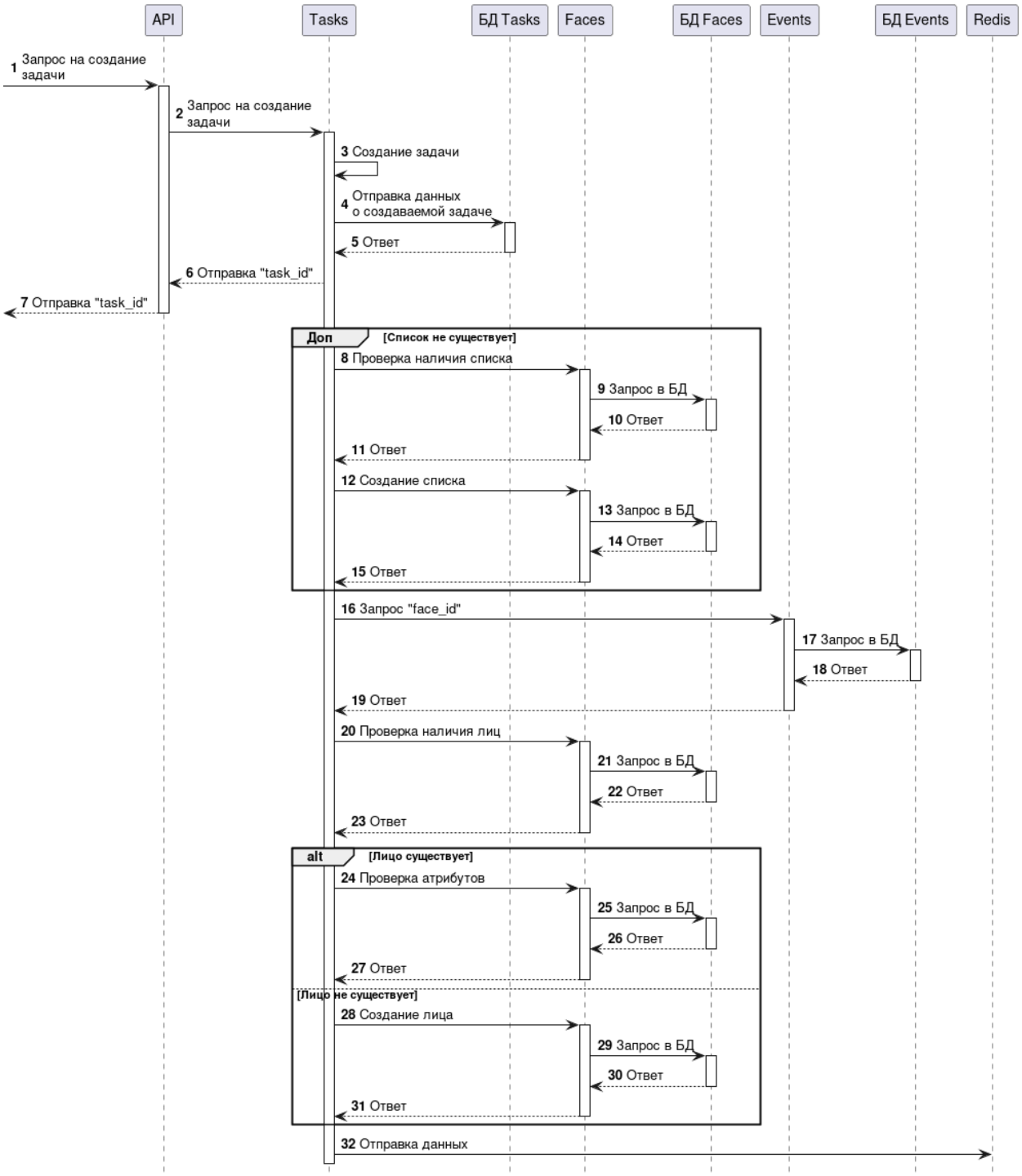


Рис. 80: Диаграмма создания задачи Linker для событий

1. Сервис API получает запрос на создание задачи.
2. Сервис API отправляет запрос в сервис Tasks.
3. Сервис Tasks создает задачу.

4. Сервис Tasks отправляют информацию в базу данных Tasks.
5. Из базы данных Tasks выдается идентификатор задачи.
6. Идентификатор задачи отправляется в сервис API.
7. Сервис API отправляет в ответ идентификатор задачи.
8. **Дополнительно.** При указании в запросе идентификатора списка, сервис Tasks проверяет наличие списка.
9. **Дополнительно.** Сервис Faces проверяет наличие списка в базе данных Faces.
10. **Дополнительно.** Из базы данных Faces отправляется ответ.
11. **Дополнительно.** Сервис Faces отправляет ответ в сервис Tasks.
12. **Дополнительно.** Если указанный список не существует или в запросе указано создание нового списка, сервис Tasks отправляет запрос на создание нового списка.
13. **Дополнительно.** Faces создает список в базе данных Faces.
14. **Дополнительно.** Из базы данных Faces отправляется ответ.
15. **Дополнительно.** Сервис Faces отправляет ответ в сервис Tasks.
16. Сервис Tasks получает идентификаторы лиц и идентификаторы атрибутов из сервиса Events.
17. Сервис Events получает данные из базы данных Events.
18. Из базы данных отправляется ответ.
19. Сервис Faces отправляет идентификаторы в сервис Tasks.
20. Сервис Tasks проверяет наличие лиц в сервисе Faces;
21. Сервис Faces отправляет запрос в базу данных Faces.
22. Из базы данных Faces отправляется информация о существовании лиц.
23. Сервис Faces отправляет данные в сервис Tasks.
24. **Наличие лиц в базе данных Faces** При наличии лица для события сервис Tasks проверяет, чтобы идентификатор атрибута лица совпадал с идентификатором атрибута события.
25. Сервис Faces отправляет запрос в базу данных Faces.
26. Из базы данных Faces отправляются идентификаторы атрибутов.
27. Сервис Faces отправляет идентификаторы атрибутов в сервис Tasks. Если идентификаторы атрибутов для существующего лица и события не совпадают, выдается ошибка «28009: Attribute is not equal».

28. **Лица отсутствуют в базе данных Faces** Если в базе данных Faces нет лиц с указанными идентификаторами, сервис Tasks отправляет запрос на создание новых лиц в базе данных Faces.
29. Сервис Faces отправляет запрос в базу данных Faces.
30. Из базы данных Faces отправляются результаты создания лиц.
31. Сервис Faces отправляет ответ в сервис Tasks.
32. Сервис Tasks отправляет данные в Redis.

9.7.4.2 Разбиение задачи Linker на подзадачи

Для выполнения задачи может использоваться несколько «рабочих процессов». См. «Разделение задачи на подзадачи» в разделе с общей диаграммой работы сервиса Tasks.

9.7.4.3 Обработка подзадач Linker

Общий процесс работы для обработки каждой подзадачи показан ниже.

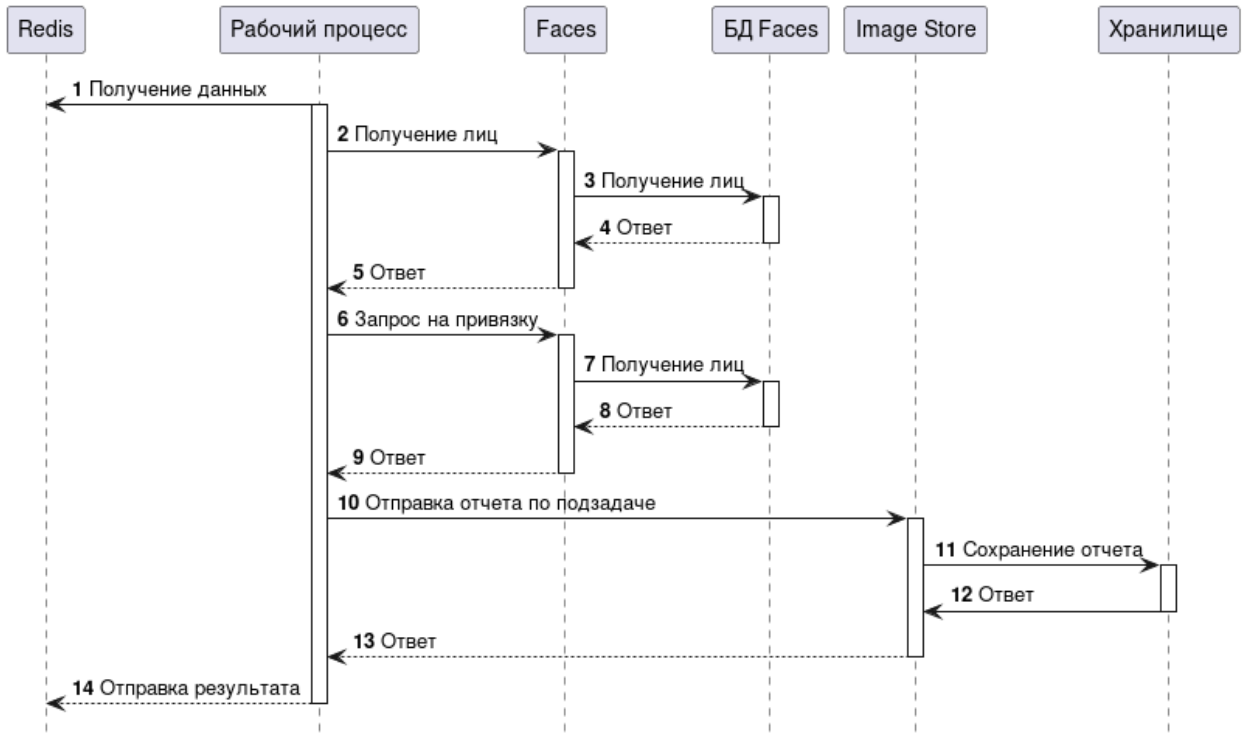


Рис. 81: Диаграмма обработки задачи Linker

1. «Рабочий процесс» Tasks получает данные из Redis.
2. «Рабочий процесс» Tasks запрашивает лица в сервисе Faces в соответствии с полученными идентификаторами лиц.

3. Сервис Faces запрашивает лица в базе данных Faces.
4. Из базы данных отправляются лица в сервис Faces.
5. Сервис Faces отправляет лица «рабочему процессу» Tasks.
6. «Рабочий процесс» Tasks отправляет запросы на прикрепление лиц к указанному списку.
7. Сервис Faces отправляет запросы на прикрепление к базе данных.
8. Из базы данных отправляется результат прикрепления в сервис Faces.
9. Сервис Faces отправляет результат «рабочему процессу» Tasks.
10. «Рабочий процесс» Tasks формирует отчет на основании результатов подзадачи и сохраняет его в сервисе Image Store.
11. Сервис Image Store сохраняет отчет в хранилище.
12. Из хранилища отправляется результат хранения. Результат имеет свой идентификатор.
13. Сервис Image Store выдает результат «рабочему процессу» Tasks.
14. «Рабочий процесс» Tasks отправляет результат в Redis.

9.7.4.4 Завершение обработки задачи Linker

Далее выполняется объединение результатов подзадач и отправка общего отчета в Image Store. См. «Объединение результатов и завершение обработки» в разделе с общей диаграммой работы сервиса Tasks.

9.7.5 Диаграмма задачи Garbage collection

Запрос	Описание	Метод
garbage collection task	Создать задачу на удаление атрибутов, по заданному фильтру времени создания. Удаляются только те атрибуты, которые не прикреплены ни к одному лицу. С помощью этой задачи можно удалить неиспользуемые данные из базы данных Faces.	POST

9.7.5.1 Создание задачи Garbage collection

Создание задачи выполняется стандартным способ, описанным в разделе «Начало создания задачи» в разделе с общей диаграммой работы сервиса Tasks. Единственным отличием является то, что задачу Garbage collection можно создать не только через сервис API, но и через сервис Admin или Tasks.

9.7.5.2 Разбиение задачи Garbage collection на подзадачи

Далее выполняется разделение задачи на подзадачи. Каждая подзадача содержит массив идентификаторов атрибутов. См. «Разделение задачи на подзадачи» в разделе с общей диаграммой работы сервиса Tasks.

9.7.5.3 Обработка подзадачи Garbage collection

Общий процесс работы для обработки каждой подзадачи показан ниже.

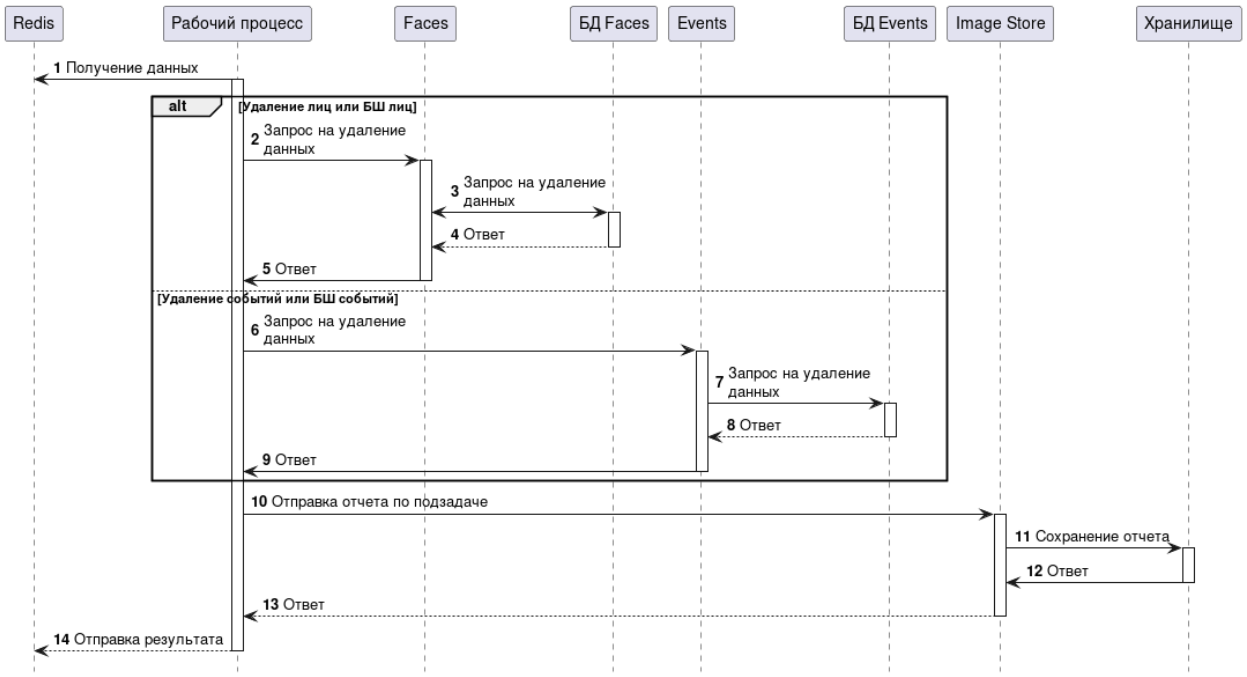


Рис. 82: Диаграмма обработки задачи Garbage collection

1. «Рабочий процесс» Tasks получает данные из Redis.
2. **Удаление лиц или БШ лиц.** В «рабочем процессе» Tasks есть массив идентификаторов атрибутов, которые необходимо удалить. «Рабочий процесс» Tasks запрашивает удаление временных атрибутов и соответствующих БШ в сервисе Faces.
3. Сервис Faces отправляет запрос в базу данных Faces на удаление атрибутов с указанными идентификаторами.
4. Сервис Faces получает ответ.
5. Сервис Faces отправляет ответ «рабочему процессу» Tasks.
6. **Удаление событий или БШ событий.** В «рабочем процессе» Tasks есть массив идентификаторов атрибутов, которые необходимо удалить. «Рабочий процесс» Tasks запрашивает удаление временных атрибутов и соответствующих БШ в сервисе Events.

7. Сервис Events отправляет запрос в базу данных Events на удаление атрибутов с указанными идентификаторами.
8. Сервис Events получает ответ.
9. Сервис Events отправляет ответ «рабочему процессу» Tasks.
10. «Рабочий процесс» Tasks формирует отчет на основании результатов подзадачи и сохраняет его в сервисе Image Store.
11. Сервис Image Store сохраняет отчет в хранилище.
12. Из хранилища отправляется результат хранения. Результат имеет свой идентификатор.
13. Сервис Image Store выдает результат «рабочему процессу» Tasks.
14. «Рабочий процесс» Tasks отправляет результат в Redis.

9.7.5.4 Завершение обработки задачи Garbage collection

Далее выполняется объединение результатов подзадач и отправка отчета в Image Store. См. «Объединение результатов и завершение обработки» в разделе с общей диаграммой работы сервиса Tasks.

9.7.6 Диаграмма задачи Reporter

Запрос	Описание	Метод
reporter task	Создать отчет для задачи Clustering. Отчет в формате CSV. Можно указать столбцы, которые необходимо добавить в отчет. Отчет находится в ZIP-архиве и содержит изображения аватаров для каждого объекта в кластере.	POST

9.7.6.1 Создание задачи Reporter

Создание задачи выполняется стандартным способ, описанным в разделе «Начало создания задачи» в разделе с общей диаграммой работы сервиса Tasks.

9.7.6.2 Разбиение задачи Garbage collection на подзадачу

Для этого типа задач создается одна подзадача. Подзадача содержит фильтры для данных, которые необходимо получить для включения в отчет. См. «Разделение задачи на подзадачи» в разделе с общей диаграммой работы сервиса Tasks.

9.7.6.3 Обработка задачи Reporter

Обработка запроса зависит от данных, хранящихся в отчете о кластеризации. Если отчет был создан для лиц, данные лица будут запрошены из сервиса Faces и добавлены в отчет. Если отчет был создан для событий, данные события будут запрошены из сервиса Events и добавлены в отчет.

Общий процесс работы для обработки подзадачи показан ниже.

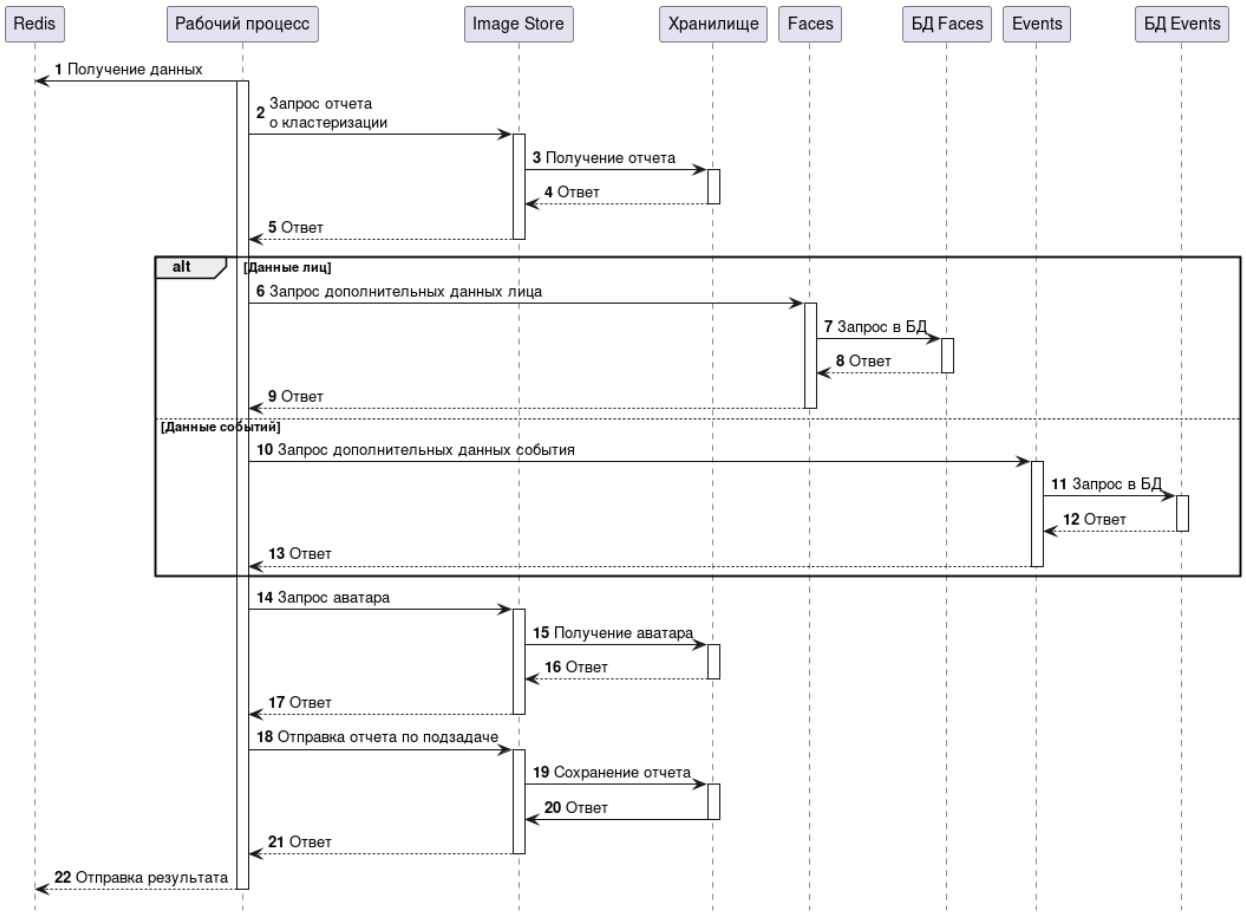


Рис. 83: Диаграмма обработки задачи Reporter

1. «Рабочий процесс» Tasks получение данных из Redis.
2. «Рабочий процесс» Tasks запрашивает отчет о кластеризации в Image Store.
3. Сервис Image Store запрашивает отчет из хранилища.
4. Из хранилища отправляется ответ.
5. Сервис Image Store отправляет отчет.
6. **Отчет по лицам**. «Рабочий процесс» Tasks получает все идентификаторы лица из кластера и запрашивает дополнительные данные лица из сервиса Faces. Запрошенные данные зависят от столбцов, указанных в запросе.

7. Сервис Faces запрашивает данные из базы данных Faces.
8. Из базы данных Faces отправляются необходимые данные.
9. Сервис Faces отправляет данные «рабочему процессу» Tasks.
10. **Отчет по событиям**. «Рабочий процесс» Tasks запрашивает дополнительные данные о событиях из сервиса Events. Запрошенные данные зависят от столбцов, указанных в запросе.
11. Сервис Events запрашивает данные из базы данных Events.
12. Из базы данных Events отправляются необходимые данные.
13. Сервис Events отправляет данные «рабочему процессу» Tasks.
14. «Рабочий процесс» запрашивает изображение аватара для каждого лица или события. Изображение для лица указывается в поле «аватар» базы данных Faces. Его можно хранить в хранилище Image Store или любом другом. Изображение для события – это соответствующий биометрический образец из хранилища Image Store.
15. Image Store запрашивает изображение из хранилища.
16. Из хранилища отправляется изображение.
17. Из сервиса Image Store отправляется изображение.
18. Из «рабочего процесса» отправляется отчет в Image Store.
19. С помощью Image Store отчет сохраняется в хранилище.
20. Хранилище отправляет ответ о сохранении.
21. Сервис Image Store отправляет ответ «рабочему процессу» Tasks.
22. «Рабочий процесс» Tasks отправляет результат в Redis.

9.7.6.4 Завершение обработки задачи Reporter

Далее сервис Tasks получает результат из Redis и обновляет статус задачи в БД Tasks.

9.7.7 Диаграмма задачи Exporter

Запрос	Описание	Метод
exporter task	Собрать данные о событиях/лицах и экспортировать в файл CSV. Экспортированный CSV файл находится в ZIP-архиве и содержит изображения аватаров для каждого объекта.	POST

9.7.7.1 Создание задачи Exporter

Создание задачи выполняется стандартным способ, описанным в разделе «Начало создания задачи» в разделе с общей диаграммой работы сервиса Tasks.

9.7.7.2 Разбиение задачи Exporter на подзадачу

Для этого типа задач создается одна подзадача. См. «Разделение задачи на подзадачи» в разделе с общей диаграммой работы сервиса Tasks.

9.7.7.3 Обработка подзадачи Exporter

Общий процесс работы для обработки подзадачи показан ниже.

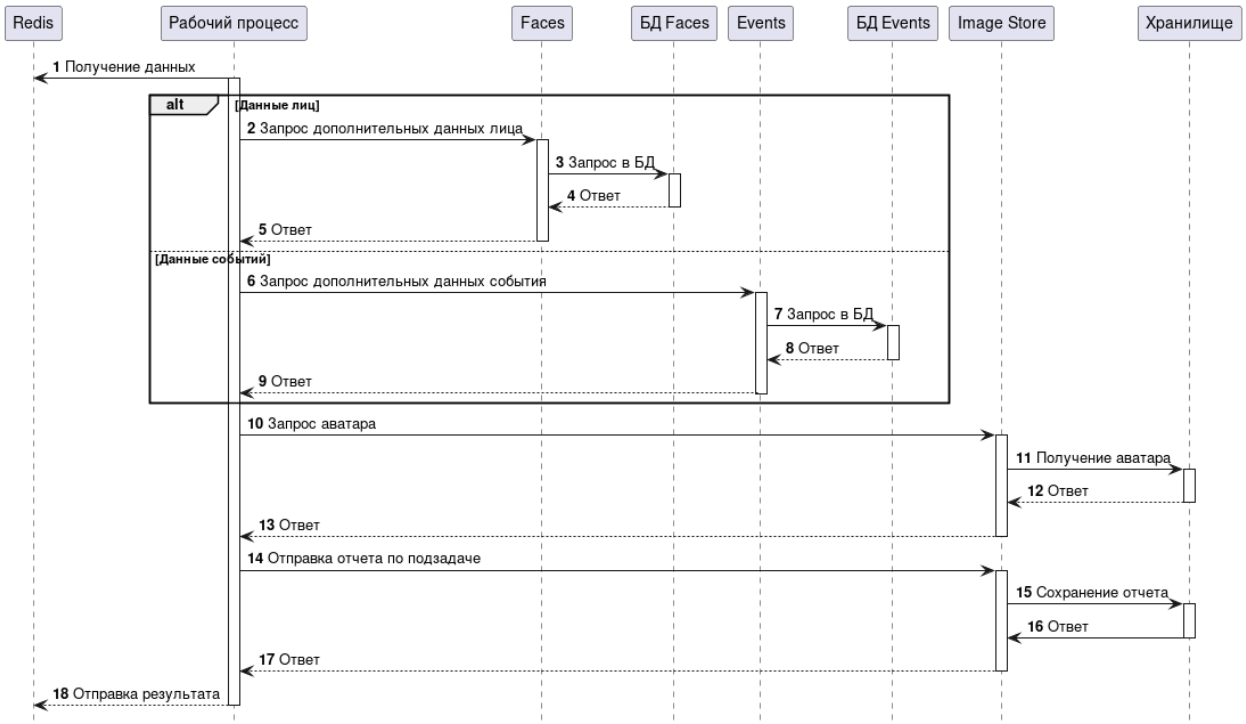


Рис. 84: Диаграмма обработки задачи Exporter

1. «Рабочий процесс» Tasks получение данных из Redis.
2. **Экспорт данных для лиц**. «Рабочий процесс» Tasks получает все идентификаторы лица и запрашивает дополнительные данные лица из сервиса Faces. Запрошенные данные зависят от столбцов, указанных в запросе.
3. Сервис Faces запрашивает данные из базы данных Faces.
4. Из базы данных Faces отправляются необходимые данные.
5. Сервис Faces отправляет данные «рабочему процессу» Tasks.

6. **Экспорт данных для событий**. «Рабочий процесс» Tasks запрашивает дополнительные данные о событиях из сервиса Events. Запрошенные данные зависят от столбцов, указанных в запросе.
7. Сервис Events запрашивает данные из базы данных Events.
8. Из базы данных Events отправляются необходимые данные.
9. Сервис Events отправляет данные «рабочему процессу» Tasks.
10. «Рабочий процесс» запрашивает изображение аватара для каждого лица или события. Изображение для лица указывается в поле «аватар» базы данных Faces. Его можно хранить в хранилище Image Store или любом другом. Изображение для события – это соответствующий образец из хранилища Image Store.
11. Image Store запрашивает изображение из хранилища.
12. Из хранилища отправляется изображение.
13. Из сервиса Image Store отправляется изображение.
14. С «рабочего процесса» отправляются экспортированные данные в Image Store.
15. С помощью Image Store отчет сохраняются экспортированные данные в хранилище.
16. Хранилище отправляет ответ о сохранении.
17. Сервис Image Store отправляет ответ «рабочему процессу» Tasks.
18. «Рабочий процесс» Tasks отправляет результат в Redis.

9.7.7.4 Завершение обработки задачи Exporter

Далее сервис Tasks получает результат из Redis и обновляет статус задачи в БД Tasks.

9.7.8 Диаграмма задачи Cross-matching

Запрос	Описание	Метод
cross-matching task	Создать задачу Cross-matching. С помощью данной задачи можно сравнить события и лица по заданным фильтрам. Лица и события могут быть указаны как эталоны и как кандидаты. Можно задать дополнительные фильтры, как для эталонов, так и для кандидатов.	POST

9.7.8.1 Создание задачи Cross-matching

Создание задачи выполняется стандартным способ, описанным в разделе «Начало создания задачи» в разделе с общей диаграммой работы сервиса Tasks.

9.7.8.2 Разбиение задачи Cross-matching на подзадачу

Для этого типа задач создается одна подзадача. Подзадача содержит все необходимые фильтры эталонов и кандидатов. См. «Разделение задачи на подзадачи» в разделе с общей диаграммой работы сервиса Tasks.

9.7.8.3 Обработка подзадач Cross-matching

Выполнение подзадач Cross-matching может варьироваться в зависимости от указанных эталонов и кандидатов.

Их можно указать с помощью лиц и/или событий. Далее на диаграммах запросы к сервисам Faces и Events отмечены как альтернативные. Запросы к сервису Faces используются, когда лица заданы в качестве эталонов или кандидатов. Запросы к сервису Events используются, когда события заданы в качестве эталонов или кандидатов.

Обработка задачи Clustering зависит от выбранных объектов (лиц или событий).

Общий процесс работы для обработки каждой подзадачи показан ниже.

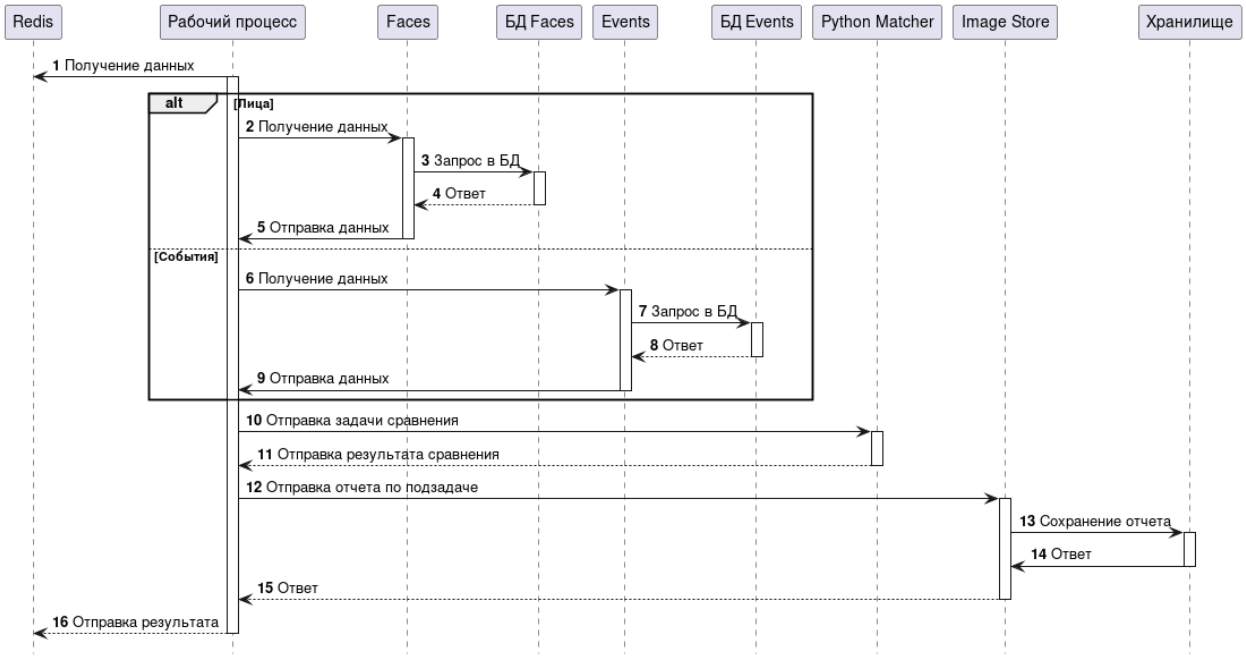


Рис. 85: Диаграммы обработки задачи Cross-matching

1. «Рабочий процесс» Tasks получение данных из Redis.

2. **Лица заданы в качестве эталонов или кандидатов**. «Рабочий процесс» Tasks отправляет запрос в сервис Faces для получения идентификаторов атрибутов для лиц. Лица отбираются согласно заданным фильтрам;
3. Сервис Faces запрашивает идентификаторы в базе данных Faces.
4. Из базы данных Faces отправляются идентификаторы атрибутов.
5. Сервис Faces отправляет идентификаторы атрибутов «рабочему процессу» Tasks.
6. **События заданы в качестве эталонов или кандидатов**. «Рабочий процесс» Tasks отправляет запрос в сервис Events для получения идентификаторов атрибутов для событий. События отбираются согласно заданным фильтрам;
7. Сервис Events запрашивает идентификаторы событий в базе данных Events.
8. Из базы данных отправляются необходимые идентификаторы.
9. Сервис Events отправляет идентификаторы «рабочему процессу» Tasks.
10. Запрос на сравнение «рабочего процесса» Tasks. Обработка запросов выполняется по одной из схем, описанных в разделе «Диаграммы сравнения».
11. Сервис Python Matcher отправляет результаты.
12. Из «рабочего процесса» отправляется отчет в Image Store.
13. С помощью Image Store отчет сохраняется в хранилище.
14. Хранилище отправляет ответ о сохранении.
15. Сервис Image Store отправляет ответ «рабочему процессу» Tasks.
16. «Рабочий процесс» Tasks отправляет результат в Redis.

9.7.8.4 Завершение обработки задачи Cross-matching

Далее сервис Tasks получает результат из Redis и обновляет статус задачи в БД Tasks.

9.7.9 Диаграмма задачи Estimator

Запрос	Описание	Метод
estimator task	Создать задачу Estimator. С помощью данной задачи можно выполнять пакетную обработку изображений с указанием заданных политик.	POST

9.7.9.1 Создание задачи Estimator

Создание задачи выполняется стандартным способ, описанным в разделе «Начало создания задачи» в разделе с общей диаграммой работы сервиса Tasks.

9.7.9.2 Разбиение задачи Estimator на подзадачу

Для этого типа задач создается одна подзадача. Подзадача содержит все необходимые данные. См. «Разделение задачи на подзадачи» в разделе с общей диаграммой работы сервиса Tasks.

9.7.9.3 Обработка подзадачи Estimator

Общий процесс работы для обработки подзадачи показан ниже.

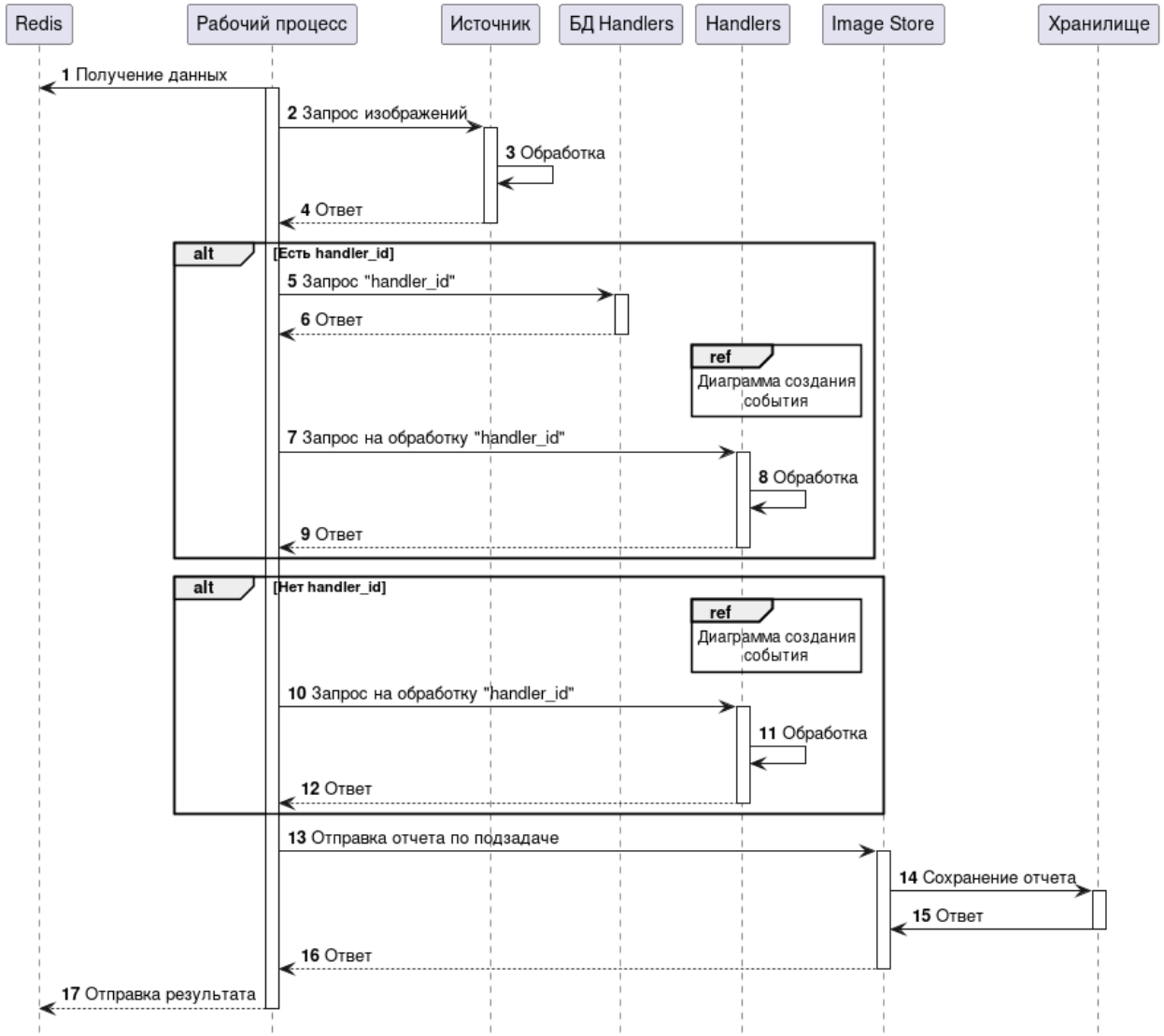


Рис. 86: Диаграмма обработки задачи Estimator

1. «Рабочий процесс» Tasks получение данных из Redis.
2. «Рабочий процесс» запрашивает пакет изображений из следующих источников:
    - сервиса Image Store в ZIP-архиве если в качестве источника изображений выбран тип «zip»;
    - хранилища S3 если в качестве источника изображений выбран тип «s3»;
    - FTP-сервера если в качестве источника изображений выбран тип «ftp»;
    - хранилища Samba если в качестве источника изображений выбран тип «samba»;
    - директории сетевого диска, которая смонтирована на локальный диск и синхронизирована с директорией в контейнерах сервиса Tasks и его «рабочего процесса» (см. подробное описание в разделе «Задача Estimator»).
3. Обработка параметров, указанных в запросе — авторизация, использование префиксов и т.п.
4. Image Store/S3/Сетевой диск/FTP-сервер/Samba отправляет изображения «рабочему процессу» Tasks.
5. **Есть handler_id**. «Рабочий процесс» Tasks отправляет запрос на получение идентификатора существующего обработчика в базу данных Handlers.
6. **Есть handler_id**. Из базы данных Handlers «рабочему процессу» приходит ответ с идентификатором handler_id.
7. **Есть handler_id**. «Рабочий процесс» отправляет запрос в сервис Handlers для обработки handler_id.
8. **Есть handler_id**. Сервис Handlers обрабатывает пакет изображений по полученному handler_id в соответствии с заданными политиками (см. диаграмму создания события).
9. **Есть handler_id**. Полученный ответ возвращается «рабочему процессу» Tasks.
10. **Нет handler_id**. «Рабочий процесс» Tasks отправляет запрос на обработку политик, указанных прямо в запросе, в сервис Handlers.
11. **Нет handler_id**. Сервис Handlers обрабатывает пакет изображений в соответствии с заданными политиками (см. диаграмму создания события).
12. **Нет handler_id**. Полученный ответ возвращается «рабочему процессу» Tasks.
13. Из «рабочего процесса» отправляется отчет в Image Store.
14. С помощью Image Store отчет сохраняется в хранилище.
15. Хранилище отправляет ответ о сохранении.
16. Сервис Image Store отправляет ответ «рабочему процессу» Tasks.
17. «Рабочий процесс» Tasks отправляет результат в Redis.

9.7.9.4 Завершение обработки задачи Estimator

Далее сервис Tasks получает результат из Redis и обновляет статус задачи в БД Tasks.

9.7.10 Диаграммы задачи Additional extraction

Запрос	Описание	Метод
create additional extraction task	Извлечь все существующие биометрические шаблоны с использованием новой версии нейросети.	POST

9.7.10.1 Создание задачи Additional extraction

Запрос отправляется с помощью сервиса Admin. Диаграмма создания задачи приведена ниже.

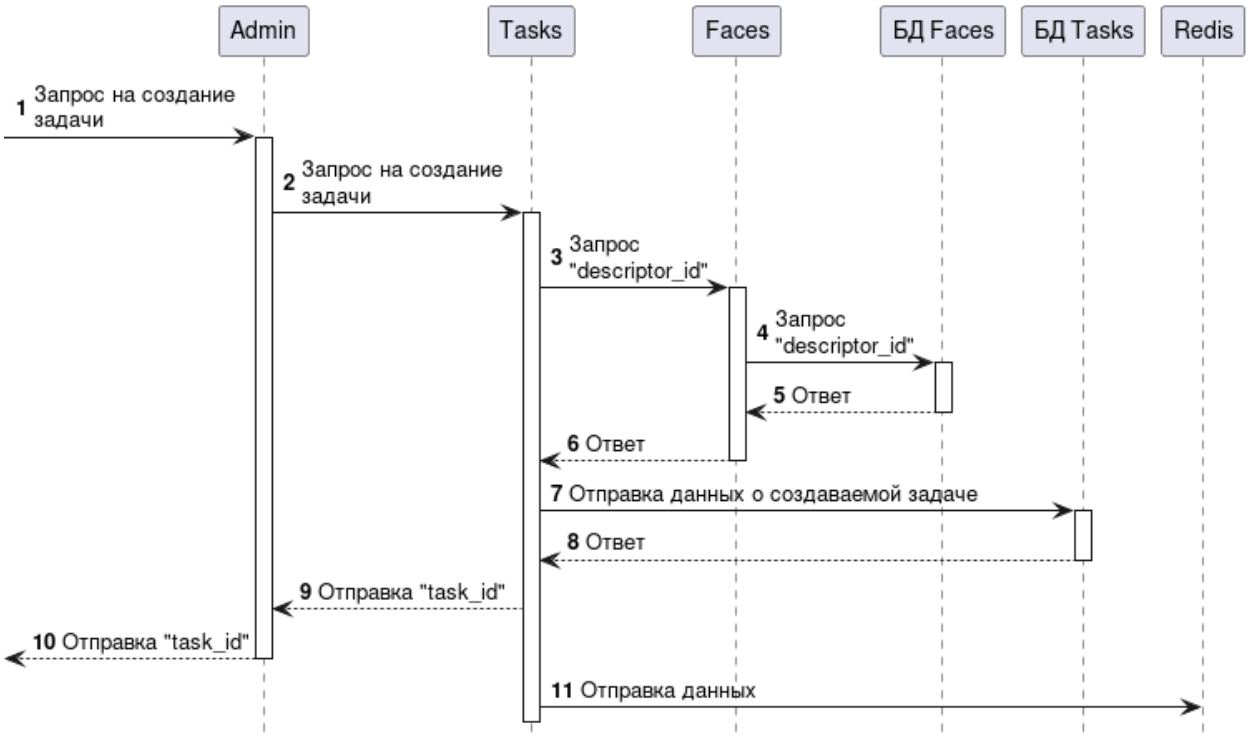


Рис. 87: Диаграмма создания задачи Additional extraction

1. Администратор отправляет запрос в сервис Admin с помощью пользовательского интерфейса администратора или любого другого приложения.
2. Сервис Admin отправляет запрос в сервис Tasks.
3. Сервис Tasks отправляет запрос к сервису Faces на получение идентификаторов биометрических шаблонов, которые не были извлечены с помощью новой нейросети.

4. Сервис Faces запрашивает необходимые идентификаторы.
5. Сервис Faces получает необходимые идентификаторы.
6. Сервис Faces отправляет идентификаторы в сервис Tasks.
7. Сервис Tasks отправляет данные о создаваемой задаче в базу данных Tasks
8. Сервис Tasks получает «task_id».
9. Сервис Tasks отправляет идентификатор задачи в сервис Admin.
10. Сервис Admin отправляет созданный идентификатор задачи в пользовательский интерфейс или приложение, используемое администратором.
11. Сервис Tasks отправляет данные в Redis, откуда «рабочий процесс» заберет задачу на дальнейшую обработку.

9.7.10.2 Разделение задачи Additional extraction на подзадачи

Далее выполняется разделение задачи на подзадачи. Каждая из подзадач содержит массив идентификаторов атрибутов. См. «Разделение задачи на подзадачи» в разделе с общей диаграммой работы сервиса Tasks.

9.7.10.3 Обработка подзадач Additional extraction

Общий процесс работы для обработки каждой подзадачи показан ниже.

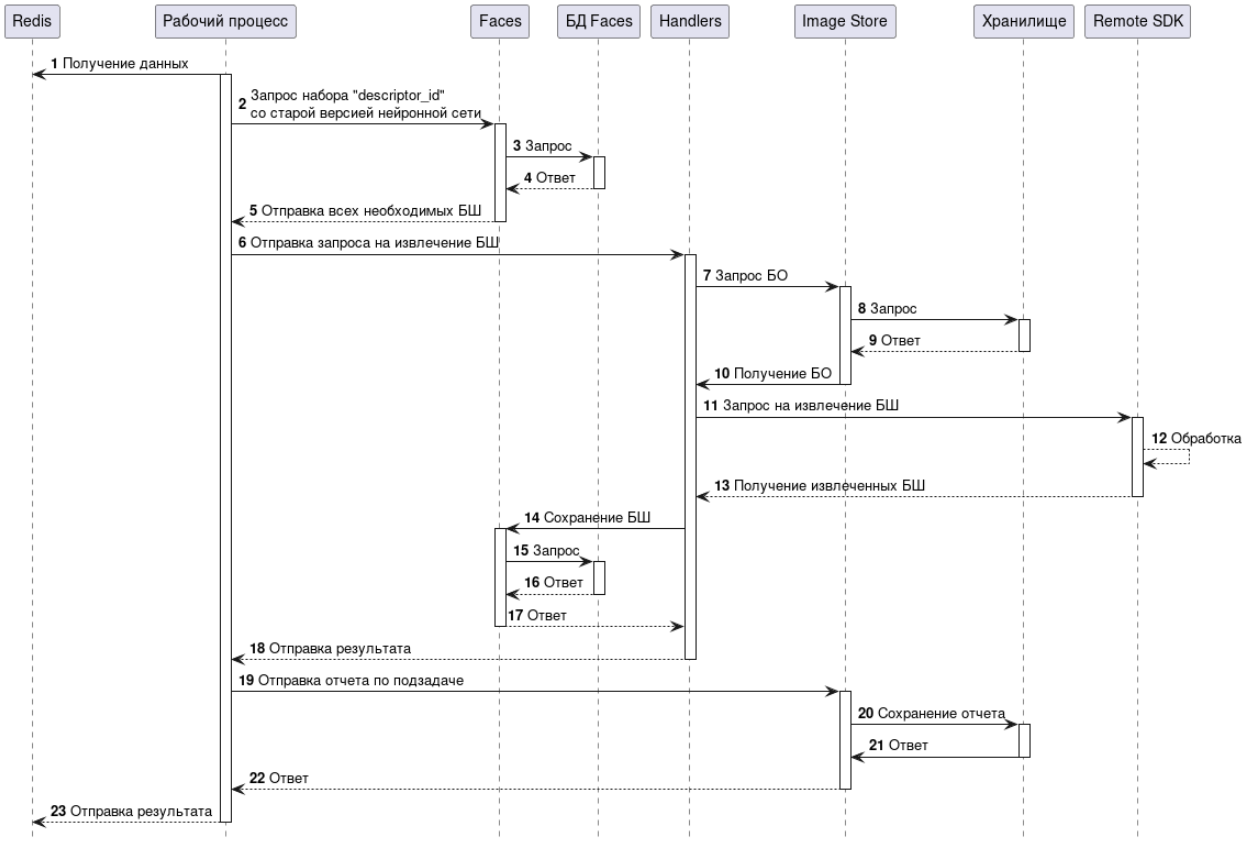


Рис. 88: Диаграмма обработки задачи Additional extraction

1. «Рабочий процесс» Tasks получает данные из Redis.
2. «Рабочий процесс» Tasks отправляет запрос в сервис Faces на получение всех требуемых биометрических шаблонов в соответствии с их идентификаторами.
3. Сервис Faces отправляет запрос в базу данных.
4. Из базы данных отправляются обратно необходимые БШ.
5. Сервис Faces отправляет БШ «рабочему процессу» Tasks.
6. Запрос на извлечение БШ отправляется в сервис Handlers.
7. Сервис Handlers запрашивает биометрические образцы для извлечения из Image Store.
8. Сервис Image Store запрашивает биометрические образцы из хранилища.
9. Сервис Image Store получает биометрические образцы из хранилища.
10. Сервис Image Store отправляет биометрические образцы в сервис Handlers.
11. Сервис Handlers отправляет запрос на извлечение БШ в сервис Remote SDK.
12. Сервис Remote SDK извлекает биометрические шаблоны.

13. Из сервиса Remote SDK отправляются извлеченные БШ.
14. Сервис Handlers отправляет БШ и базовые атрибуты в сервис Faces.
15. Сервис Faces отправляет запрос на сохранение данных в базу данных.
16. Из базы данных выдается результат сохранения данных.
17. Сервис Faces выдает результат сохранения данных.
18. Сервис Handlers отправляет результат выполнения задачи в сервис Tasks.
19. «Рабочий процесс» Tasks отправляет результат в Redis.

Описанная последовательность выполняется для каждого «рабочего процесса» Tasks.

9.7.10.4 Завершение обработки задачи Additional extraction

Далее выполняется объединение результатов подзадач и отправка отчета в Image Store. См. «Объединение результатов и завершение обработки» в разделе с общей диаграммой работы сервиса Tasks.

9.7.11 Диаграммы предоставления информации о задачах

С помощью запросов, приведенных ниже, предоставляется информация о статусе существующих задач.

Запрос	Описание	Метод
cancel task	Отменить выполнение задачи по ее task_id.	PATCH
get tasks	Получить информацию о задачах. Можно установить фильтры для задач.	GET
get tasks count	Получить количество задач по заданным фильтрам.	GET
get task	Получить информацию о задаче по её task_id	GET
get subtasks	Получить информацию обо всех подзадачах заданной задачи.	GET

С помощью запросов, приведенных ниже, можно предоставить информацию об ошибках, возникших при обработке задач.

Запрос	Описание	Метод
get errors of task	Получить информацию обо всех ошибках заданной задачи по task_id.	GET

Запрос	Описание	Метод
get errors	Получить информацию обо всех ошибках по заданным фильтрам.	GET
get errors count	Получить количество ошибок по заданному фильтру.	GET
get error	Получить информацию об ошибке по task_id.	GET

Соответствующая диаграмма представлена ниже.

Рис. 89: Диаграмма получения информации о задаче или отмене задачи

1. Запрос отправляется в API.
2. Сервис API отправляет запрос в сервис на получение требуемых данных или выполнение требуемых действий.
3. Сервис Tasks отправляет запрос в базу данных Tasks на получение требуемых данных или выполнение требуемых действий.
4. Из базы данных отправляется ответ в сервис Tasks.
5. Сервис Tasks отправляет ответ в сервис API.
6. Сервис API выдает информацию.

Запрос	Описание	Метод
delete task	Удалить заданную задачу и ее результаты.	DELETE
get task result	Получить результаты заданной задачи.	GET

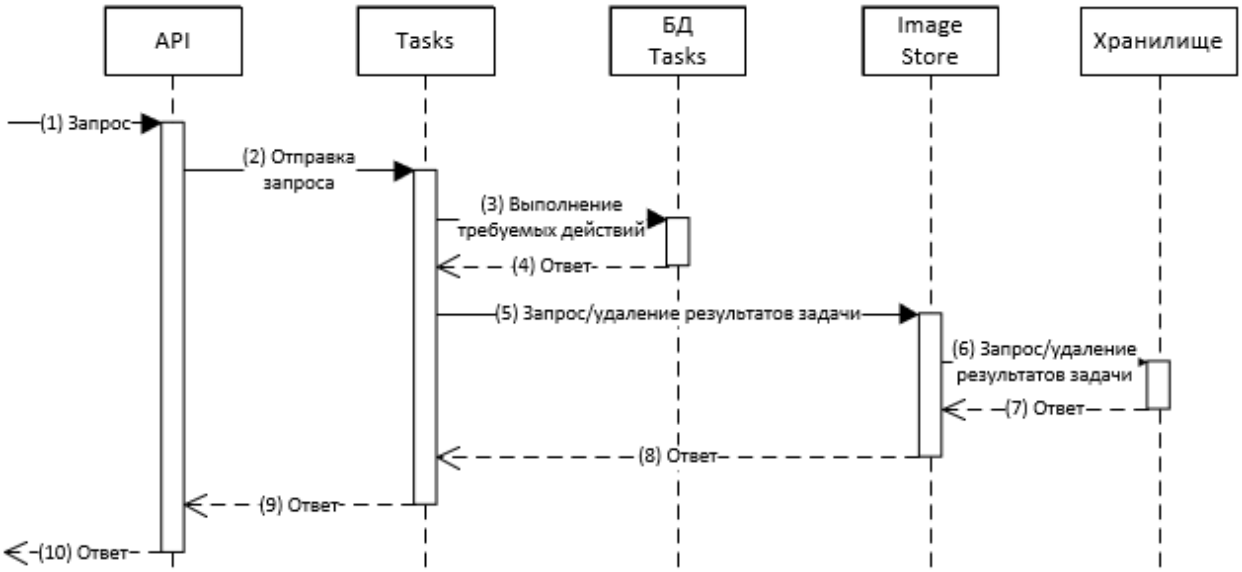


Рис. 90: Диаграмма удаления задачи/получения результата задачи

1. Запрос отправляется в API.
2. Сервис API отправляет запрос в сервис Tasks на получение требуемых данных задачи или удаление задачи.
3. Сервис Tasks отправляет запрос в базу данных Tasks на получение требуемых данных или удаление задачи.
4. Из базы данных отправляется ответ в сервис Tasks.
5. Сервис Tasks отправляет запрос в Image Store для получения или удаления результатов заданной задачи.
6. Сервис Image Store отправляет запрос в хранилище.
7. Из хранилища отправляется ответ в сервис Image Store.
8. Сервис Image Store выдает результат в сервис Tasks.
9. Сервис Tasks отправляет ответ в сервис API.
10. Сервис API отправляет ответ.

9.8 Диаграммы lambda

9.8.1 Диаграмма создания lambda

Данная диаграмма описывает общий процесс создания lambda.

Запрос	Описание	Метод
create lambda	Создать lambda.	POST

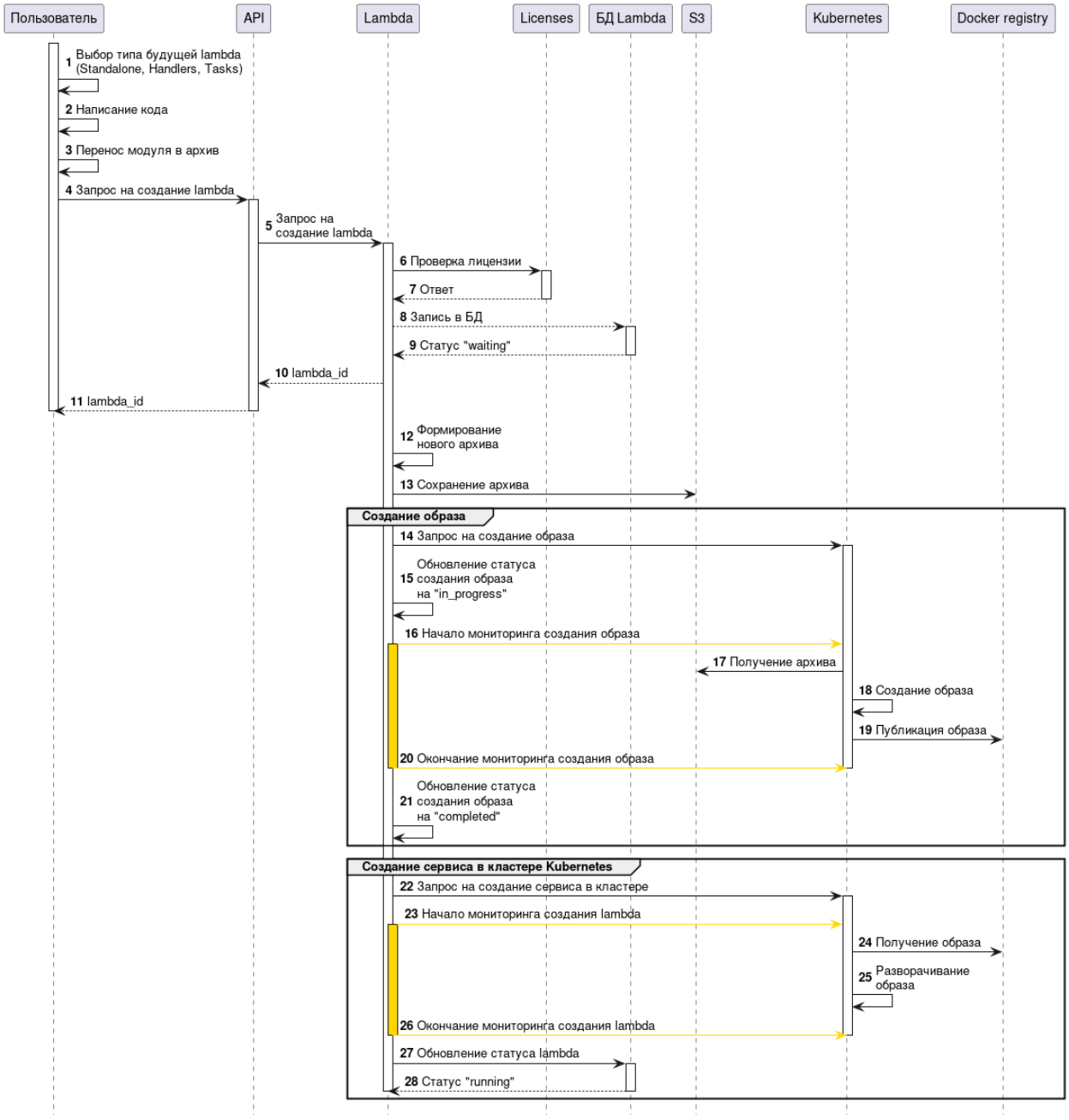


Рис. 91: Диаграмма создания lambda

Общая схема обработки запроса:

1. Пользователь определяет какой тип lambda ему будет нужен — Handlers-lambda, Standalone-lambda, Tasks-lambda.
2. Пользователь пишет Python-код в соответствии с типом будущей lambda и требованиями к коду.
3. Пользователь переносит модуль в архив в соответствии с требованиями к архиву.
4. Пользователь выполняет запрос «create lambda» к сервису API.
5. Сервис API отправляет запрос в сервис Lambda.
6. Сервис Lambda отправляет запрос на проверку лицензии в сервис Licenses.
7. Сервис Lambda получает ответ.
8. Сервис Lambda записывает данные в БД Lambda, обновляя статус создания lambda на «waiting».
9. Сервис Lambda получает ответ.

> Во время создания lambda доступна возможность получения статуса её создания с помощью запроса «get lambda status».

10. Сервис Lambda возвращает сервису API идентификатор «lambda_id».
11. Сервис API возвращает пользователю идентификатор «lambda_id».
12. Сервис Lambda добавляет дополнительные файлы и формирует новый архив.
13. Сервис Lambda сохраняет новый архив в хранилище S3.

Создание образа

14. Сервис Lambda оправляет запрос на создание образа в Kubernetes.
15. Сервис Lambda обновляет статус создания образа на «in_progress».
16. Сервис Lambda начинает отслеживать статус создания образа в Kubernetes.

> Пользователь может получить статус и логи создания образа с помощью запросов «get lambda image creation status» и «get lambda image creation logs».

17. Kubernetes получает архив из хранилища S3.
18. Kubernetes создает образ.
19. Kubernetes публикует образ в Docker registry.

> Настройки S3, Docker registry и Kubernetes задаются в настройках сервиса Lambda.

20. Сервис Lambda фиксирует публикацию образа.
21. Сервис Lambda обновляет статус создания образа на «completed».

Создание lambda в кластере Kubernetes

22. Сервис Lambda отправляет запрос на создание lambda в кластере Kubernetes.
23. Сервис Lambda начинает отслеживать статус создания lambda в Kubernetes.

> Во время создания lambda невозможно получить статус и логи создания образа. Статус создания и логи создания lambda можно получить с помощью запросов «get lambda status» и «get lambda logs».

24. Kubernetes получает образ из Docker registry.
25. Kubernetes разворачивает полученный образ.
26. Сервис Lambda фиксирует состояние развернутого образа.
27. Сервис Lambda обновляет статус lambda на «running» в БД Lambda.
28. Сервис Lambda получает ответ.

Далее можно приступить к отправке запросов lambda.

9.8.2 Диаграмма обработки lambda

Данная диаграмма описывает общий процесс обработки lambda. Процесс обработки lambda зависит от её типа — Standalone-lambda, Handlers-lambda или Tasks-lambda.

Обработка типа Tasks-lambda не отражена на данной диаграмме. Она выполняется согласно общему процессу создания задач, **за исключением того**, что вместо «рабочего процесса» Tasks используется сервис Lambda.

Для типов Handlers-lambda и Standalone-lambda можно выполнять запросы «proxy post request to lambda» для прямого взаимодействия с Kubernetes. Если Handlers-lambda может имитировать ответ классического обработчика, то можно выполнить запрос «generate events».

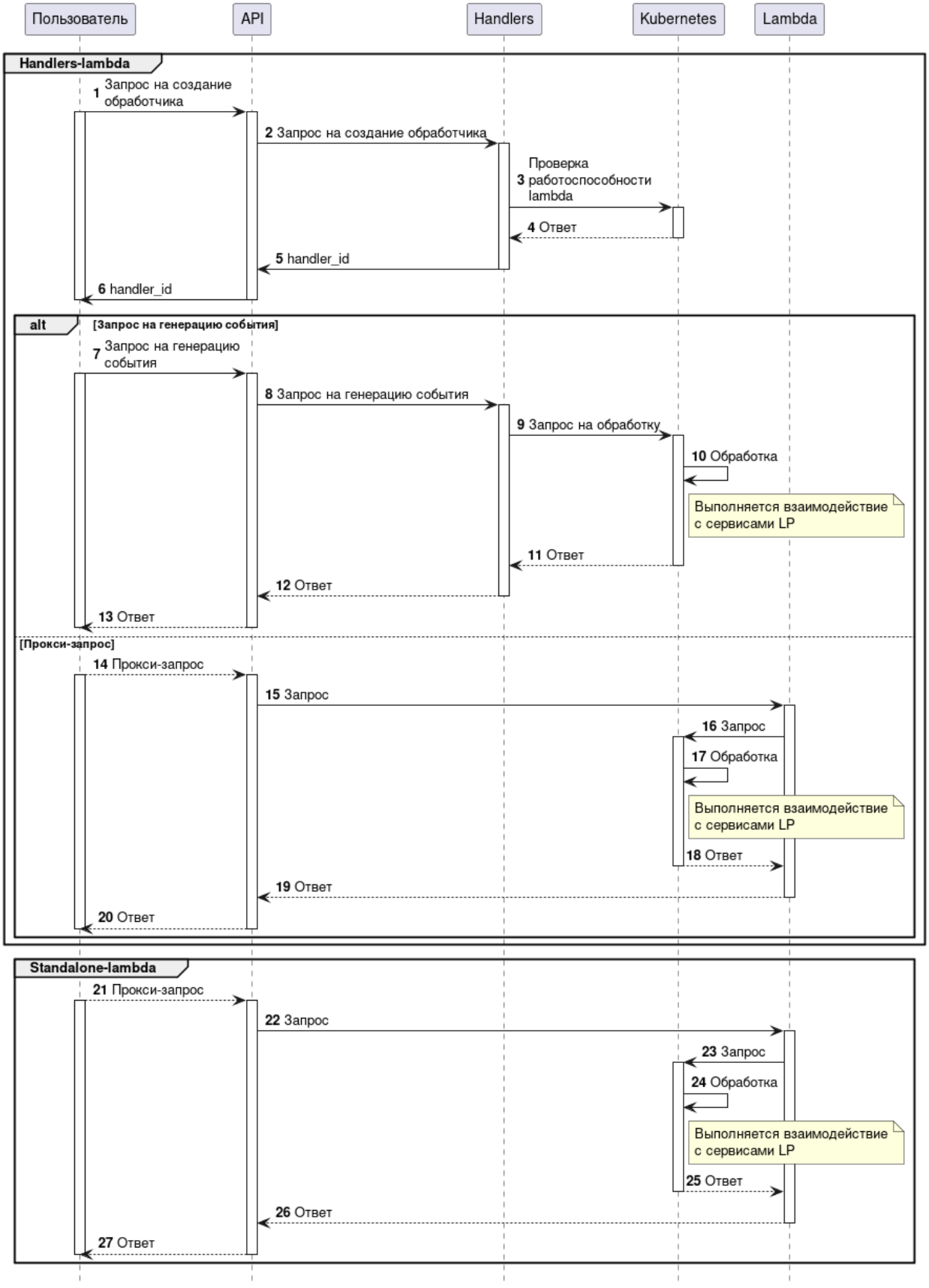


Рис. 92: Диаграмма обработки lambda

Общая схема обработки запроса:

Handlers-lambda

1. Пользователь выполняет запрос «create handler» к сервису API, указывая «handler_type» = «2» и полученный «lambda_id» (см. «Диаграмма создания lambda»).
2. Сервис API отправляет запрос в сервис Handlers.
3. Сервис Handlers выполняет проверку работоспособности (healthcheck) lambda, развернутой в Kubernetes, и создает обработчик.
4. Сервис Handlers получает ответ.
5. Сервис API получает идентификатор «handler_id».
6. Пользователь получает идентификатор «handler_id».

Запрос на генерацию события

7. Пользователь выполняет запрос «generate events» к сервису API, указывая полученный на предыдущем шаге «handler_id».
8. Сервис API отправляет запрос в сервис Handlers.
9. Сервис Handlers отправляет запрос на обработку в Kubernetes.
10. В кластере Kubernetes обрабатывается пользовательский код.

> Во время работы Handlers-lambda будет выполнено взаимодействие с некоторыми сервисами LUNA PLATFORM. См. раздел «Handlers-lambda» для дополнительной информации.

11. Сервис Handlers получает ответ.
12. Сервис API получает ответ.
13. Пользователь получает ответ.

Прокси-запрос

14. Пользователь выполняет запрос «proxy post request to lambda» к сервису API.
15. Сервис API отправляет запрос в сервис Lambda.
16. Сервис Lambda отправляет запрос на обработку в Kubernetes.
17. В кластере Kubernetes обрабатывается пользовательский код.

> Во время работы Handlers-lambda будет выполнено взаимодействие с некоторыми сервисами LUNA PLATFORM. См. раздел «Handlers-lambda» для дополнительной информации.

18. Сервис Lambda получает ответ.
19. Сервис API получает ответ.

20. Пользователь получает ответ.

Прокси-запрос для Standalone-lambda

21. Пользователь отправляет запрос «proxy post request to lambda» к сервису API.
22. Сервис API отправляет запрос в сервис Lambda.
23. Сервис Lambda отправляет запрос на обработку в Kubernetes.
24. В кластере Kubernetes обрабатывается пользовательский код.

> Во время работы Handlers-lambda будет выполнено взаимодействие с некоторыми сервисами LUNA PLATFORM. См. раздел «Standalone-lambda» для дополнительной информации.

25. Сервис Lambda получает ответ.
26. Сервис API получает ответ.
27. Пользователь получает ответ.

10 Описание баз данных

Все данные с метками времени хранятся в формате RFC 3339.

Время, используемое для хранения данных в базе данных можно установить в конфигурационном файле для каждого сервиса в параметре «STORAGE_TIME». Можно выбрать время хранения: LOCAL или UTC.

Если «STORAGE_TIME» задано как LOCAL, но время приходит в формате UTC, данные преобразуются в местное время. Если задан формат UTC, а время приходит местное, оно также преобразуется в UTC.

10.1 Описание базы данных Faces

В данном разделе приводится описание полей базы данных Faces.

См. подробную информацию в разделе «Сервис Faces».

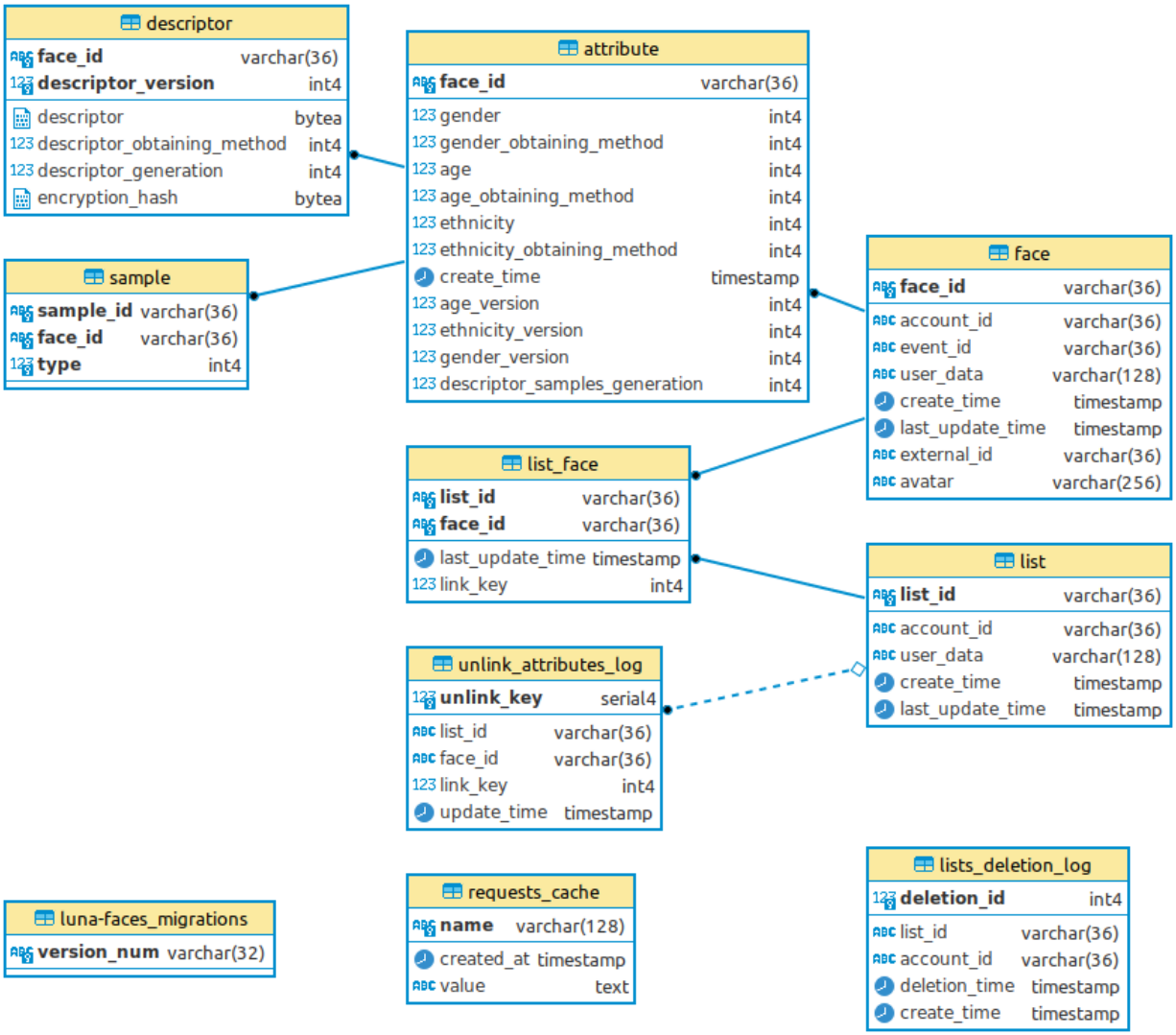


Рис. 93: Схема БД Faces

10.1.1 Модель таблицы attribute

Модель таблицы базы данных описывает атрибуты, прикреплённые к лицам.

Имя	Тип	Описание
face_id	varchar(36)	ID лица.
gender	integer	Результат оценки пола по изображению лица: • «0» — женщина • «1» — мужчина

gender_obtaining_-method	integer	Алгоритм, используемый для оценки пола по изображению лица.
gender_version	integer	Версия алгоритма оценки пола по изображению лица.
age	integer	Результат оценки возраста по изображению лица.
age_obtaining_-method	integer	Алгоритм, используемый для оценки возраста по изображению лица.
age_version	integer	Версия алгоритма оценки возраста по изображению лица.
ethnicity	integer	Результат оценки этнической принадлежности.
ethnicity_-obtaining_method	integer	Алгоритм, используемый для оценки этнической принадлежности.
ethnicity_version	integer	Версия алгоритма оценки этнической принадлежности.
create_time	timestamp	Дата и время создания атрибута.
account_id	varchar(36)	ID аккаунта, которому принадлежит атрибут.
descriptor_-samples_-generation	integer	Поколение используемых БО. Данное значение изменяется при обновлении БО атрибута. Начальное значение — 0.

10.1.2 Модель таблицы descriptor

Модель таблицы базы данных описывает биометрических шаблонов.

Имя	Тип	Описание
attribute_id	varchar(36)	ID атрибута.
descriptor_-version	integer	Версия нейросети, которая использовалась для извлечения биометрического шаблона.
descriptor	bytea	Биометрический шаблон в двоичном формате.
descriptor_-obtaining_-method	integer	Алгоритм, используемый для получения биометрического шаблона.
descriptor_-generation	integer	Поколение БШ. Данное значение изменяется при обновлении БШ атрибута. Значение показывает, что БШ не соответствует существующим биометрическим образцам. Начальное значение 0.

encryption_hash	bytea	Хэш-сумма ключа шифрования и алгоритма. См. раздел «Шифрование биометрических шаблонов».

10.1.3 Модель таблицы face

Модель таблицы базы данных описывает существующие лица.

Имя	Тип	Описание
face_id	varchar(36)	ID лица.
account_id	varchar(36)	ID аккаунта, которому принадлежит лицо.
event_id	varchar(36)	ID события. Ссылка на событие, которое создало лицо.
user_data	varchar(128)	Данные, заданные пользователем для лица.
create_-time	timestamp	Дата и время создания лица.
last_-update_-time	timestamp	Дата и время последнего обновления лица.
external_id	varchar(36)	Внешний ID. Внешний ID указывается в запросе на создание лица или в запросе на генерацию события (политика «face_policy»).
avatar	varchar(256)	URL фотоизображения, соответствующего лицу.

10.1.4 Модель таблицы list

Модель таблицы базы данных описывает существующие списки.

Имя	Тип	Описание
list_id	varchar(36)	ID списка.
account_id	varchar(36)	ID аккаунта, которому принадлежит список.
user_data	varchar(128)	Пользовательские данные для списка.
create_time	timestamp	Дата и время создания списка.
last_update_time	timestamp	Дата и время последнего обновления списка.

10.1.5 Модель таблицы list_face

Модель таблицы базы данных описывает историю прикрепления лиц к спискам. При прикреплении лица к списку появляется новая запись.

Имя	Тип	Описание
list_id	varchar(36)	ID списка.
face_id	varchar(36)	ID лица.
last_update_time	timestamp	Дата и время последнего прикрепления лица к списку.
link_key	integer	Порядковый номер прикрепления лица к списку.

10.1.6 Модель таблицы unlink_attributes_log

Модель таблицы базы данных описывает историю открепления лиц от списков. Если лицо было откреплено от списка, появляется новая запись.

Имя	Тип	Описание
unlink_key	integer	Порядковый номер открепления лица от списка.
list_id	varchar(36)	ID списка.
face_id	varchar(36)	ID лица.
link_key	integer	Порядковый номер прикрепления лица к списку.
update_time	timestamp	Дата и время последнего открепления лица от списка.

10.1.7 Модель таблицы sample

Модель таблицы базы данных описывает связи между биометрическими образцами и лицами.

Имя	Тип	Описание
sample_id	varchar(36)	Биометрический образец.
face_id	varchar(36)	ID лица, связанного с биометрическим образцом.
type	integer	Способ использования биометрического образца: • «1» — для извлечения биометрического шаблона • «5» — для создания базовых атрибутов

10.1.8 Модель таблицы list_deletion_log

Модель таблицы базы данных описывает историю удаления списков. Если список был удален, то появляется новая запись.

Имя	Тип	Описание
list_id	varchar(36)	ID списка.
account_id	varchar(36)	ID аккаунта, к которому был прикреплен список.
deletion_time	timestamp	Дата и время удаления списка.
create_time	timestamp	Время создания списка.
deletion_id	integer	ID удаления.

10.1.9 Модель таблицы requests_cache

Модель таблицы базы данных описывает кеш максимального количества лиц с привязанными БШ или базовыми атрибутами для запросов лицензии.

Имя	Тип	Описание
created_at	timestamp	Дата и время кеширования данных.
value	text	Шифрованный кеш значения.
name	varchar(36)	Уникальное имя кеша.

10.1.10 Модель таблицы luna-faces_migrations

Имя	Тип	Описание
version_num	varchar(32)	Параметр, необходимый для миграции БД.

10.2 Описание базы данных Events

В данном разделе приводится описание полей базы данных Events.

См. подробную информацию в разделе «Сервис Events».

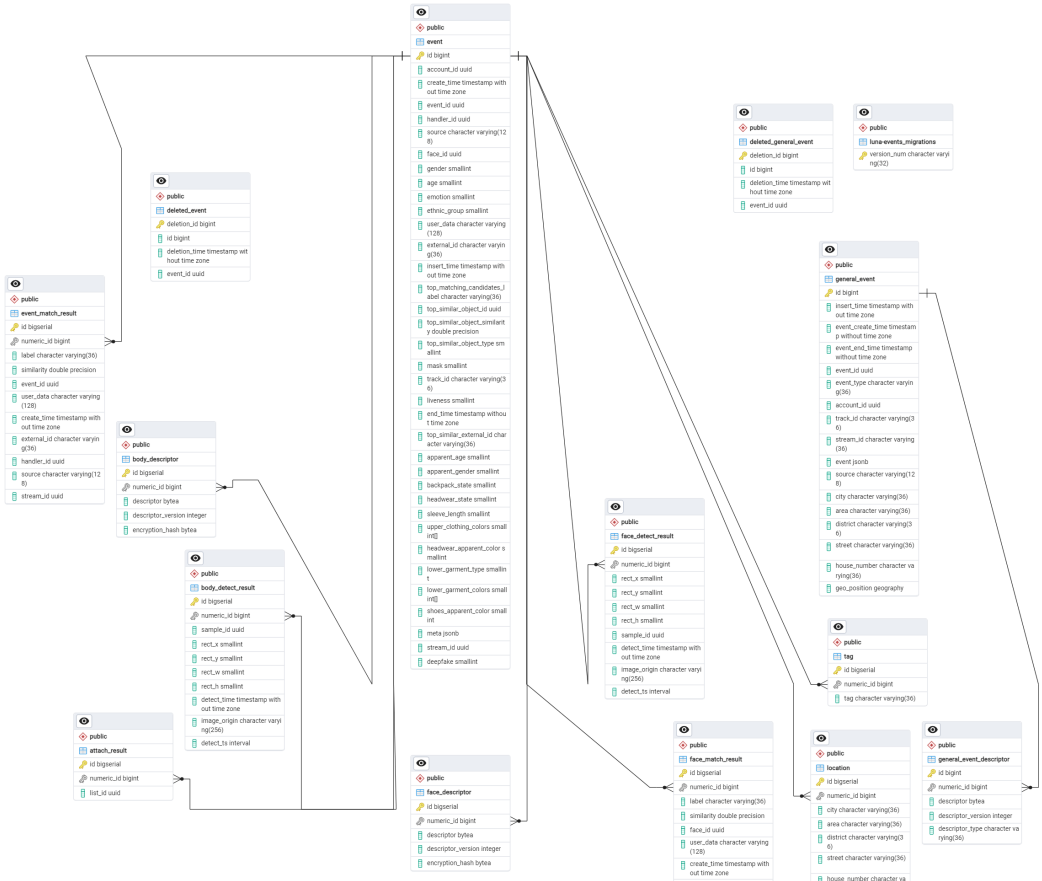


Рис. 94: Схема БД Events

10.2.1 Модель таблицы event

Модель таблицы базы данных описывает существующие события. Она включает в себя информацию о созданных событиях и лицах.

Имя	Тип	Описание
id	bigint	Первичный ключ таблицы (цифровой).
account_id	uuid	ID аккаунта, которому принадлежит событие.
create_time	timestamp	Временной код возникновения события в видеопотоке. Этот параметр используется для мониторинга создания событий в реальном времени.

event_id	uuid	ID события.
handler_id	uuid	ID обработчика, который создал событие.
source	varchar(128)	Источник события. Источник указывается в запросе на генерацию события.
face_id	uuid	ID лица, соответствующего событию.
gender	smallint	Результат оценки пола по изображению лица.
age	smallint	Результат оценки возраста по изображению лица.
emotion	smallint	Результат оценки эмоций.
ethnic_group	smallint	Результат оценки этнической группы.
user_data	varchar(128)	Пользовательские данные для лица, соответствующего событию. Пользовательские данные можно указать в запросе на генерацию события.
external_id	varchar(36)	Внешний ID лица, соответствующего событию. Идентификатор можно указать в запросе на генерацию события.
insert_time	timestamp	Дата и время создания события.
top_matching_-candidates_-label	varchar(36)	Метка группы кандидатов, используемых для сравнения.
top_similar_-object_id	uuid	ID наиболее похожего объекта из результатов сравнения. Результаты сравнения получают, когда активирована политика «match_policy» обработчика.
top_similar_-object_-similarity	double precision	Степень схожести наиболее похожего объекта из результатов сравнения. Результаты сравнения получают, когда активирована политика «match_policy» обработчика.
top_similar_-object_type	smallint	Тип наиболее похожего объекта: • «0» — лицо • «1» — событие
mask	smallint	Результат оценки наличия медицинской маски: • «1» — отсутствует • «2» — медицинская маска • «3» — рот перекрыт

track_id	varchar(36)	ID трека. Идентификатор можно указать в запросе на генерацию события.
liveness	smallint	Результат проверки Liveness: • «0» — spoof (человек не является реальным) • «1» — real (человек является реальным) • «2» — неизвестно
end_time	timestamp	Временной код окончания события в видеопотоке. Этот параметр используется для мониторинга создания событий в реальном времени. Задаётся равным «create_time», если не указан явно.
top_similar_-external_id	varchar(36)	External ID наиболее похожего объекта из результатов сравнения . Результаты сравнения получают, когда активирована политика «match_policy» обработчика.
apparent_age	smallint	Результат оценки возраста по изображению тела.
apparent_-gender	smallint	Результат оценки пола по изображению тела: • «0» — женский • «1» — мужской • «2» — неизвестно
backpack_state	smallint	Результат оценки состояния рюкзака: • «0» — отсутствует • «1» — присутствует • «2» — неизвестно
headwear_state	smallint	Результат оценки состояния головного убора: • «0» — отсутствует • «1» — присутствует • «2» — неизвестно
sleeve_length	smallint	Результат оценки длины рукавов: • «0» — короткие • «1» — длинные • «2» — неизвестно

upper_-clothing_color	array[int]	Результат оценки цветов верхней одежды: • «0» — неизвестно • «1» — чёрный • «2» — синий • «3» — зеленый • «4» — серый • «5» — оранжевый • «6» — фиолетовый • «7» — красный • «8» — белый • «9» — желтый • «10» — розовый • «11» — коричневый • «12» — бежевый • «13» — хаки • «14» — разноцветный
headwear_-apparent_color	smallint	Результат оценки цвета головного убора: • «0» — неизвестно • «1» — белый • «2» — черный • «3» — прочий
lower_-garment_type	smallint	Результат оценки типа нижней одежды: • «0» — неизвестно • «1» — брюки • «2» — шорты • «3» — юбка

lower_-garment_colors	array[int]	Результат оценки цветов нижней одежды: • «0» — неизвестно • «1» — чёрный • «2» — синий • «3» — зеленый • «4» — серый • «5» — оранжевый • «6» — фиолетовый • «7» — красный • «8» — белый • «9» — желтый • «10» — розовый • «11» — коричневый • «12» — бежевый • «13» — хаки • «14» — разноцветный
shoes_-apparent_color	smallint	Результат оценки цвета обуви: • «0» — неизвестно • «1» — белый • «2» — черный • «3» — прочий
meta	jsonb	Пользовательская метаинформация.
stream_id	uuid	ID потока, создаваемого в результате запроса «create stream»
deepfake	smallint	Результат оценки Deepfake: • «0» — spoof (человек не является реальным) • «1» — real (человек является реальным)

10.2.2 Модель таблицы general_event

Имя	Тип	Описание
id	bigint	Первичный ключ таблицы (цифровой).
insert_time	timestamp	Дата и время создания общего события.

Имя	Тип	Описание
event_create_time	timestamp	Временной код возникновения общего события в видеопотоке. Этот параметр используется для мониторинга создания общих событий в реальном времени.
event_end_time	timestamp	Временной код окончания общего события в видеопотоке. Этот параметр используется для мониторинга создания общих событий в реальном времени. Задаётся равным «create_time», если не указан явно.
event_id	uuid	ID общего события.
event_type	varchar(36)	Тип общего события.
account_id	uuid	ID аккаунта, которому принадлежит обобщенное событие.
track_id	varchar(36)	ID трека. Идентификатор можно указать в запросе на генерацию общего события.
stream_id	varchar(36)	ID потока.
event	jsonb	Содержимое общего события.
sourse	varchar(128)	Источник общего события. Источник указывается в запросе на генерацию общего события.

10.2.3 Модель таблицы general_event_location

Имя	Тип	Описание
id	bigint	Первичный ключ таблицы (цифровой).
numeric_id	bigint	Внешний ключ общего события.
city	varchar(36)	Город для общего события.
area	varchar(36)	Область для общего события.
district	varchar(36)	Район для общего события.
street	varchar(36)	Улица для общего события.
house_-number	varchar(36)	Номер дома для общего события.

Имя	Тип	Описание
geo_position	geography	Географические координаты (широта, долгота) для общего события.

10.2.4 Модель таблицы deleted_event

Модель таблицы базы данных описывает удаленные события.

Имя	Тип	Описание
deletion_id	bigint	Первичный ключ таблицы (цифровой).
id	bigint	Внешний ключ таблицы.
deletion_time	timestamp without time zone	Время удаления события.
event_id	uuid	Идентификатор удаленного события.

10.2.5 Модель таблицы deleted_general_event

Модель таблицы базы данных описывает удаленные общие события.

Имя	Тип	Описание
deletion_id	bigint	Первичный ключ таблицы (цифровой).
id	bigint	Внешний ключ таблицы.
deletion_time	timestamp without time zone	Время удаления события.
event_id	uuid	Идентификатор удаленного события.

10.2.6 Модель таблицы face_detect_result

Модель таблицы базы данных описывает обнаружение лиц.

Имя	Тип	Описание
id	bigint	Первичный ключ таблицы (цифровой).
numeric_-id	bigint	Внешний ключ события.

Имя	Тип	Описание
rect_x	smallint	Координата верхнего левого угла ограничивающего прямоугольника лица по оси «X».
rect_y	smallint	Координата верхнего левого угла ограничивающего прямоугольника лица по оси «Y».
rect_w	smallint	Ширина ограничивающего прямоугольника.
rect_h	smallint	Высота ограничивающего прямоугольника.
sample_id	uuid	ID биометрического образца.
detect_-time	timestamp	Время детекции лица.
image_-origin	varchar(256)	URL исходного изображения с лицом.
detect_ts	interval	Время относительно чего-либо, например, относительно начала видеофайла.

10.2.7 Модель таблицы body_detect_result

Модель таблицы базы данных описывает обнаружение тел.

Имя	Тип	Описание
id	bigint	Первичный ключ таблицы (цифровой).
numeric_-id	bigint	Внешний ключ события.
rect_x	smallint	Координата верхнего левого угла ограничивающего прямоугольника тела по оси «X».
rect_y	smallint	Координата верхнего левого угла ограничивающего прямоугольника тела по оси «Y».
rect_w	smallint	Ширина ограничивающего прямоугольника.
rect_h	smallint	Высота ограничивающего прямоугольника.
sample_id	uuid	ID биометрического образца.
detect_-time	timestamp	Время детекции тела.
image_-origin	varchar(256)	URL исходного изображения с телом.

Имя	Тип	Описание
detect_ts	interval	Время относительно чего-либо, например, относительно начала видеофайла.

10.2.8 Модель таблицы face_descriptor

Модель таблицы базы данных описывает биометрические шаблоны лиц, хранящиеся в базе данных.

Имя	Тип	Описание
id	bigint	Первичный ключ таблицы (цифровой).
numeric_id	bigint	Внешний ключ события.
descriptor	bytea	Двоичный БШ лица.
descriptor_version	integer	Версия нейросети, используемая для извлечения БШ.
encryption_hash	bytea	Хэш-сумма ключа шифрования и алгоритма. См. раздел «Шифрование биометрических шаблонов».

10.2.9 Модель таблицы body_descriptor

Модель таблицы базы данных описывает биометрические шаблоны тел, хранящиеся в базе данных.

Имя	Тип	Описание
id	bigint	Первичный ключ таблицы (цифровой).
numeric_id	bigint	Внешний ключ события.
descriptor	bytea	Двоичный БШ тела.
descriptor_version	integer	Версия нейросети, используемая для извлечения БШ.
encryption_hash	bytea	Хэш-сумма ключа шифрования и алгоритма. См. раздел «Шифрование биометрических шаблонов».

10.2.10 Модель таблицы event_match_result

Модель таблицы базы данных описывает результаты сравнения, полученные посредством политики «match_policy» обработчика. Каждая запись включает в себя информацию о событии, ис-

пользуемом для сравнения, и степень схожести.

Имя	Тип	Описание
id	bigint	Первичный ключ таблицы (цифровой).
numeric_-id	bigint	Внешний ключ события.
label	varchar(36)	Метка, указываемая для результатов сравнения.
similarity	double precision	Степень схожести, полученная после сравнения БШ события с заданным БШ.
event_id	uuid	ID события.
user_data	varchar(128)	Пользовательские данные, связанные с событием.
create_-time	timestamp	Время создания события.
external_-id	varchar(36)	Внешний ID лица, созданного во время создания события.
handler_-id	uuid	ID, используемый для создания события.
source	varchar(128)	Источник события.
stream_id	uuid	ID потока, создаваемого в результате запроса «create stream»

10.2.11 Модель таблицы face_match_result

Модель таблицы базы данных описывает результаты сравнения, полученные посредством политики «match_policy» обработчика. Каждая запись включает в себя информацию о лице, используемом для сравнения, и степень схожести.

Имя	Тип	Описание
id	bigint	Первичный ключ таблицы (цифровой).
numeric_-id	bigint	Внешний ключ события.
label	varchar(36)	Метка, указываемая для результатов сравнения.
similarity	double precision	Степень схожести, полученная после сравнения БШ события с заданным БШ лица.
face_id	uuid	ID лица.

Имя	Тип	Описание
user_data	varchar(128)	Пользовательские данные, связанные с лицом.
create_-time	timestamp	Время создания лица.
external_-id	varchar(36)	Внешний ID лица.

10.2.12 Модель таблицы location

Имя	Тип	Описание
id	bigint	Первичный ключ таблицы (цифровой).
numeric_id	bigint	Внешний ключ события.
city	varchar(36)	Город.
area	varchar(36)	Область.
district	varchar(36)	Район.
street	varchar(36)	Улица.
house_number	varchar(36)	Номер дома.
geo_position	geography	Географические координаты (широта, долгота).

10.2.13 Модель таблицы tag

Модель таблицы базы данных описывает теги для событий. Теги указываются в запросе на создание событий.

Имя	Тип	Описание
id	bigint	Первичный ключ таблицы (цифровой).
numeric_id	bigint	Внешний ключ события.
tag	varchar(36)	Тег события.

10.2.14 Модель таблицы attach_result

Модель таблицы базы данных описывает прикрепление лица, созданного из события, к списку. Лицо создается с помощью политики «face_policy». Лицо прикрепляется к списку с помощью политики «link_to_lists_policy».

Имя	Тип	Описание
id	bigint	Первичный ключ таблицы (цифровой).
numeric_id	bigint	Внешний ключ события.
list_id	uuid	Список, к которому прикреплено созданное лицо.

10.3 Описание базы данных Tasks

В данном разделе приводится описание полей базы данных Tasks.

См. подробную информацию в разделе «Сервис Tasks».

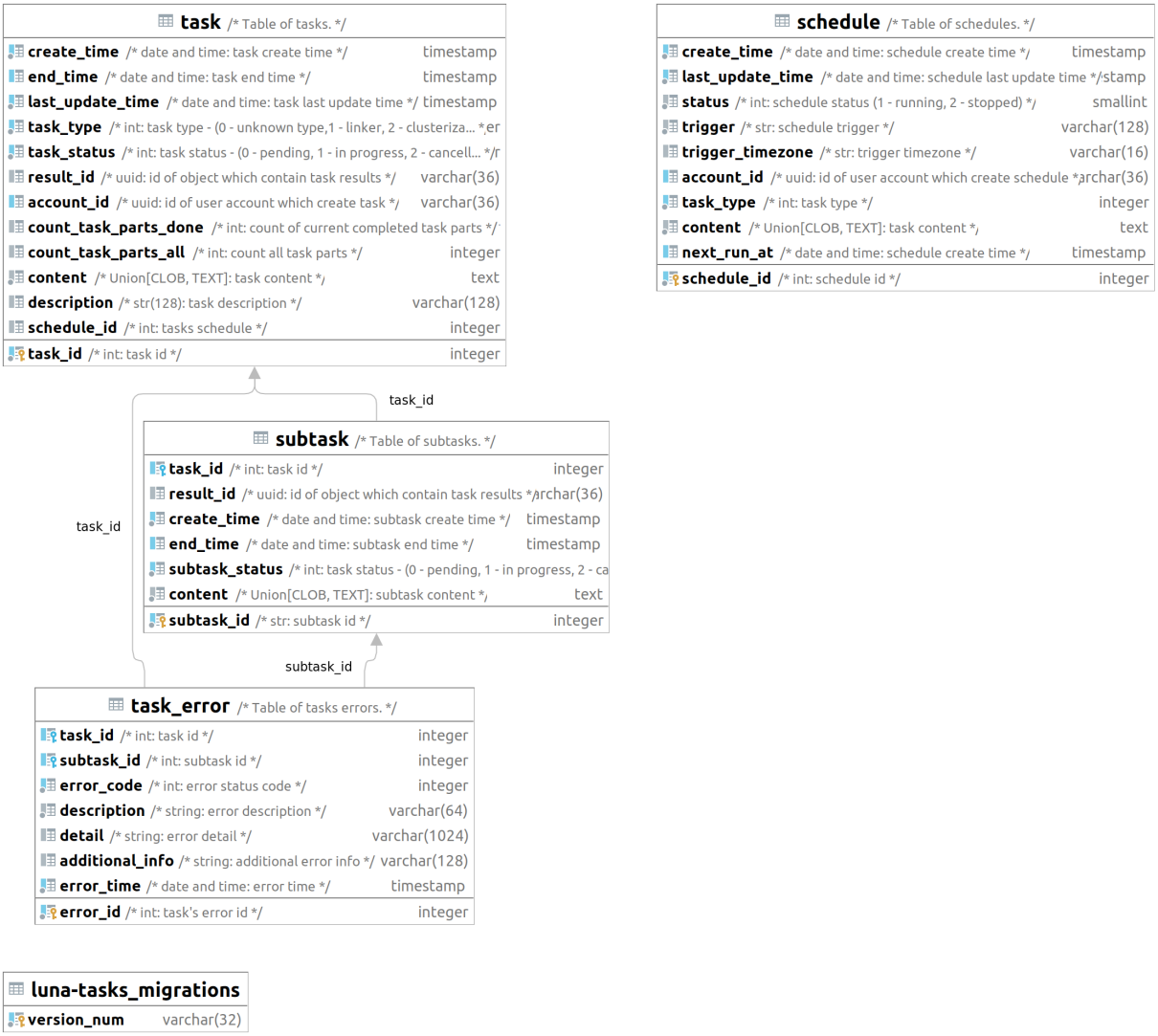


Рис. 95: Схема БД Tasks

10.3.1 Модель таблицы task

Модель таблицы базы данных описывает созданные задачи. Она включает в себя основную информацию о задаче и ее содержимом.

Имя	Тип	Описание

task_id	integer	ID задачи.
create_time	timestamp	Время создания задачи.
end_time	timestamp	Время завершения задачи.
last_update_time	timestamp	Время последнего обновления задачи.
task_type	integer	Тип задачи: • «0» — неизвестная • «1» — Linker • «2» — Clustering • «3» — Reporter • «4» — Garbage collection • «5» — Additional extraction • «6» — Cross-matching • «7» — ROC-curve calculating • «8» — Exporter • «9» — Estimator
task_status	integer	Статус задачи: • «0» — в ожидании • «1» — обрабатывается • «2» — отменена • «3» — сбой • «4» — сбор результатов • «5» — выполнена
result_id	varchar(36)	ID результата задачи.
account_id	varchar(36)	ID аккаунта, которому принадлежит задача.
count_task_parts_-done	integer	Количество завершенных подзадач.
count_task_parts_all	integer	Общее количество подзадач.
content	text	Фильтры и параметры запросов для обработки задачи.
description	varchar(128)	Заданное пользователем описание задачи.
schedule_id	integer	ID расписания.

10.3.2 Модель таблицы subtask

Модель таблицы базы данных включает в себя информацию о созданных подзадачах. В зависимости от типа задачи подзадач может быть одна или несколько.

Имя	Тип	Описание
subtask_id	integer	ID подзадачи.
task_id	integer	ID соответствующей задачи.
result_id	varchar(36)	ID результата подзадачи.
create_time	timestamp	Время создания подзадачи.
end_time	timestamp	Время завершения подзадачи.
subtask_status	integer	Статус подзадачи: • «0» — в ожидании • «1» — обрабатывается • «2» — отменена • «3» — сбой • «4» — сбор результатов • «5» — выполнена
content	text	Фильтры и параметры запросов для обработки задачи.

10.3.3 Модель таблицы task_error

Модель таблицы базы данных включает в себя информацию об ошибках, произошедших в процессе обработки задачи. Ошибки добавляются в таблицу рабочими процессами Tasks.

Имя	Тип	Описание
error_id	integer	ID ошибки задачи.
task_id	integer	ID соответствующей задачи.
error_code	integer	Код ошибки.
description	varchar(64)	Краткое описание ошибки.
detail	varchar(102	Подробное описание ошибки.
additional_info	varchar(128)	Дополнительная информация об ошибке. Может включать в себя ID потерянных объектов и любую другую полезную информацию.

Имя	Тип	Описание
error_-time	timestamp	Время возникновения ошибки.

10.3.4 Модель таблицы schedule

Модель таблицы базы данных включает в себя информацию о расписании выполнения задач.

Имя	Тип	Описание
schedule_id	integer	ID расписания.
create_time	timestamp	Дата создания расписания.
last_update_-time	timestamp	Дата и время последних изменений расписания.
status	smallint	Статус расписания: • «1» — запущено • «2» — остановлено
trigger	varchar(128)	Cron-выражение.
trigger_-timezone	varchar(16)	Временная зона (UTC или LOCAL).
account_id	varchar(36)	ID аккаунта, которому принадлежит расписание.
task_type	integer	Тип задачи, которая выполняется по расписанию.
content	text	Содержимое задачи которая выполняется по расписанию (фильтры, список и пр.).
next_run_at	timestamp	Время следующего запуска задачи.

10.3.5 Модель таблицы luna-tasks_migrations

Имя	Тип	Описание
version_num	varchar(32)	Параметр, необходимый для миграции БД.

10.4 Описание базы данных Handlers

В данном разделе приводится описание полей базы данных Handlers.

См. подробную информацию о сервисе Handlers в разделе «Сервис Handlers».

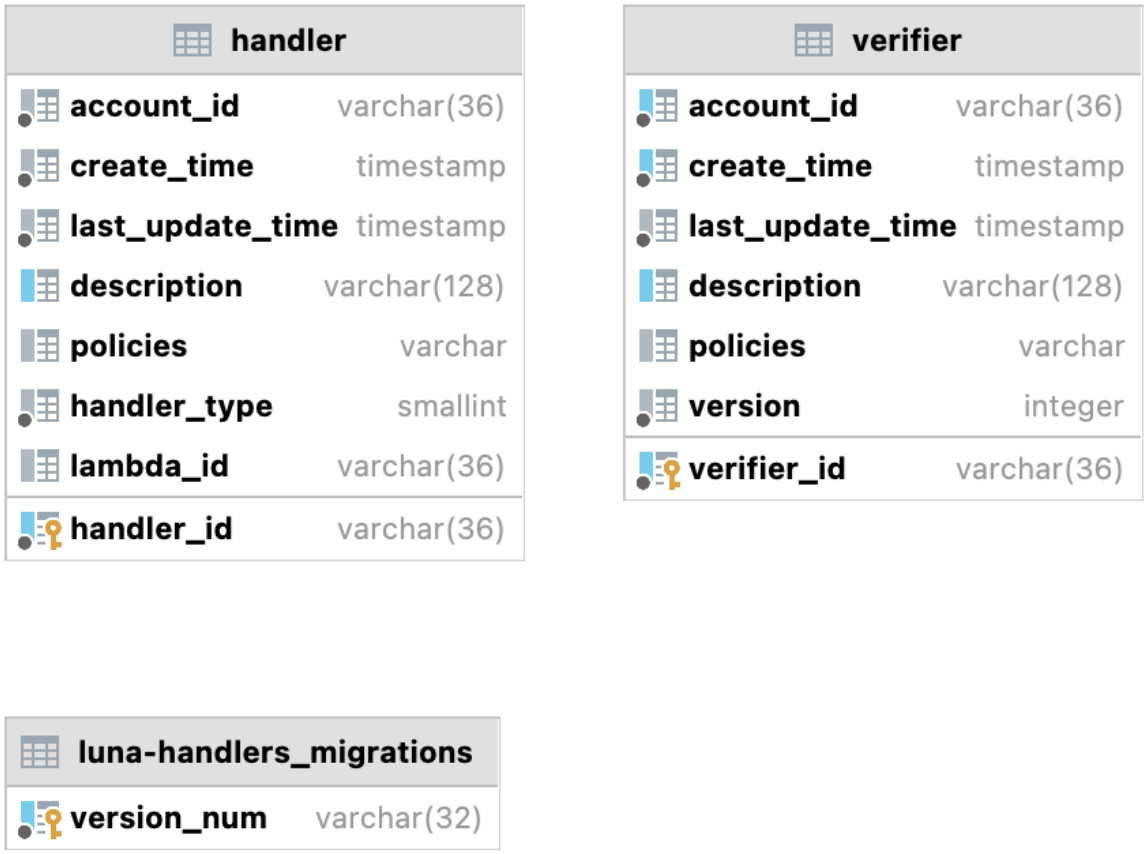


Рис. 96: Схема БД Handlers

10.4.1 Модель таблицы handler

Имя	Тип	Описание
handler_id	varchar(36)	ID обработчика в формате UUID4 («xxxxxxxx-xxxx-4xxx-{8-9}xx-xxxxxxxxxxxx»).

account_id	varchar(36)	ID аккаунта в формате UUID4, которому принадлежит обработчик.
create_time	timestamp	Дата и время создания обработчика.
last_update_-time	timestamp	Дата и время последних изменений обработчика.
description	varchar(128)	Описание обработчика, заданное пользователем.
handler_type	smallint	Тип обработчика: • «0» — статический • «1» — динамический • «2» — lambda
lambda_id	smallint	Lambda ID.
policies	varchar(36)	Политики обработчика.

10.4.2 Модель таблицы verifier

Имя	Тип	Описание
verifier_id	varchar(36)	ID верификатора в формате UUID4 в формате «xxxxxxxx-xxxx-4xxx-{8-9}xx-xxxxxxxxxxxx».
account_id	varchar	ID аккаунта в формате UUID4, которому принадлежит верификатор.
create_time	timestamp	Дата и время создания верификатора.
description	varchar	Описание верификатора, заданное пользователем.
last_-update_-time	timestamp	Дата и время последних изменений.
policies	varchar(2048)	Политики верификатора.
version	integer	Версия верификатора.

10.4.3 Модель таблицы luna-handlers_migrations

Имя	Тип	Описание
version_num	varchar(32)	Параметр, необходимый для миграции БД.

10.5 Описание базы данных Configurator

В данном разделе приводится описание полей базы данных Configurator.

См. подробную информацию в разделе «Сервис Configurator».

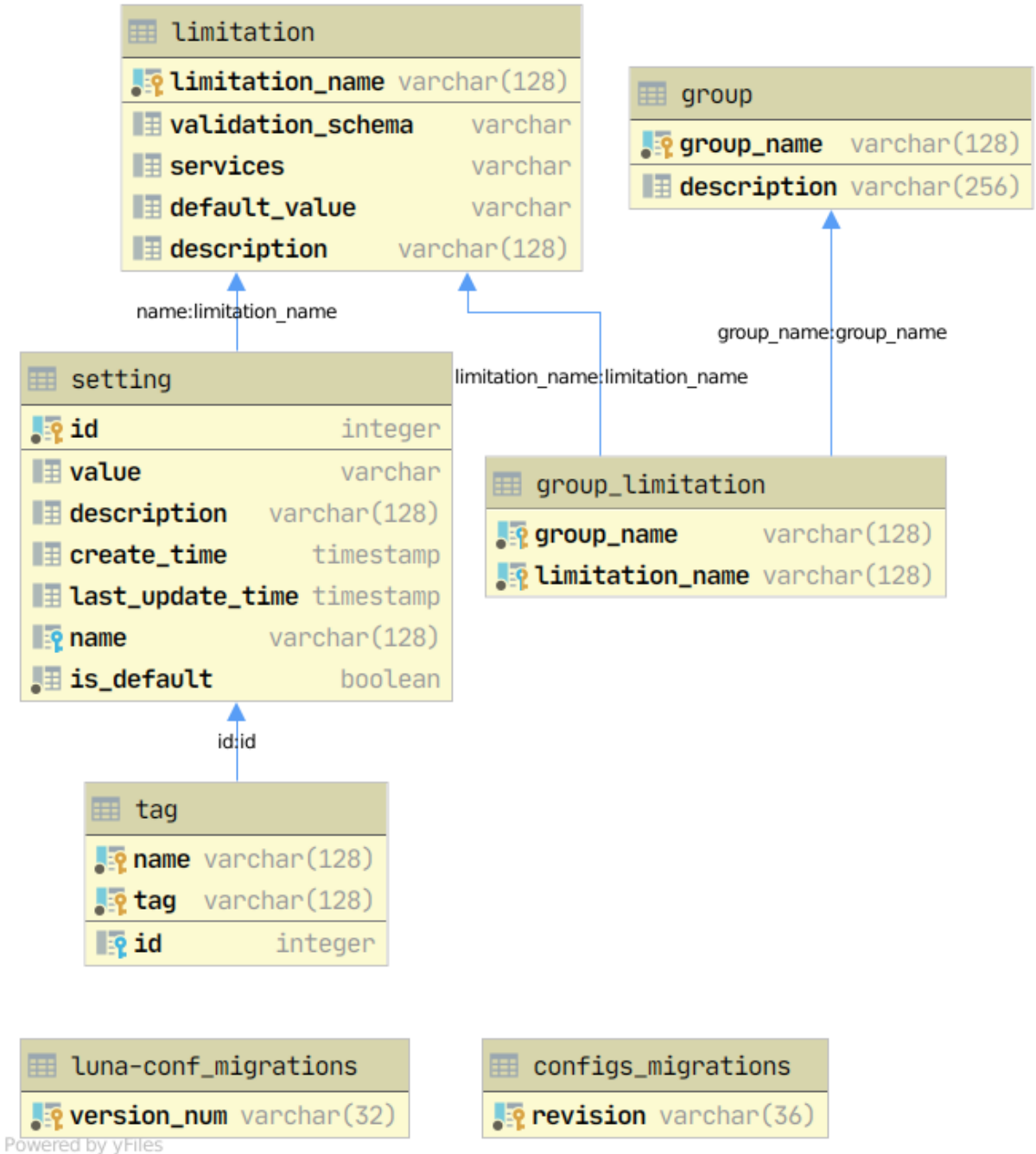


Рис. 97: Схема БД Configurator

10.5.1 Модель таблицы limitation

Имя	Тип	Описание
limitation_name	varchar(128)	Название ограничения.
validation_schema	varchar	Схема валидации ограничения.
services	varchar	Список сервисов.
default_value	varchar	Значение ограничения по умолчанию.
description	varchar(128)	Описание ограничения.

10.5.2 Модель таблицы setting

Имя	Тип	Описание
id	integer	ID настройки.
value	varchar	Значение настройки.
description	varchar(128)	Описание настройки.
create_time	timestamp	Время создания настройки.
last_update_time	timestamp	Время последнего изменения настройки.
name	varchar(128)	Название настройки.
is_default	boolean	Является ли эта настройка настройкой по умолчанию.

10.5.3 Модель таблицы tag

Имя	Тип	Описание
id	integer	ID настройки.
name	varchar(128)	Название настройки.
tag	varchar(128)	Строка тега настройки.

10.5.4 Модель таблицы group

Имя	Тип	Описание
group_name	varchar(128)	Имя группы.

Имя	Тип	Описание
description	varchar(256)	Описание группы.

10.5.5 Модель таблицы group_limitation

Имя	Тип	Описание
group_name	varchar(128)	Имя группы.
limitation_name	varchar(128)	Имя ограничения.

10.5.6 Модель таблицы configs_migration

Имя	Тип	Описание
revision	varchar(36)	Ревизия миграции настроек.

10.5.7 Модель таблицы luna-conf_migrations

Имя	Тип	Описание
version_num	varchar(32)	Параметр, необходимый для миграции БД.

10.6 Описание базы данных Backport3

В данном разделе приводится описание полей базы данных Backport 3.

См. подробную информацию в разделе «Сервис Backport 3».

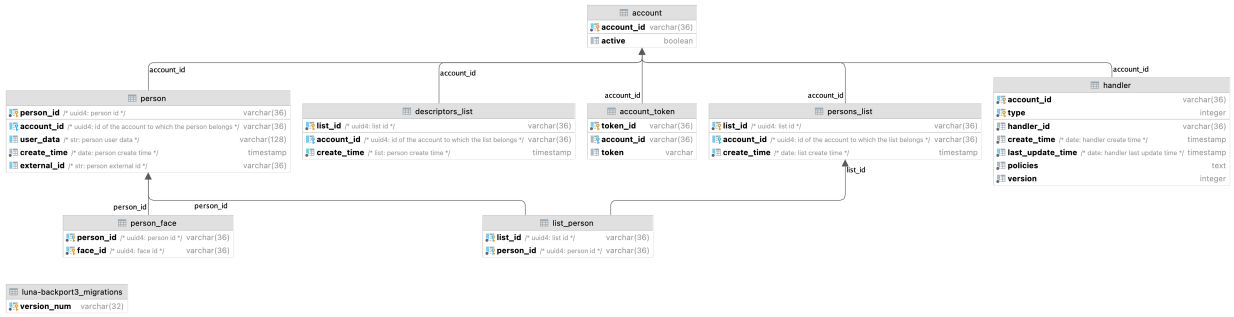


Рис. 98: Схема БД Backport3

10.6.1 Модель таблицы account

Имя	Тип	Описание
account_id	varchar(36)	ID аккаунта в формате UUID4.
active	boolean	Статус аккаунта.

10.6.2 Модель таблицы account_token

Имя	Тип	Описание
token_id	varchar(36)	ID токена.
account_id	varchar(36)	ID аккаунта, к которому привязан токен.
token	varchar(128)	Строка с данными токена.

10.6.3 Модель таблицы person

Имя	Тип	Описание
person_id	varchar(36)	ID персоны в формате UUID4 в формате «xxxxxxxx-xxxx-4xxx-{8-9}xx-xxxxxxxxxxxx».

Имя	Тип	Описание
account_-id	varchar(36)	ID аккаунта в формате UUID4, которому принадлежит персона.
user_data	varchar(128)	Пользовательские данные персоны.
create_-time	timestamp	Дата и время создания персоны.
external_-id	varchar(36)	ID персоны во внешней системе.

10.6.4 Модель таблицы persons_list

Имя	Тип	Описание
list_id	varchar(36)	ID списка в формате UUID4 в формате «xxxxxxxx-xxxx-4xxx-{8-9}xx-xxxxxxxxxxxx».
account_-id	varchar(36)	ID аккаунта в формате UUID4 в формате «xxxxxxxx-xxxx-4xxx-{8-9}xx-xxxxxxxxxxxx».
create_-time	timestamp	Дата и время создания списка.

10.6.5 Модель таблицы descriptors_list

Имя	Тип	Описание
list_id	varchar(36)	ID списка в формате UUID4 в формате «xxxxxxxx-xxxx-4xxx-{8-9}xx-xxxxxxxxxxxx».
account_-id	varchar(36)	ID аккаунта, которому принадлежит список, в формате UUID4 в формате «xxxxxxxx-xxxx-4xxx-{8-9}xx-xxxxxxxxxxxx».
create_-time	timestamp	Дата и время создания списка.

10.6.6 Модель таблицы list_person

Модель таблицы базы данных для связей между персонами и списками.

Имя	Тип	Описание
list_id	varchar(36)	ID списка в формате UUID4 в формате «xxxxxxxx-xxxx-4xxx-{8-9}xx-xxxxxxxxxxxx».
person_-id	varchar(36)	ID персоны в формате UUID4 в формате «xxxxxxxx-xxxx-4xxx-{8-9}xx-xxxxxxxxxxxx».

10.6.7 Модель таблицы person_face

Модель таблицы базы данных для связей между персонами и лицами.

Имя	Тип	Описание
person_-id	varchar(36)	ID персоны в формате UUID4 в формате «xxxxxxxx-xxxx-4xxx-{8-9}xx-xxxxxxxxxxxx».
face_id	varchar(36)	ID лица в формате UUID4 в формате «xxxxxxxx-xxxx-4xxx-{8-9}xx-xxxxxxxxxxxx».

10.6.8 Модель таблицы luna-backport3_migrations

Имя	Тип	Описание
version_num	varchar(32)	Параметр, необходимый для миграции БД

10.6.9 Модель таблицы handler

Имя	Тип	Описание
account_-id	varchar(36)	ID аккаунта, которому принадлежит обработчик, в формате UUID4 в формате «xxxxxxxx-xxxx-4xxx-{8-9}xx-xxxxxxxxxxxx».
type	integer	Тип обработчика.
handler_-id	varchar(36)	ID обработчика в формате UUID4 в формате «xxxxxxxx-xxxx-4xxx-{8-9}xx-xxxxxxxxxxxx».
create_-time	timestamp	Дата и время создания обработчика.

Имя	Тип	Описание
last_-update_-time	timestamp	Дата и время последнего изменения обработчика.
policies	varchar(2048)	Политики обработчика.
version	integer	Версия обработчика.

10.7 Описание базы данных Accounts

В данном разделе приводится описание полей базы данных Accounts.

См. подробную информацию в разделе «Сервис Accounts».

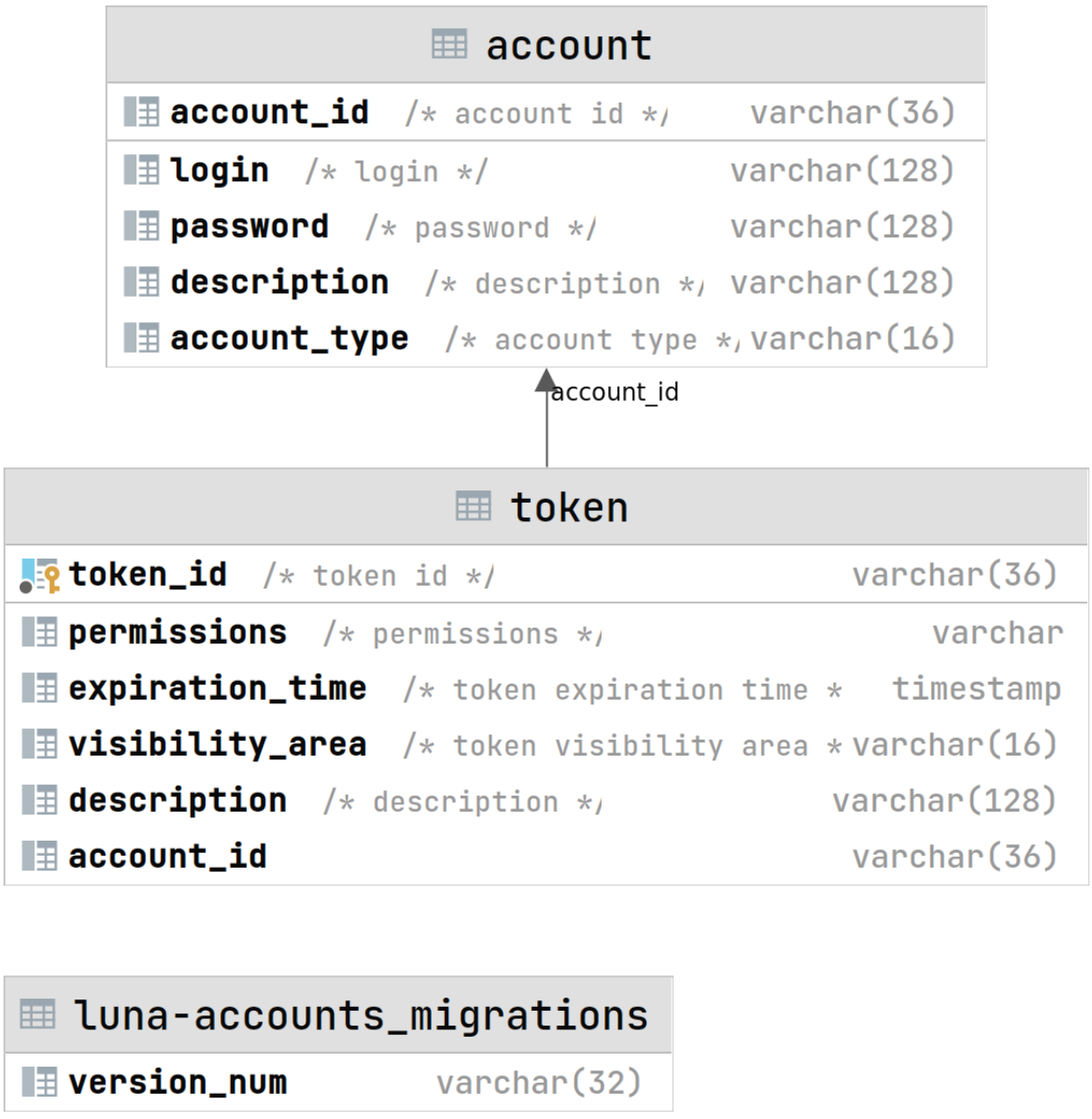


Рис. 99: Структура БД Accounts

10.7.1 Модель таблицы account

Имя	Тип	Описание
account_id	varchar(36)	ID аккаунта в формате UUID4.
login	varchar(128)	Логин.
password	varchar(128)	Пароль.

description	varchar(128)	Описание аккаунта.
account_type	varchar(16)	Тип аккаунта: • «user» • «advanced_user» • «admin»
create_time	timestamp	Дата и время создания аккаунта.
last_update_time	timestamp	Дата и время последнего изменения аккаунта.

10.7.2 Модель таблицы token

Имя	Тип	Описание
token_id	varchar(36)	ID токена в формате UUID4.
permissions	varchar(128)	Разрешения токена.
expiration_time	varchar(128)	Время действия токена.
description	varchar(128)	Описание аккаунта.
visibility_area	varchar(128)	Видимость объектов токеном: • «all» — все объекты • «account» — только объекты своего аккаунта
account_id	varchar(16)	ID аккаунта в формате UUID4, к которому привязан токен.

10.7.3 Модель таблицы luna-accounts_migration

Имя	Тип	Описание
version_num	varchar(32)	Параметр, необходимый для миграции БД.

10.8 Описание базы данных Lambda

В данном разделе приводится описание полей базы данных Lambda.

См. подробную информацию в разделе «Сервис Lambda».

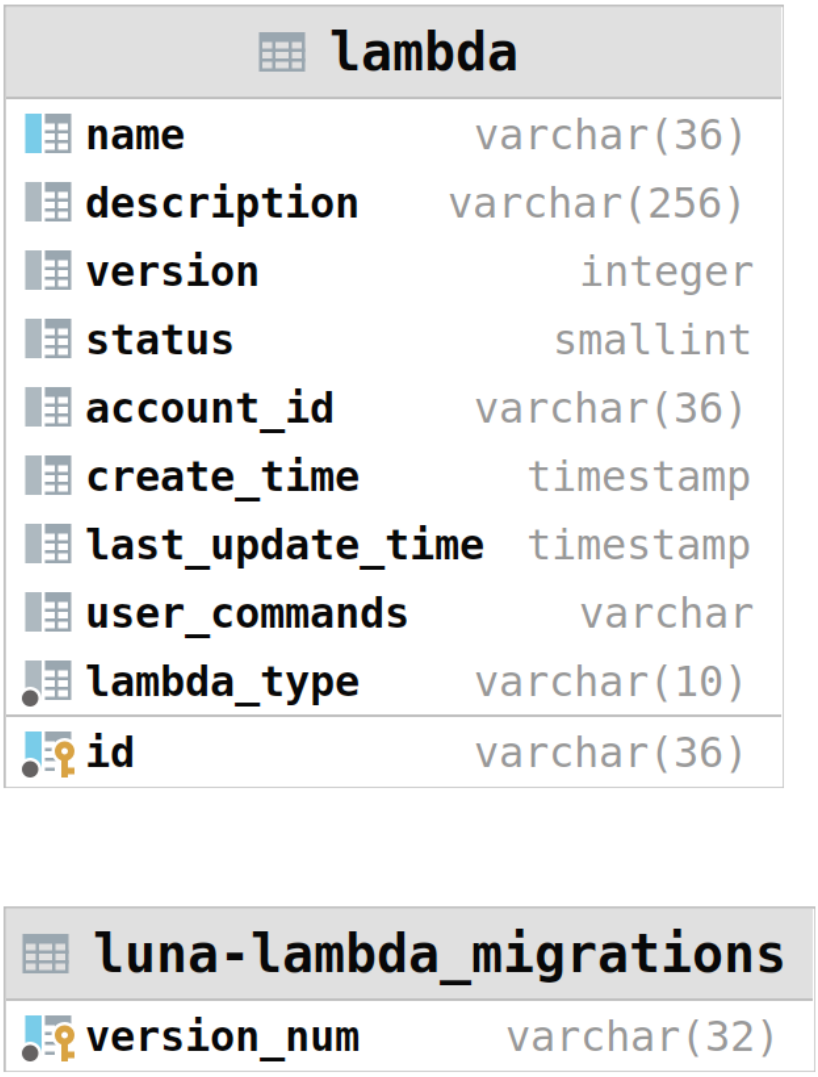


Рис. 100: Структура БД Lambda

10.8.1 Модель таблицы lambda

Имя	Тип	Описание
name	varchar(36)	Имя lambda.
description	varchar(256)	Описание lambda.
version	integer	Версия lambda.

account_id	varchar(16)	ID аккаунта в формате UUID4, к которому привязана lambda.
status	smallint	Статус создания lambda: • «running» — запущена • «waiting» — ожидает запуска • «terminated» — остановлена • «not_found» — не найдена
create_time	timestamp	Время создания lambda.
last_update_-time	timestamp	Дата и время последнего изменения lambda.
user_commands	varchar	Список дополнительных команд Docker для создания lambda-контейнера.
lambda_type	varchar(10)	Тип lambda: • «handlers» • «standalone»
id	varchar(36)	ID lambda.

10.8.2 Модель таблицы luna-lambda_migration

Имя	Тип	Описание
version_num	varchar(32)	Параметр, необходимый для миграции БД.

10.9 Описание базы данных Video Manager

В данном разделе приводится описание полей базы данных Video Manager.

См. подробную информацию в разделе «Сервисы видеоаналитики».

Рис. 101: Структура БД Video Manager

10.9.1 Модель таблицы stream

Имя	Тип	Описание
id	bigint	Внутренний идентификатор потока.
account_id	varchar(36)	ID аккаунта в формате UUID4, к которому привязан поток.
name	varchar(128)	Имя потока.
description	varchar(512)	Описание потока.
type	varchar(36)	Тип потока - «videofile» или «stream».
reference	varchar(512)	Адрес потока.
rotation	smallint	Угол поворота кадра камеры.

status	smallint	Статус обработки поткоа: • «0» — Поток ожидает обработки («pending»). • «1» — Обработка потока в процессе («in_progress»). • «2» — Обработка потока выполнена («done»). • «3» — Обработка потока отправлена в перезагрузку сервером («restart»). • «4» — Обработка потока выполнена неуспешно («failure»). • «5» — Обработка потока остановлена пользователем («stop»).
version	varchar(10)	Версия потока.
create_time	timestamp	Время создания потока.
status_last_-update_time	timestamp	Последнее время обновления статуса.
last_feedback_time	timestamp	Последнее время отправки обратной связи.
stream_id	varchar(36)	Идентификатор потока.

10.9.2 Модель таблицы group

В данной таблице содержится информация о группе.

Имя	Тип	Описание
id	bigint	Внутренний идентификатор группы.
group_name	varchar(128)	Название группы.
account_id	varchar(36)	ID аккаунта в формате UUID4, к которому привязан поток.
description	varchar(256)	Описание группы.
create_time	timestamp	Время создания группы.
group_id	varchar(36)	Идентификатор группы.

10.9.3 Модель таблицы group_stream

В данной таблице содержится информация о связи групп и потоков.

Имя	Тип	Описание
group_id	varchar(36)	Идентификатор группы.
group_name	varchar(128)	Имя группы.
stream_id	varchar(36)	Идентификатор потока.
id	bigint	Внутренний идентификатор потока.

10.9.4 Модель таблицы restart

В данной таблице содержится информация об автоматическом перезапуске потока.

- ———————– + ———– + ———————————————————————- + | **Имя** | **Тип** | **Описание** | +———————-+———–+———————————————————————+ | restart | smallint | Включен ли автоматический перезапуск потоков. «1» — включен, «0» — выключен. | +———————-+———–+———————————————————————+ | attempt_count | smallint | Количество попыток, которые нужно делать для автоматического перезапуска. | +———————-+———–+———————————————————————+ | delay | integer | Задержка автоматического перезапуска. | +———————-+———–+———————————————————————+ | current_attempt | smallint | Текущее количество выполненных попыток перезапуска. | +———————-+———–+———————————————————————+ | last_attempt_time | timestamp | Время последней попытки перезапуска. | +———————-+———–+———————————————————————+ | status | smallint | Статус автоматического перезапуска потока: | | | | | | | | - «0» — Автоматический перезапуск отключен («disabled»). | | | | - «1» — Автоматический перезапуск включен («enabled»). | | | | - «2» — Автоматический перезапуск в процессе («in_progress»). | | | | - «3» — Ошибка автоматического перезапуска («failed»). | +———————-+———–+———————————————————————+ | stream_id | varchar(36) | Идентификатор потока. | +———————-+———–+———————————————————————+

10.9.5 Модель таблицы stream_meta

В данной таблице содержится информация о дополнительной информации потока, задаваемой в запросе на его создание.

Имя	Тип	Описание
account_id	varchar(36)	ID аккаунта в формате UUID4, к которому привязан поток.
city	varchar(36)	Город.
area	varchar(36)	Область.

Имя	Тип	Описание
district	varchar(36)	Район.
street	varchar(36)	Улица.
house_-number	varchar(36)	Номер дома.
geo_position	geography	Географические координаты (широта, долгота).
stream_id	varchar(36)	Идентификатор потока.

10.9.6 Модель таблицы video_analytic

В данной таблице содержится информация о видеоаналитике.

Имя	Тип	Описание
analytic_name	varchar(36)	Имя аналитики.
description	varchar(512)	Описание аналитики.
documentation	bytea	Файл документации.
version	smallint	Версия аналитики.
validation_schema	varchar	Схема валидации.
default_-parameters	varchar	Стандартные параметры аналитики.
account_id	varchar(36)	ID аккаунта в формате UUID4, к которому привязан поток.
create_time	timestamp	Время создания аналитики.
last_update_time	timestamp	Последнее время обновления аналитики.
analytic_id	varchar(36)	Идентификатор аналитики.

10.9.7 Модель таблицы agent

В данной таблице содержится информация об агенте.

Имя	Тип	Описание
agent_name	varchar(128)	Имя агента.
description	varchar(512)	Описание агента.

status	smallint	Статус агента: • «0» — Не готов («not_ready»). • «1» — Готов («ready»).
max_stream_count	smallint	Максимальное количество потоков, которые может обработать агент.
active_stream_-count	smallint	Текущее количество потоков, обрабатываемых агентом.
account_id	varchar(36)	ID аккаунта в формате UUID4, к которому привязан агент.
create_time	timestamp	Время создания агента.
last_update_time	timestamp	Время последнего обновления агента.
last_feedback_-time	timestamp	Последнее время отправки обратной связи.
agent_id	varchar(36)	Идентификатор агента.

10.9.8 Модель таблицы stream_analytic

В данной таблице содержится информация какие аналитики нужны потоку.

Имя	Тип	Описание
analytic_id	varchar(36)	Идентификатор аналитики.
stream_id	varchar(36)	Идентификатор потока.
analytic_name	varchar(36)	Название аналитики.
analytic_parameters	varchar	Параметры аналитики.
id	bigint	Внутренний идентификатор аналитики.

10.9.9 Модель таблицы agent_analytic

В данной таблице содержится информация с какими аналитиками может работать агент.

Имя	Тип	Описание
agent_id	varchar(36)	Идентификатор агента.
analytic_id	varchar(36)	Идентификатор аналитики.

Имя	Тип	Описание
analytic_name	varchar(36)	Имя аналитики.
id	bigint	Внутренний идентификатор агента.

10.9.10 Модель таблицы agent_stream

Имя	Тип	Описание
status	smallint	Внутренний статус агента.
agent_id	varchar(36)	Идентификатор агента.
stream_id	varchar(36)	Идентификатор потока.
id	bigint	Внутренний идентификатор.

10.9.11 Модель таблицы log

В данной таблице содержится информация о логах обработки потока.

Имя	Тип	Описание
stream_id	varchar(36)	Идентификатор потока.
time	timestamp	Время создания лога.
error	varchar	Описание ошибки обработки потока.
status	smallint	Статус обработки потока. • «0» — Поток ожидает обработки («pending»). • «1» — Обработка потока в процессе («in_progress»). • «2» — Обработка потока выполнена («done»). • «3» — Обработка потока отправлена в перезагрузку сервером («restart»). • «4» — Обработка потока выполнена неуспешно («failure»). • «5» — Обработка потока остановлена пользователем («stop»).
status_last_update_-time	timestamp	Время последнего обновления статуса.
meta	varchar	В настоящее время параметр не используется.

log_id	bigint	Идентификатор лога.

10.9.12 Модель таблицы luna_video_migrations

Имя	Тип	Описание
version_num	varchar(32)	Параметр, необходимый для миграции БД.

11 Ошибки API

Данный раздел описывает ошибки, которые возвращает сервис API. Каждая из ошибок API имеет свой уникальный номер. Его удобно использовать, чтобы найти ошибку.

Некоторые из этих ошибок могут иметь несколько различных причин возникновения.

В случае возникновения «Internal server error» или иной непредвиденной ошибки следует обратиться к логам сервиса для получения дополнительной информации о проблеме.

11.1 Общие ошибки

11.1.1 Вернулся код 0

Текст:

Success

Источник:

Общие

Описание:

Данный код возвращается, когда ошибок нет и запрос успешно обработан.

Обратите внимание, что результат все еще может содержать изображения, отфильтрованные в соответствии с параметрами, определенными в запросе.

11.1.2 Вернулся код ошибки 1

Текст ошибки:

Internal server error Unknown error

Источник ошибки:

Общие ошибки

Описание ошибки:

Произошла неожиданная внутренняя ошибка. Ошибка неправильно обработана в коде.

Если предоставляется какая-либо дополнительная трассировка, а ошибка продолжает появляться, отправьте трассировку в службу технической поддержки VisionLabs.

11.2 Ошибки HTTP-клиента

11.2.1 Вернулся код ошибки 3

Текст ошибки:

Invalid url {value}

Источник ошибки:

Ошибки HTTP-клиента

Описание ошибки:

В запросе был задан недопустимый URL.

Проверьте URL в запросе. Он может содержать пробелы или ошибочные символы.

Проверьте документацию API для примеров запросов. Выберите нужную версию LUNA PLATFORM с помощью переключателя версий.

11.2.2 Вернулся код ошибки 4

Текст ошибки:

Client payload error {value}

Источник ошибки:

Ошибки HTTP-клиента

Описание ошибки:

Ошибка в сообщении (payload), передаваемом в запросе.

Возможные причины: объект ответа был закрыт до того, как ответ получил всю информацию. Произошла ошибка шифрования передачи данных.

11.2.3 Вернулся код ошибки 5

Текст ошибки:

Server fingerprint mismatch {value}

Источник ошибки:

Ошибки HTTP-клиента

Описание ошибки:

Ошибка хэша публичного ключа сервера (fingerprint).

Была выполнена попытка подключиться к неправильному серверу или был использован неправильный ключ.

11.2.4 Вернулся код ошибки 6

Текст ошибки:

Socket read timeout Request timeout on {value} method {value}

Источник ошибки:

Ошибки HTTP-клиента

Описание ошибки:

Ошибка чтения данных через программный интерфейс. Превышено время ожидания выполнения запроса.

Проблема может возникнуть по следующим причинам:

- Проблемы в сети. Проверьте сеть.
- Внутренние ошибки сервера. Проверьте логи сервисов.

11.2.5 Вернулся код ошибки 7

Текст ошибки:

Socket connect timeout Request timeout on {value} method {value}

Источник ошибки:

Ошибки HTTP-клиента

Описание ошибки:

Ошибка соединения через программный интерфейс. Превышено время ожидания выполнения запроса.

Проблема может возникнуть по следующим причинам:

- Проблемы в сети. Проверьте сеть.
- Внутренние ошибки сервера. Проверьте логи сервисов.
- Высокая нагрузка на сервис.

11.2.6 Вернулся код ошибки 8

Текст ошибки:

Connect timeout on {value} method {value}

Источник ошибки:

Ошибки HTTP-клиента

Описание ошибки:

Ошибка подключения. Превышено время ожидания подключения.

Проблема может возникнуть по следующим причинам:

- Проблемы в сети. Проверьте сеть.
- Сервис недоступен. Убедитесь, что сервис запущен, и проверьте его логи.

11.2.7 Вернулся код ошибки 9

Текст ошибки:

Request timeout on {value} method {value}

Источник ошибки:

Ошибки HTTP-клиента

Описание ошибки:

Ошибка выполнения запроса. Превышено время ожидания выполнения запроса.

Проблема может возникнуть по следующим причинам:

- Проблемы в сети. Проверьте сеть.
- Сервис недоступен. Убедитесь, что сервис запущен, и проверьте его логи.

11.2.8 Вернулся код ошибки 10

Текст ошибки:

Server disconnected {value}

Источник ошибки:

Ошибки HTTP-клиента

Описание ошибки:

Связь с сервером потеряна.

Проблема может возникнуть по следующим причинам:

- Проблемы в сети. Проверьте сеть.
- Сервис недоступен. Убедитесь, что сервис запущен.

11.2.9 Вернулся код ошибки 11

Текст ошибки:

Server connection error {value}

Источник ошибки:

Ошибки HTTP-клиента

Описание ошибки:

Ошибка подключения к серверу.

Проблема может возникнуть по следующим причинам:

- Проблемы в сети. Проверьте сеть.
- Сервис недоступен. Убедитесь, что сервис запущен.

11.2.10 Вернулся код ошибки 12

Текст ошибки:

Client proxy connection error {value}

Источник ошибки:

Ошибки HTTP-клиента

Описание ошибки:

Ошибка подключения к прокси серверу.

Проблема может возникнуть по следующим причинам:

- Проблемы в сети. Проверьте сеть.
- Прокси-соединение недоступно.

11.2.11 Вернулся код ошибки 13

Текст ошибки:

Client connector SSL error {value}

Источник ошибки:

Ошибки HTTP-клиента

Описание ошибки:

Ошибка подключения с использованием SSL сертификата. Проверьте SSL сертификат.

11.2.12 Вернулся код ошибки 14

Текст ошибки:

Client connector certificate error {value}

Источник ошибки:

Ошибки HTTP-клиента

Описание ошибки:

Ошибка в SSL сертификате на стороне клиента.

Возможные причины: неверный сертификат SSL, устаревшая версия SSL клиента. Проверьте SSL сертификат.

11.2.13 Вернулся код ошибки 15

Текст ошибки:

Client SSL error {value}

Источник ошибки:

Ошибки HTTP-клиента

Описание ошибки:

Ошибка в SSL сертификате на стороне клиента.

Возможные причины: неверный сертификат SSL, устаревшая версия SSL клиента. Проверьте SSL сертификат.

11.2.14 Вернулся код ошибки 16

Текст ошибки:

Client connector error {value}

Источник ошибки:

Ошибки HTTP-клиента

Описание ошибки:

Ошибка подключения клиента.

11.2.15 Вернулся код ошибки 17

Текст ошибки:

Client OS error {value}

Источник ошибки:

Ошибки HTTP-клиента

Описание ошибки:

Произошла ошибка в операционной системе клиента.

Проверьте ОС клиента.

11.2.16 Вернулся код ошибки 18

Текст ошибки:

Client connection error {value}

Источник ошибки:

Ошибки HTTP-клиента

Описание ошибки:

Ошибка подключения клиента.

11.2.17 Вернулся код ошибки 19

Текст ошибки:

Client HTTP proxy error {value}

Источник ошибки:

Ошибки HTTP-клиента

Описание ошибки:

Произошла ошибка HTTP прокси сервера на стороне клиента.

11.2.18 Вернулся код ошибки 20

Текст ошибки:

WS server handshake error {value}

Источник ошибки:

Ошибки HTTP-клиента

Описание ошибки:

Ошибка установления связи через веб-сокет перед началом обмена данными.

11.2.19 Вернулся код ошибки 21

Текст ошибки:

Content-Type error {value}

Источник ошибки:

Ошибки HTTP-клиента

Описание ошибки:

Ошибка типа содержимого запроса. Неверное значение поля «Content-Type» в параметрах заголовка запроса. Оно не совпадает с данными, указанными в запросе.

См. раздел «HEADER PARAMETERS» в документации API для вашего запроса. Выберите спецификацию для требуемого сервиса. Убедитесь, что версия спецификации соответствует используемой в настоящее время версии LUNA PLATFORM.

11.2.20 Вернулся код ошибки 22

Текст ошибки:

Client response error {value}

Источник ошибки:

Ошибки HTTP-клиента

Описание ошибки:

Ошибка при получении ответа от клиента.

11.2.21 Вернулся код ошибки 23

Текст ошибки:

HTTP client error {value}

Источник ошибки:

Ошибки HTTP-клиента

Описание ошибки:

Ошибка HTTP на стороне клиента.

11.2.22 Вернулся код ошибки 24

Текст ошибки:

HTTP client error {value}

Источник ошибки:

Ошибки HTTP-клиента

Описание ошибки:

Ошибка HTTP на стороне клиента. Ответ был получен не в формате JSON.

11.3 Ошибки предыдущих версий

11.3.1 Вернулся код ошибки 5101

Текст ошибки:

Reference uuid is missing. Reference uuid is missing

Источник ошибки:

Ошибки предыдущих версий

Описание ошибки:

Ошибка сервиса Matcher.

Предоставленный UUID эталона для сравнения не был найден. Убедитесь, что UUID указан правильно и что под этим UUID существует биометрический шаблон. Убедитесь, что эталон доступен для текущего идентификатора учетной записи.

11.3.2 Вернулся код ошибки 5102

Текст ошибки:

Reference uuid has no extracted descriptor. Reference uuid has no extracted descriptor

Источник ошибки:

Ошибки предыдущих версий

Описание ошибки:

Ошибка сервиса Matcher.

Не найден извлечённый биометрический шаблон для UUID эталона. Биометрический шаблон не был извлечен для предоставленного эталона или он был извлечен с использованием другой версии нейронной сети.

Проверьте существование биометрического шаблона и его версию.

11.4 Ошибки базы данных

11.4.1 Вернулся код ошибки 10015

Текст ошибки:

SQL error SQL request execution failed

Источник ошибки:

Ошибки базы данных

Описание ошибки:

Выполнение запроса SQL не удалось. Возникла ошибка с БД.

БД недоступна по неизвестной причине, либо в ней отсутствуют необходимые таблицы. Следует запросить статус БД, проверить доступность БД по сети, проверить существование необходимых таблиц в БД.

11.4.2 Вернулся код ошибки 10016

Текст ошибки:

Database error Could not connect to database

Источник ошибки:

Ошибки базы данных

Описание ошибки:

Не удалось выполнить подключение к базе данных.

Проверьте доступность базы данных. При запуске базы данных могут возникать ошибки или она может быть недоступна в текущей сети (из-за ограничений сервера или сетевых проблем). Проверьте настройки базы данных в параметрах сервисов. Они могут быть неверными.

11.4.3 Вернулся код ошибки 10017

Текст ошибки:

Database error Database connection timeout error

Источник ошибки:

Ошибки базы данных

Описание ошибки:

Превышен лимит времени подключения к базе данных.

Проверьте доступность базы данных. При запуске базы данных могут возникать ошибки или она может быть недоступна в текущей сети (из-за ограничений сервера или сетевых проблем). Проверьте настройки базы данных в параметрах сервисов. Они могут быть неверными.

11.4.4 Вернулся код ошибки 10018

Текст ошибки:

Database error {value}

Источник ошибки:

Ошибки базы данных

Описание ошибки:

В базе данных произошла ошибка.

11.5 Ошибки сервиса API

11.5.1 Вернулся код ошибки 11009

Текст ошибки:

Internal server error

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Ошибка возникает при неизвестном исключении, которое пришло на уровень обработчика REST ресурса.

Рекомендуется проверить логи сервиса, чтобы узнать больше информации об ошибке. Если предоставлена какая-либо дополнительная трассировка и ошибка продолжает появляться, отправьте трассировку в службу технической поддержки VisionLabs.

11.5.2 Вернулся код ошибки 11020

Текст ошибки:

Object not found One or more {value} not found

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Один или более указанных объектов не были найдены.

Проверьте существование данных объектов.

11.5.3 Вернулся код ошибки 11027

Текст ошибки:

External request failed Failed to download image by url {value}

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Внешний запрос не удался. Не удалось получить изображение по указанному в теле запроса URL.

Убедитесь, что URL-адрес указан правильно, а его данные доступны и существуют.

11.5.4 Вернулся код ошибки 11028

Текст ошибки:

Bad/incomplete input data Bad content type of image {value}

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Недопустимый тип содержимого изображения в запросе. Проверьте поле «Content-Type» в запросе.

11.5.5 Вернулся код ошибки 11029

Текст ошибки:

Bad/incomplete input data Bad content type of image in multipart body

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Неверные входные данные в составном запросе (multipart). Одно или несколько изображений имеют недопустимый формат или размер.

Убедитесь, что поле Content-Type запроса задано правильно. Проверьте требования к изображению для запроса.

Проверьте запросы в спецификации API сервиса API.

11.5.6 Вернулся код ошибки 11030

Текст ошибки:

Bad/incomplete input data Image size is not equal to 250x250

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Размер передаваемого в запросе нормализованного изображения не равен 250x250. Скорее всего, изображение не является нормализованным.

11.5.7 Вернулся код ошибки 11031

Текст ошибки:

Bad/incomplete input data Sample with id {value} not found

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Не удалось найти нормализованное изображение с указанным ID.

Убедитесь, что введен корректный идентификатор, и он создан под вашим текущим идентификатором аккаунта.

11.5.8 Вернулся код ошибки 11032

Текст ошибки:

Object not found Face with id {value} does not have attributes yet

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

У лица с указанным ID ещё нет атрибутов.

Ошибка может возникнуть при запросе на сравнение Лица, у которого нет биометрического шаблона.

Проверьте, что введен корректный идентификатор.

Если для Лица нет биометрического шаблона, вы можете прикрепить его с помощью запросов PUT face attributes or create a new face.

11.5.9 Вернулся код ошибки 11034

Текст ошибки:

Bad/incomplete input data Descriptor for {value} «{value}» was not extracted

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Биометрический шаблон для указанного объекта не был извлечён. Следует проверить корректность ID объекта в запросе.

11.5.10 Вернулся код ошибки 11035

Текст ошибки:

Service name not found Service name {value} not found

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Указанное имя сервиса не найдено.

11.5.11 Вернулся код ошибки 11036

Текст ошибки:

Forbidden Luna-Account-Id header is required for requests which change the state of system

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

В запросе требуется указать корректный заголовок «Luna-Account-Id».

Заголовок определяет пользователя, который меняет состояние системы. Возможно, указан неверный пользователь, который не имеет доступа к БД.

Проверьте раздел «HEADER PARAMETERS» в спецификации API сервиса API.

11.5.12 Вернулся код ошибки 11037

Текст ошибки:

Bad/incomplete input data Luna-Account-Id header is not UUID format: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Заголовок Luna-Account-Id в запросе задан не в формате UUID.

Необходимый формат: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx.

11.5.13 Вернулся код ошибки 11038

Текст ошибки:

Multiple faces found Multiple faces found in image {value} detect {value}

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

На изображении больше одного Лица.

Значение параметра «multiface_policy» в запросе равно «0», или несколько лиц не обрабатываются в запросе по умолчанию. Изображения с несколькими лицами в этом случае не обрабатываются и возвращается ошибка.

Кроме того, данная ошибка может появиться, если на изображении найдено более одного лица и предпринята попытка агрегирования со значением «multiface_policy», отличным от «2». При выполнении агрегирования убедитесь, что значение параметра «multiface_policy» в обработчике установлено равным «2».

11.5.14 Вернулся код ошибки 11039

Текст ошибки:

Forbidden Luna Tasks service is disabled

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Запрос в сервис Tasks не может быть обработан.

Использование сервиса Tasks отключено в конфигурации API.

Проверьте параметр «ADDITIONAL_SERVICES_USAGE» в сервисе Configurator или в файле конфигурации.

11.5.15 Вернулся код ошибки 11040

Текст ошибки:

Forbidden Luna Events service is disabled

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Запрос в сервис Events не может быть обработан.

Использование сервиса Events отключено в конфигурации API.

Проверьте параметр «ADDITIONAL_SERVICES_USAGE» в сервисе Configurator или в файле конфигурации (измените этот параметр в конфигурации каждого сервиса, который использует сервис Events).

11.5.16 Вернулся код ошибки 11041

Текст ошибки:

Object not found No one face found with external id {value}

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Объект face для указанного external ID не найден.

Проверьте external_id и account_id в запросе.

11.5.17 Вернулся код ошибки 11042

Текст ошибки:

Internal server error Unhandled exception: {value}

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Произошло необработанное исключение.

Рекомендуется проверить логи сервиса, чтобы узнать больше информации об ошибке.

Если предоставлена какая-либо дополнительная трассировка и ошибка продолжает появляться, отправьте трассировку в службу технической поддержки VisionLabs.

11.5.18 Вернулся код ошибки 11043

Текст ошибки:

Bad/incomplete input data. «{value}»

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

В запросе указаны неверные или неполные входные данные для биометрического образца.

Проверьте формат входных данных.

11.5.19 Вернулся код ошибки 11044

Текст ошибки:

Forbidden {value} turned off on luna-api instance

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Для экземпляра API процесс отключен. Проверьте предоставленную информацию в конфигурации экземпляра API.

11.5.20 Вернулся код ошибки 11045

Текст ошибки:

Forbidden Only one detection rect for each image available at the moment

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

На данный момент для каждого изображения доступен только один bbox. Ошибка возвращается, когда в запросе указано несколько bbox.

11.5.21 Вернулся код ошибки 11046

Текст ошибки:

Bad/incomplete input data More than one file named {value} found

Источник ошибки: Ошибки сервиса API

Описание ошибки:

В запросе было отправлено несколько изображений с одинаковым именем. Изображения должны иметь уникальные имена.

11.5.22 Вернулся код ошибки 11047

Текст ошибки:

Bad/incomplete input data Bounding box not available for warp images

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Bbox, заданный в запросе, недоступен для нормализованных изображений.

11.5.23 Вернулся код ошибки 11048

Текст ошибки:

Bad/incomplete input data Multiplie bounding boxes lists in request

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

В запросе указано несколько bbox. Для изображения можно указать только один bbox.

11.5.24 Вернулся код ошибки 11049

Текст ошибки:

Bad/incomplete input data {value}

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

В составном запросе (multipart) представлены неверные или неполные входные данные. Проверьте описание в сообщении ошибки.

11.5.25 Вернулся код ошибки 11050

Текст ошибки:

Bad/incomplete input data More than one bounding box for image named {value} found

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Изображение имеет несколько bbox.

11.5.26 Вернулся код ошибки 11051

Текст ошибки:

Internal server error Service «static» folder not being loaded

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Директория службы `static` не была загружена.

11.5.27 Вернулся код ошибки 11052

Текст ошибки:

Bad/incomplete input data No one attributes was settled for the extract

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Оба параметра «extract_descriptor» и «extract_basic_attributes» имеют значение «0». Вы должны включить хотя бы один параметр в запросе.

11.5.28 Вернулся код ошибки 11053

Текст ошибки:

Multiple human bodies found on image {value} detect {value}

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

На изображении обнаружено несколько тел людей.

11.5.29 Вернулся код ошибки 11055

Текст ошибки:

Forbidden License problem: «{value}»

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Ошибка лицензии. Лицензия истекла или недоступна.

11.5.30 Вернулся код ошибки 11056

Текст ошибки:

Bad/incomplete input data {value}

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Запрос недействителен. Данные запроса неполные. Проверьте сообщение об ошибке и входные данные в запросе.

11.5.31 Вернулся код ошибки 11057

Текст ошибки:

Object not found No one event found with external id {value}

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Не было найдено ни одного события с указанным «external_id». Проверьте заданные эталоны для сравнения в теле запроса.

11.5.32 Вернулся код ошибки 11058

Текст ошибки:

Object not found No one event found with track id {value}

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Не было найдено ни одного события с указанным «track_id». Проверьте заданные эталоны для сравнения в теле запроса.

11.5.33 Вернулся код ошибки 11059

Текст ошибки:

Forbidden Request denied due to token restrictions, required «{value}» permission for «{value}»

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Запрос отклонен из-за отсутствия указанных разрешений в используемом токене.

Проверьте поле «permissions» у используемого токена.

11.5.34 Вернулся код ошибки 11060

Текст ошибки:

Forbidden Access to the resource is denied due to token restrictions. Required «{value}» resource permission

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Доступ к ресурсу отклонен из-за отсутствия разрешения на использование указанного ресурса в используемом токене.

Проверьте поле «permissions» > «resources» у используемого токена.

11.5.35 Вернулся код ошибки 11061

Текст ошибки:

Forbidden Authorization with «Luna-Account-Id» header is disabled

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Авторизация с помощью Luna-Account-Id отключена.

Проверьте настройку «ALLOW_LUNA_ACCOUNT_AUTH_HEADER» в сервисе Configurator.

11.5.36 Вернулся код ошибки 11062

Текст ошибки:

Forbidden Specified token corrupted

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Токен поврежден.

Данная ошибка может появиться при попытке воспользоваться токеном с параметром «visibility_-area» = «all» с типом аккаунта «user», который был ранее понижен с типа «advanced_user» или «admin».

11.5.37 Вернулся код ошибки 11063

Текст ошибки:

Forbidden The query parameter `account_id` usage denied due to account restrictions

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Использование параметра запроса `account_id` запрещено в связи с ограничениями аккаунта.

Данная ошибка может появиться при попытке использования фильтра `account_id` со значением, отличным от идентификатора аккаунта пользователя с типом «user».

11.5.38 Вернулся код ошибки 11064

Текст ошибки:

Forbidden Access to the resource is denied using authorization by token

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Доступ к ресурсу запрещен при авторизации по токену.

11.5.39 Вернулся код ошибки 11065

Текст ошибки:

Authorization failed, {value}

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Неуспешная авторизация. В ошибке указаны параметры, не прошедшие авторизацию.

11.5.40 Вернулся код ошибки 11066

Текст ошибки:

Bad/incomplete input data, Luna-Account-Id header not found

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Не найден обязательный заголовок Luna-Account-Id при попытке создать токен в сервисе Accounts.

11.5.41 Вернулся код ошибки 11067

Текст ошибки:

Forbidden, Request denied due to token restrictions. According to `emit_events` the handler is not whitelisted or handler is blacklisted.

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Запрос отклонен из-за ограничений токена. Используемый `handler_id` либо внесен в черный список, либо отсутствует в белом списке.

Проверьте поле «permissions» > «emit_events» > «black_list»/«white_list» у используемого токена.

11.5.42 Вернулся код ошибки 11068

Текст ошибки:

Forbidden Request denied due to token restrictions. Events generation is not allowed

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Запрос отклонен из-за ограничений токена. Запрещена генерация событий по всем существующим обработчикам.

Проверьте поле «permissions» > «emit_events» > «allowed» у используемого токена.

11.5.43 Вернулся код ошибки 11069

Текст ошибки:

Forbidden Account id specified in {value} is different from requester account id, that is not acceptable due to account/token visibility area restrictions

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Идентификатор аккаунта, заданный в указанной политике, отличается от идентификатора аккаунта запроса, что недопустимо из-за ограничений области видимости аккаунта/токена.

Убедитесь, что идентификаторы аккаунтов совпадают или измените область видимости аккаунта/токена.

11.5.44 Вернулся код ошибки 11070

Текст ошибки:

Forbidden, Luna Image Store service is disabled

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Запрос в сервис Image Store не может быть обработан.

Использование сервиса Image Store отключено в конфигурации API.

Проверьте параметр «ADDITIONAL_SERVICES_USAGE» в сервисе Configurator или в файле конфигурации (измените этот параметр в конфигурации каждого сервиса, который использует сервис Image Store).

11.5.45 Вернулся код ошибки 11071

Текст ошибки:

Forbidden, Luna Handlers service is disabled

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Запрос в сервис Handlers не может быть обработан.

Использование сервиса Handlers отключено в конфигурации API.

Проверьте параметр «ADDITIONAL_SERVICES_USAGE» в сервисе Configurator или в файле конфигурации (измените этот параметр в конфигурации каждого сервиса, который использует сервис Handlers).

11.5.46 Вернулся код ошибки 11072

Текст ошибки:

Forbidden, Luna Lambda service is disabled

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Запрос в сервис Lambda не может быть обработан.

Использование сервиса Lambda отключено в конфигурации API.

Проверьте параметр «ADDITIONAL_SERVICES_USAGE» в сервисе Configurator или в файле конфигурации (измените этот параметр в конфигурации каждого сервиса, который использует сервис Lambda).

11.5.47 Вернулся код ошибки 11074

Текст ошибки:

Forbidden. Request denied due to token restrictions. According to `verify` the verifier is not whitelisted or verifier is blacklisted.

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Запрос отклонен из-за ограничений токена. Используемый `verifier_id` либо внесен в черный список, либо отсутствует в белом списке.

Проверьте поле «permissions» > «verify» > «black_list»/«white_list» у используемого токена.

11.5.48 Вернулся код ошибки 11075

Текст ошибки:

Forbidden Luna Video Manager service is disabled

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Сервис Video Manager отключён.

Проверьте настройки «ADDITIONAL_SERVICES_USAGE» в сервисе Configurator или в конфигурационном файле.

11.5.49 Вернулся код ошибки 11076

Текст ошибки:

Bad/incomplete input data. LUNA-STREAMS-API-VERSION header only allows «1» or «2» values.

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Запрос отклонен из-за некорректных данных, переданных в заголовке «LUNA-STREAMS-API-VERSION».

Можно использовать только значения «1» или «2».

11.5.50 Вернулся код ошибки 11077

Текст ошибки:

Forbidden. Video Agent service is disabled

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Сервис Video Agent отключён.

Проверьте настройки «ADDITIONAL_SERVICES_USAGE» в сервисе Configurator или в конфигурационном файле.

11.5.51 Вернулся код ошибки 11078

Текст ошибки:

Forbidden. Luna Streams Retranslator service is disabled.

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Сервис Streams Retranslator отключён.

Проверьте настройки «ADDITIONAL_SERVICES_USAGE» в сервисе Configurator или в конфигурационном файле.

11.5.52 Вернулся код ошибки 11079

Текст ошибки:

Forbidden. Luna Faces service is disabled.

Источник ошибки:

Ошибки сервиса API

Описание ошибки:

Сервис Faces отключён.

Проверьте настройки «ADDITIONAL_SERVICES_USAGE» в сервисе Configurator или в конфигурационном файле.

11.6 Общие ошибки REST API

11.6.1 Вернулся код ошибки 12002

Текст ошибки:

Bad/incomplete input data Request does not contain json

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Запрос не содержит JSON.

Проверьте синтаксис JSON в теле сообщения:

- Могут быть добавлены лишние символы или пробел.
- Может быть нарушена структура JSON.
- Могли быть упущены важные синтаксические элементы.

11.6.2 Вернулся код ошибки 12003

Текст ошибки:

Bad/incomplete input data Field {value} not found in json

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Указанное поле не найдено в теле JSON. Проверьте, что в запросе указаны все обязательные поля.

См. документацию API для запроса.

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.6.3 Вернулся код ошибки 12005

Текст ошибки:

Bad/incomplete input data Request contain empty json

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Запрос содержит пустой JSON. Для выполнения запроса требуется тело запроса.

См. документацию API для запроса.

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.6.4 Вернулся код ошибки 12010

Текст ошибки:

Bad/incomplete input data This resource needs «Authorization» authorization headers

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Для выполнения запроса требуется указать заголовок «Authorization». Обычно авторизация требуется для выполнения запросов к сервисам Admin и Backport 3.

11.6.5 Вернулся код ошибки 12012

Текст ошибки:

Bad/incomplete input data Bad query parameters {value}

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Указанные параметры запроса заданы неправильно.

11.6.6 Вернулся код ошибки 12013

Текст ошибки:

Resource not found Page not found

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

HTTP ресурс не найден.

Указанный в запросе URL-путь не найден.

Например, в запросе указан путь «/6/sample» вместо «/6/samples». Проверьте путь и сравните с указанным в документации.

См. документацию API для запроса.

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.6.7 Вернулся код ошибки 12014

Текст ошибки:

Bad/incomplete input data Required parameters {value} not found

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

В запросе не найдены обязательные параметры запроса. Эти параметры перечислены в ошибке.

См. документацию API для запроса.

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.6.8 Вернулся код ошибки 12016

Текст ошибки:

Bad/incomplete input data No one parameters {value} not found

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

В запросе не найден ни один из перечисленных параметров запроса.

См. документацию API для запроса. Сравните ваши параметры запроса с параметрами из раздела «QUERY PARAMETERS».

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.6.9 Вернулся код ошибки 12017

Текст ошибки:

Bad/incomplete input data Bad content type

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Недопустимое значение заголовка «Content-Type», передаваемого в запросе.

Заголовок не задан, задано недопустимое значение заголовка, или в заголовке допущена ошибка.

См. документацию API для запроса. Сравните ваши параметры запроса с параметрами из раздела «HEADER PARAMETERS».

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.6.10 Вернулся код ошибки 12021

Текст ошибки:

Method not allowed Method not allowed

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

HTTP ресурс не поддерживает данный метод. Проверьте HTTP-метод, указанный для отправки запроса.

См. документацию API для запроса. Проверьте метод (POST, GET, PATCH и т.д.), указываемый для требуемых запросов. Например, POST /6/handlers.

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.6.11 Вернулся код ошибки 12022

Текст ошибки:

Bad/incomplete input data Failed to validate input json. Path: «{value}» message: «{value}»

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Ошибка в синтаксисе JSON по указанному пути.

Тело запроса содержит недопустимые поля. Проверьте запрос на соответствие шаблону, указанному в документации API для запроса.

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

Подробное описание ошибки приведено в поле «message».

11.6.12 Вернулся код ошибки 12023

Текст ошибки:

Bad/incomplete input data Content type is unacceptable allowed types: «{value}»

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Запрос содержит некорректный тип содержимого. Скорее всего была допущена опечатка в поле «Content-Type» запроса.

Проверьте доступные типы содержимого. См. документацию API для запроса.

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.6.13 Вернулся код ошибки 12024

Текст ошибки:

Bad/incomplete input data Unsupported media type

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Тип данных, передаваемых в запросе, не поддерживается.

Проверьте доступные типы содержимого для запроса. См. документацию API для запроса.

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.6.14 Вернулся код ошибки 12025

Текст ошибки:

Bad/incomplete input data Specified content type does not match data type

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Заданный тип содержимого «Content-Type» не соответствует типу данных.

См. документацию API для запроса. Проверьте доступные типы содержимого для запроса.

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.6.15 Вернулся код ошибки 12027

Текст ошибки:

Bad/incomplete input data Failed to validate input json. Message: {value}

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

В теле запроса неправильно заданы параметры. Проверьте тело JSON на соответствие требованиям к составлению запроса. Подробности указаны в поле «Message:».

См. документацию API для запроса. Проверьте «REQUEST BODY SCHEMA» запроса. Ознакомьтесь с примерами запросов в документации.

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.6.16 Вернулся код ошибки 12028

Текст ошибки:

Bad/incomplete input data Failed to parse Flatbuf

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Внутренняя ошибка. Не удалось десериализировать Flatbuf.

11.6.17 Вернулся код ошибки 12029

Текст ошибки:

Functionality is not implemented Required server functionality is not implemented

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Ошибка возвращается если запрашиваемый функционал сервера не реализован.

11.6.18 Вернулся код ошибки 12030

Текст ошибки:

Bad/incomplete input data Request does not contain data

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Запрос не содержит данных. Тело запроса пустое.

См. документацию API для запроса. Проверьте «REQUEST BODY SCHEMA» запроса и прочие параметры запроса. Ознакомьтесь с примерами запросов в документации.

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.6.19 Вернулся код ошибки 12031

Текст ошибки:

Bad/incomplete input data {value}

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Введенные данные неполные или неверные.

Подробное описание ошибки приведено в поле «message».

11.6.20 Вернулся код ошибки 12032

Текст ошибки:

Internal server error Document file not found

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Документ с указанным именем не найден. Ошибка возвращается при запросе документации сервиса.

Проверьте корректность запроса и имени документа.

См. документацию API для запроса.

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.6.21 Вернулся код ошибки 12033

Текст ошибки:

Forbidden Access to this resource on the server is denied

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Доступ к ресурсу на сервере запрещен.

Проверьте доступность ресурсов. Доступ может быть ограничен в настройках сервера.

11.6.22 Вернулся код ошибки 12034

Текст ошибки:

Bad/incomplete input data Descriptor has incorrect length {value}

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Биометрический шаблон имеет неправильную длину.

Убедитесь, что используется биометрический шаблон соответствующего формата. Биометрические шаблоны, полученные из разных источников, могут иметь разную длину.

Используйте ресурс «/sdk», чтобы получить необработанный биометрический шаблон требуемого формата и использовать его в своих запросах.

См. документацию API для запроса.

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.6.23 Вернулся код ошибки 12035

Текст ошибки:

Bad/incomplete input data Failed to parse xpk file

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Не удалось проанализировать файл XPK с биометрическим шаблоном. Проверьте правильность файла.

11.6.24 Вернулся код ошибки 12036

Текст ошибки:

Bad/incomplete input data Descriptor version {value} is not registered in the system

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Указанная версия биометрического шаблона не зарегистрирована в системе.

Установлена неправильная версия или используется несовместимая версия сервиса, которая не может обработать эту версию биометрического шаблона.

Проверьте, что используется корректная версия биометрического шаблона. См. доступные версии биометрических шаблонов в разделе «Нейронные сети».

11.6.25 Вернулся код ошибки 12037

Текст ошибки:

Bad/incomplete input data XPK file does not contain descriptor

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Входные данные XPK не содержат биометрического шаблона. Проверьте файл.

11.6.26 Вернулся код ошибки 12038

Текст ошибки:

Bad/incomplete input data SDK descriptor is not valid

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Биометрический шаблон SDK недействителен. Проверьте переданные данные биометрического шаблона.

Убедитесь, что используется биометрический шаблон соответствующего формата. Биометрические шаблоны, полученные из разных источников, могут иметь разную длину.

Используйте ресурс «/sdk», чтобы получить необработанный биометрический шаблон требуемого формата и использовать его в своих запросах.

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.6.27 Вернулся код ошибки 12039

Текст ошибки:

Bad/incomplete input data Unknown multipart name «{value}»

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

В запросе указано неизвестное составное (multipart) имя. Проверьте корректность имени в поле «message».

11.6.28 Вернулся код ошибки 12040

Текст ошибки:

Bad/incomplete input data Duplicate multipart name «{value}»

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

В запросе содержится дублированное составное (multipart) имя. Проверьте имя в поле «message». Не допускается использование дублированных имен.

11.6.29 Вернулся код ошибки 12041

Текст ошибки:

Bad/incomplete input data Multipart with name «{value}» has bad Content-Type

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Входные данные с указанным именем имеют неверный «Content-Type» в составном запросе. Проверьте указанный «Content-Type».

11.6.30 Вернулся код ошибки 12042

Текст ошибки:

Gone Access to this resource on the server is no longer available

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Доступ к ресурсу на сервере недоступен. Убедитесь, что соответствующий сервис работает и доступен. Также убедитесь, что ресурс указан правильно.

11.6.31 Вернулся код ошибки 12043

Текст ошибки:

Unknown error Service «{value}» unknown error method: «{value}» url: «{value}»

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Произошла неизвестная ошибка сервиса. Проверьте метод и URL, предоставленный в сообщении ошибки.

11.6.32 Вернулся код ошибки 12044

Текст ошибки:

Bad/incomplete input data Failed to decompress input body using gzip encoder

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Не удалось распаковать тело запроса с помощью кодировщика GZIP.

11.6.33 Вернулся код ошибки 12045

Текст ошибки:

Bad/incomplete input data Failed to decompress input body using deflate encoder

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Не удалось распаковать тело запроса с помощью кодировщика deflate.

11.6.34 Вернулся код ошибки 12046

Текст ошибки:

Bad/incomplete input data. Invalid http request: {value}

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Задан некорректный HTTP-запрос.

11.6.35 Вернулся код ошибки 12047

Текст ошибки:

Bad/incomplete input data. Failed to validate input msgpack. Path: «{value}», message: «{value}»

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Входные данные, переданные в формате MessagePack, не соответствуют ожидаемой схеме данных. Конкретное сообщение об ошибке «Path: „{value}“, message: „{value}“» предоставляет информацию о местоположении и содержании ошибочных данных.

Проверьте входные данные и убедитесь, что они соответствуют ожидаемой схеме данных.

11.6.36 Вернулся код ошибки 12048

Текст ошибки:

Bad/incomplete input data. Archive file does not contain plugin.py

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Приложенный архив не содержит обязательного файла `plugin.py`.

Убедитесь, что файл существует в архиве.

11.6.37 Вернулся код ошибки 12049

Текст ошибки:

Forbidden. Resource is disabled

Источник ошибки:

Общие ошибки REST API

Описание ошибки:

Использование ресурса `/metrics` отключено в параметре «enabled» группы «LUNA_SERVICE_-METRICS».

Включите параметр «enabled» и повторите запрос после перезапуска сервисов.

11.7 Ошибки сервиса Image Store

11.7.1 Вернулся код ошибки 13003

Текст ошибки:

Object not found Image with id {value} not found

Источник ошибки:

Ошибки сервиса Image Store

Описание ошибки:

Изображение с указанным ID не найдено не найдено в хранилище. Скорее всего была допущена опечатка в идентификатора, файл был удален или не существует.

11.7.2 Вернулся код ошибки 13004

Текст ошибки:

Bad/incomplete input data Image count exceeded limit 1000

Источник ошибки:

Ошибки сервиса Image Store

Описание ошибки:

Количество изображений в запросе превысило лимит 1000.

Уменьшите количество удаляемых изображений в запросе.

11.7.3 Вернулся код ошибки 13005

Текст ошибки:

Object not found Bucket with name {value} not found

Источник ошибки:

Ошибки сервиса Image Store

Описание ошибки:

Указанный бакет не найден.

Следует проверить наличие бакета.

Путь до бакета задается в конфигурационном файле сервиса Image Store или в сервисе Configurator в переменной «LOCAL_STORAGE». При отсутствии бакета следует вручную создать его. См. раздел «Создание бакетов» в документе по установке.

11.7.4 Вернулся код ошибки 13006

Текст ошибки:

Unique constraint error Bucket with name {value} already exist

Источник ошибки:

Ошибки сервиса Image Store

Описание ошибки:

Бакет с указанным именем уже существует. Ошибка возникает при попытке создать два бакета с одинаковыми именами. Не может быть два бакета с одинаковыми именами.

11.7.5 Вернулся код ошибки 13007

Текст ошибки:

Object not found Object with id {value} not found

Источник ошибки:

Ошибки сервиса Image Store

Описание ошибки:

Объект с указанным ID не найден в бакете. Объект не существует, удален или в ID допущена опечатка.

11.7.6 Вернулся код ошибки 13008

Текст ошибки:

Unique constraint error Object with id {value} already exist

Источник ошибки:

Ошибки сервиса Image Store

Описание ошибки:

Объект с указанным ID уже существует в бакете.

11.7.7 Вернулся код ошибки 13009

Текст ошибки:

Bad/incomplete input data Object count exceeded limit 1000

Источник ошибки:

Ошибки сервиса Image Store

Описание ошибки:

Количество объектов, передаваемых в бакет в запросе, превысило лимит 1000.

11.8 Ошибки сервиса Admin

11.8.1 Вернулся код ошибки 15012

Текст ошибки:

Object not found Account with id «{value}» not found

Источник ошибки:

Ошибки сервиса Admin

Описание ошибки:

Аккаунт с указанным идентификатором не найден в сервие LUNA Admin.

11.8.2 Вернулся код ошибки 15013

Текст ошибки:

Unique constraint error Account with same email or id already exist

Источник ошибки:

Ошибки сервиса Admin

Описание ошибки:

Аккаунт с указанным e-mail или идентификатором уже существует.

11.8.3 Вернулся код ошибки 15014

Текст ошибки:

Bad/incomplete input data Login or password is incorrect

Источник ошибки:

Ошибки сервиса Admin

Описание ошибки:

Неверно указаны данные. Введён неправильный логин или пароль от аккаунта.

11.8.4 Вернулся код ошибки 15015

Текст ошибки:

Bad/incomplete input data, Admin account type required

Источник ошибки:

Ошибки сервиса Admin

Описание ошибки:

Неверно указаны данные. Для выполнения запроса требуется тип аккаунта «admin».

11.9 Ошибки обработки изображений

11.9.1 Вернулся код ошибки 18001

Текст ошибки:

Bad/incomplete input data Failed convert data from base64 to bytes

Источник ошибки:

Ошибки обработки изображений

Описание ошибки:

Не удалось сконвертировать BASE64 строку в байты.

Переданная строка BASE64 повреждена.

11.9.2 Вернулся код ошибки 18002

Текст ошибки:

Bad/incomplete input data Failed convert bytes to {value}

Источник ошибки:

Ошибки обработки изображений

Описание ошибки:

Не удалось сконвертировать байты в указанный формат изображения. В запросе не было передано изображение.

11.9.3 Вернулся код ошибки 18003

Текст ошибки:

Bad/incomplete input data Failed read bytes as image

Источник ошибки:

Ошибки обработки изображений

Описание ошибки:

Невозможно прочесть байты изображения. Переданное изображение повреждено или имеет недопустимый формат.

11.10 Ошибки сервиса Image Store, хранящего данные на локальном хранилище

11.10.1 Вернулся код ошибки 19001

Текст ошибки:

Internal server error Failed to save image in the storage

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные на локальном хранилище

Описание ошибки:

Сервис Image Store не смог сохранить изображение на диск.

Следует проверить логи сервиса Image Store. Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с диском, проблема в сервисе Image Store.

11.10.2 Вернулся код ошибки 19002

Текст ошибки:

Internal server error Failed to remove image from the storage

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные на локальном хранилище

Описание ошибки:

Сервис Image Store не смог удалить изображение с диска.

Следует проверить логи сервиса Image Store. Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с диском, проблема в сервисе Image Store.

11.10.3 Вернулся код ошибки 19003

Текст ошибки:

Internal server error Failed to remove image from the storage

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные на локальном хранилище

Описание ошибки:

Сервис Image Store не смог удалить несколько изображений с диска.

Следует проверить логи сервиса Image Store. Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с диском, проблема в сервисе Image Store.

11.10.4 Вернулся код ошибки 19004

Текст ошибки:

Internal server error Failed to create bucket

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные на локальном хранилище

Описание ошибки:

Сервис Image Store не смог создать бакет на хранилище.

Следует проверить логи сервиса Image Store. Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с диском, проблема в сервисе Image Store.

11.10.5 Вернулся код ошибки 19005

Текст ошибки:

Internal server error Failed to get bucket list

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные на локальном хранилище

Описание ошибки:

Сервис Image Store не смог получить список бакетов на хранилище.

Следует проверить логи сервиса Image Store. Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с диском, проблема в сервисе Image Store.

11.10.6 Вернулся код ошибки 19006

Текст ошибки:

Internal server error Failed to get image from the storage

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные на локальном хранилище

Описание ошибки:

Сервис Image Store не смог получить изображение с диска.

Следует проверить логи сервиса Image Store. Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с диском, проблема в сервисе Image Store.

11.10.7 Вернулся код ошибки 19007

Текст ошибки:

Internal server error Failed to save object in the storage

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные на локальном хранилище

Описание ошибки:

Сервис Image Store не смог сохранить объект на диск.

Следует проверить логи сервиса Image Store. Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с диском, проблема в сервисе Image Store.

11.10.8 Вернулся код ошибки 19008

Текст ошибки:

Internal server error Failed to remove object from the storage

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные на локальном хранилище

Описание ошибки:

Сервис Image Store не смог удалить объект с диска.

Следует проверить логи сервиса Image Store. Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с диском, проблема в сервисе Image Store.

11.10.9 Вернулся код ошибки 19009

Текст ошибки:

Internal server error Failed to remove objects from the storage

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные на локальном хранилище

Описание ошибки:

Сервис Image Store не смог удалить несколько объектов с диска.

Следует проверить логи сервиса Image Store. Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с диском, проблема в сервисе Image Store.

11.10.10 Вернулся код ошибки 19010

Текст ошибки:

Internal server error Failed to get object from the storage

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные на локальном хранилище

Описание ошибки:

Сервис Image Store не смог получить объект с диска.

Следует проверить логи сервиса Image Store. Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с диском, проблема в сервисе Image Store.

11.10.11 Вернулся код ошибки 19011

Текст ошибки:

Internal server error Failed to get objects from the storage

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные на локальном хранилище

Описание ошибки:

Сервис Image Store не смог получить несколько объектов с диска.

Следует проверить логи сервиса Image Store. Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с диском, проблема в сервисе Image Store.

11.10.12 Вернулся код ошибки 19012

Текст ошибки:

Internal server error Failed to delete bucket

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные на локальном хранилище

Описание ошибки:

Сервис Image Store не смог удалить бакет с диска.

Следует проверить логи сервиса Image Store. Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с диском, проблема в сервисе Image Store.

11.10.13 Вернулся код ошибки 19013

Текст ошибки:

Internal server error Failed to get bucket info

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные на локальном хранилище

Описание ошибки:

Сервис Image Store не смог получить информацию о бакете.

Следует проверить логи сервиса Image Store. Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с диском, проблема в сервисе Image Store.

11.11 Ошибки сервиса Image Store, хранящего данные в хранилище S3

11.11.1 Вернулся код ошибки 20001

Текст ошибки:

Internal server error Failed to save image in the storage

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные в хранилище S3

Описание ошибки:

Сервис Image Store не смог сохранить изображение в хранилище Image Store (S3).

Следует проверить логи сервиса Image Store. Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с хранилищем, проблема в сервисе Image Store.

11.11.2 Вернулся код ошибки 20002

Текст ошибки:

Internal server error Failed to remove image from the storage

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные в хранилище S3

Описание ошибки:

Сервис Image Store не смог удалить изображение из хранилища Image Store (S3).

Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с хранилищем, проблема в сервисе Image Store.

11.11.3 Вернулся код ошибки 20003

Текст ошибки:

Internal server error Failed to remove image from the storage

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные в хранилище S3

Описание ошибки:

Сервис Image Store не смог удалить изображения из хранилища Image Store (S3).

Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с хранилищем, проблема в сервисе Image Store.

11.11.4 Вернулся код ошибки 20004

Текст ошибки:

Internal server error Failed to create bucket

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные в хранилище S3

Описание ошибки:

Сервис Image Store не смог создать бакет в хранилище Image Store (S3).

Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с хранилищем, проблема в сервисе Image Store.

11.11.5 Вернулся код ошибки 20005

Текст ошибки:

Internal server error Request to S3 Failed

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные в хранилище S3

Описание ошибки:

Не удалось выполнить запрос к хранилищу Image Store (S3).

Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с хранилищем, проблема в сервисе Image Store.

11.11.6 Вернулся код ошибки 20006

Текст ошибки:

Internal server error Request to S3 Forbidden

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные в хранилище S3

Описание ошибки:

Не удалось получить доступ к хранилищу Image Store (S3).

Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с хранилищем, проблема в сервисе Image Store.

11.11.7 Вернулся код ошибки 20007

Текст ошибки:

Internal server error Request time to S3 is longer than the established time

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные в хранилище S3

Описание ошибки:

Время запроса к хранилищу Image Store (S3) превысило установленный лимит.

Следует проверить доступность хранилища. Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с хранилищем, проблема в в сервисе Image Store.

11.11.8 Вернулся код ошибки 20008

Текст ошибки:

Internal server error S3 Connection Refused

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные в хранилище S3

Описание ошибки:

Отказано в подключении к хранилищу Image Store (S3).

Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с хранилищем, проблема в сервисе Image Store.

11.11.9 Вернулся код ошибки 20009

Текст ошибки:

Internal server error Connect time to S3 is longer than the established time

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные в хранилище S3

Описание ошибки:

Время подключения к хранилищу Image Store (S3) превысило установленный лимит.

Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с хранилищем, проблема в сервисе Image Store.

11.11.10 Вернулся код ошибки 20010

Текст ошибки:

Internal server error Unknown s3 error

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные в хранилище S3

Описание ошибки:

Неизвестная ошибка хранилища Image Store (S3).

Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с хранилищем, проблема в сервисе Image Store.

11.11.11 Вернулся код ошибки 20011

Текст ошибки:

Internal server error Failed to get bucket list

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные в хранилище S3

Описание ошибки:

Сервис Image Store не смог получить список бакетов из хранилища Image Store (S3).

Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с хранилищем, проблема в сервисе Image Store.

11.11.12 Вернулся код ошибки 20012

Текст ошибки:

Internal server error Failed to get image from the storage

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные в хранилище S3

Описание ошибки:

Сервис Image Store не смог получить изображение из хранилища Image Store (S3).

Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с хранилищем, проблема в сервисе Image Store.

11.11.13 Вернулся код ошибки 20013

Текст ошибки:

Internal server error Failed to save object in the storage

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные в хранилище S3

Описание ошибки:

Сервис Image Store не смог сохранить объект в хранилище Image Store (S3).

Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с хранилищем, проблема в сервисе Image Store.

11.11.14 Вернулся код ошибки 20014

Текст ошибки:

Internal server error Failed to remove object from the storage

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные в хранилище S3

Описание ошибки:

Сервис Image Store не смог удалить объект из хранилища Image Store (S3).

Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с хранилищем, проблема в сервисе Image Store.

11.11.15 Вернулся код ошибки 20015

Текст ошибки:

Internal server error Failed to remove object list from the storage

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные в хранилище S3

Описание ошибки:

Сервис Image Store не смог удалить Список объектов из хранилища Image Store (S3).

Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с хранилищем, проблема в сервисе Image Store.

11.11.16 Вернулся код ошибки 20016

Текст ошибки:

Internal server error Failed to get object from the storage

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные в хранилище S3

Описание ошибки:

Сервис Image Store не смог получить объект из хранилища Image Store (S3).

Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с хранилищем, проблема в сервисе Image Store.

11.11.17 Вернулся код ошибки 20017

Текст ошибки:

Internal server error Failed to get object list from the storage

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные в хранилище S3

Описание ошибки:

Сервис Image Store не смог получить несколько объектов из хранилища Image Store (S3).

Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с хранилищем, проблема в сервисе Image Store.

11.11.18 Вернулся код ошибки 20018

Текст ошибки:

Internal server error Failed to delete bucket

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные в хранилище S3

Описание ошибки:

Сервис Image Store не смог удалить бакет из хранилища Image Store (S3).

Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с хранилищем, проблема в сервисе Image Store.

11.11.19 Вернулся код ошибки 20019

Текст ошибки:

Bad/incomplete query Failed to create bucket with specified name

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные в хранилище S3

Описание ошибки:

Бакет с данным именем не может быть создан. Имя содержит недопустимые символы. Такое имя не может быть использовано в качестве имени бакета.

11.11.20 Вернулся код ошибки 20020

Текст ошибки:

Internal server error Failed to get bucket info

Источник ошибки:

Ошибки сервиса Image Store, хранящего данные в хранилище S3

Описание ошибки:

Сервис Image Store не смог получить данные бакета.

Следует проверить логи сервиса Image Store. Причиной ошибки может быть: отказ в доступе, недоступность сервиса по сети, проблема с диском, проблема в сервисе Image Store.

11.12 Ошибки сервиса Faces

11.12.1 Вернулся код ошибки 22001

Текст ошибки:

Unique constraint error Face with the same attribute_id already exists

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

Лицо с указанным в запросе «attribute_id» уже существует. Атрибут может быть прикреплен только к одному лицу.

11.12.2 Вернулся код ошибки 22002

Текст ошибки:

Object not found Face with id «{value}» not found

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

Лицо с указанным в запросе «face_id» не найдено. В поле «face_id» допущена ошибка или лицо с таким ID не существует.

11.12.3 Вернулся код ошибки 22003

Текст ошибки:

Object not found List with id «{value}» not found

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

Список с данным ID не найден. Необходимо указать в запросе существующий список.

11.12.4 Вернулся код ошибки 22004

Текст ошибки:

Object not found One or more faces not found including face with id «{value}»

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

Одно или более лицо не найдено, включая лицо с указанным ID. В поле «face_id» допущена ошибка или лицо с таким ID не существует.

11.12.5 Вернулся код ошибки 22005

Текст ошибки:

Object not found One or more lists not found including list with id «{value}»

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

Один или более списков не найдены, включая список с указанным ID. Указанные списки не существуют, необходимо указать в запросе существующий список.

11.12.6 Вернулся код ошибки 22009

Текст ошибки:

Bad/incomplete configuration Face avatar url is not correct

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

URL в поле «avatar» в запросе задан неверно. Проверьте переданный URL. URL не содержит данных, поврежден или данные недоступны без авторизации.

11.12.7 Вернулся код ошибки 22010

Текст ошибки:

Object not found Attributes with id «{value}» for update not found

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

Ошибка при обновлении полей существующего Лица. В запросе неверно задано поле «attribute_-id». Атрибут с указанным ID не найден.

Атрибут удаляется по истечении срока действия его существования (TTL). Создайте новый атрибут для прикрепления к лицу.

См. информацию о создании атрибутов в разделе Объект «Атрибут».

11.12.8 Вернулся код ошибки 22011

Текст ошибки:

Object not found Attributes with id {value} not found

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

Атрибуты с указанными ID не найдены в БД. Следует указать существующий «attribute_id» в запросе.

Атрибут удаляется по истечении срока действия его существования (TTL). Создайте новый атрибут для прикрепления к лицу.

См. информацию о создании атрибутов в разделе Объект «Атрибут».

11.12.9 Вернулся код ошибки 22012

Текст ошибки:

Bad/incomplete input data Failed to decode descriptor from base64

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

Не удалось декодировать биометрический шаблон из формата BASE64.

Если ошибка повторяется, проверьте файл BASE64. Он может быть поврежден.

11.12.10 Вернулся код ошибки 22013

Текст ошибки:

Bad/incomplete input data Failed to encode descriptor to base64

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

Не удалось кодировать биометрический шаблон в формат BASE64. Произошла ошибка ПО.

11.12.11 Вернулся код ошибки 22015

Текст ошибки:

Conflict input data Attribute «{value}» generation should be «{value}» but «{value}» was provided

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

Поколение атрибута не совпадает с переданным.

Ошибка возникает при извлечении биометрического шаблона новой версии. Исходный биометрический шаблон был обновлён, поэтому в БД нельзя добавить биометрический шаблон новой версии. Требуется повторно извлечь биометрический шаблон, используя исходное изображение и требуемую версию нейронной сети.

См. раздел «Задача Additional extraction» для получении информации о задаче Additional extraction.

См. раздел «Изменение используемой модели нейросети» для получения информации о процессе переключения версии нейронной сети.

11.12.12 Вернулся код ошибки 22016

Текст ошибки:

Bad input data «{value}» is not valid target to get faces. Valid target should be one of {value}.

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

Недействительный «target» для получения Лиц.

Корректный «target» должен быть одним из указанных в этой ошибке.

См. документацию API для запроса.

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.12.13 Вернулся код ошибки 22017

Текст ошибки:

Bad/incomplete input data Match reference must be specified either as (descriptor) or (attribute_id) or (face_id)

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

Ссылка для сравнения должна быть указана как биометрический шаблон, идентификатор атрибута или идентификатор лица.

11.12.14 Вернулся код ошибки 22018

Текст ошибки:

Object not found Descriptor of version {value} is not found for object with id «{value}».

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

Для объекта с указанным идентификатором нет биометрического шаблона с указанной версией.

Предоставленный объект может включать в себя несколько биометрических шаблонов разных версий. Но нет БШ версии, используемого в системе.

См. версию БШ по умолчанию в параметре «DEFAULT_FACE_DESCRIPTOR_VERSION» в сервисе Configurator или в файлах конфигурации сервисов LP.

11.12.15 Вернулся код ошибки 22020

Текст ошибки:

Internal server error Corrupted attribute with id «{value}»

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

Атрибут с этим идентификатором поврежден. Причина неизвестна.

11.12.16 Вернулся код ошибки 22021

Текст ошибки:

Integrity error Attribute with id «{value}» already exist

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

Атрибут с указанным идентификатором уже существует.

11.12.17 Вернулся код ошибки 22022

Текст ошибки:

Bad input data Attribute does not contain «descriptors» and «basic_attributes»

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

Входящий атрибут не содержит никаких данных: биометрических шаблонов или базовых атрибутов.

11.12.18 Вернулся код ошибки 22023

Текст ошибки:

Bad input data Attribute contains «{value}» samples but corresponding attributes is empty

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

Атрибут содержит биометрические образцы без данных.

11.12.19 Вернулся код ошибки 22024

Текст ошибки:

Bad input data Attribute contains descriptors of identical versions: «{value}»

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

Атрибут содержит биометрические шаблоны одной версии. В одном атрибуте не может быть двух биометрических шаблонов одной и той же версии. Для каждой версии доступен только один БШ.

11.12.20 Вернулся код ошибки 22025

Текст ошибки:

Conflict input data Attribute samples of face (face_id «{value}») do not match specified ones

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

Биометрические образцы атрибутов лица с указанным «face_id» не соответствуют указанным биометрическим образцам.

11.12.21 Вернулся код ошибки 22026

Текст ошибки:

Unique constraint error List with id «{value}» already exist

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

Список с указанным идентификатором уже существует и не может быть создан.

11.12.22 Вернулся код ошибки 22027

Текст ошибки:

Bad input data. «{value}» is not valid target for face deletion info. Valid target should be one of {value}

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

Указанное значение в query-параметре запроса «targets» не совпадает с перечисленными доступными значениями для данного параметра.

Проверьте query-параметр «targets» запроса «delete faces with filters».

11.12.23 Вернулся код ошибки 22028

Текст ошибки:

Bad/incomplete input data«,»Invalid descriptor encryption

Источник ошибки:

Ошибки сервиса Faces

Описание ошибки:

Ошибка возникает при сохранении биометрического шаблона, который был зашифрован другим ключом или с использованием другого алгоритма шифрования.

Сервис ожидает, что биометрический шаблон будет зашифрован определённым ключом и алгоритмом, и если это не так, то возникает ошибка при попытке обработки такого биометрического шаблона.

11.13 Ошибки сервиса Events

11.13.1 Вернулся код ошибки 23001

Текст ошибки:

Object not found Event with id {value} not found

Источник ошибки:

Ошибки сервиса Events

Описание ошибки:

Событие с указанным ID не найдено сервисом Events.

Событие не существует. Оно было удалено или была допущена опечатка в идентификаторе.

11.13.2 Вернулся код ошибки 23002

Текст ошибки:

Object not found One or more events not found including event with id {value}

Источник ошибки:

Ошибки сервиса Events

Описание ошибки:

Одно или несколько событий не были найдены сервисом Events, включая указанное событие. Эти события были удалены или изменены, или в их идентификаторах есть ошибка.

11.13.3 Вернулся код ошибки 23003

Текст ошибки:

Unique constraint error One or more event id from {value} already exist

Источник ошибки:

Ошибки сервиса Events

Описание ошибки:

Один или несколько ID событий из списка уже существуют и не могут быть созданы.

Событие должно иметь уникальное имя.

11.13.4 Вернулся код ошибки 23004

Текст ошибки:

Unique constraint error One or more attribute id from «{value}» already exist

Источник ошибки:

Ошибки сервиса Events

Описание ошибки:

Один или более из указанных Attribute ID уже существует.

11.13.5 Вернулся код ошибки 23005

Текст ошибки:

Bad input data «{value}» is not valid target to get events. Valid events should be one of {value}

Источник ошибки:

Ошибки сервиса Events

Описание ошибки:

Указанный «target» недействителен. Укажите один из доступных.

См. документацию API для запроса.

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.13.6 Вернулся код ошибки 23006

Текст ошибки:

Internal server error Timeout ({value} seconds) for saving events into history database has been expired

Источник ошибки:

Ошибки сервиса Events

Описание ошибки:

Истекло время ожидания для сохранения событий в базе данных. Запрос занял слишком много времени из-за большого количества переданных данных или проблем с подключением к базе данных.

См. раздел «Продвинутая настройка PostgreSQL» для получения информации о настройке СУБД PostgreSQL.

11.13.7 Вернулся код ошибки 23007

Текст ошибки:

Object not found Human descriptor of version {value} is not found for object with id «{value}».

Источник ошибки:

Ошибки сервиса Events

Описание ошибки:

Биометрический шаблон для объекта с указанным идентификатором не был найден. БШ не был извлечен или был удален.

11.13.8 Вернулся код ошибки 23008

Текст ошибки:

Internal server error Copywriter queue is full

Источник ошибки:

Ошибки сервиса Events

Описание ошибки:

Очередь копирайтера заполнена.

11.13.9 Вернулся код ошибки 23009

Текст ошибки:

Internal server error Events are shutting down

Источник ошибки:

Ошибки сервиса Events

Описание ошибки:

Сервис Events завершает работу. Проверьте логи сервиса Events. В сервисе произошла ошибка.

11.13.10 Вернулся код ошибки 23010

Текст ошибки:

Internal server error Events saving failed reason: {value}

Источник ошибки:

Ошибки сервиса Events

Описание ошибки:

Сохранение события не удалось из-за внутренней ошибки сервера.

Рекомендуется проверить логи сервиса для получения более подробной информации об ошибке.

Если предоставлена какая-либо дополнительная трассировка и ошибка продолжает появляться, отправьте трассировку в службу технической поддержки VisionLabs.

11.13.11 Вернулся код ошибки 23011

Текст ошибки:

Bad input data. «{value}» is not valid target for event deletion info. Valid target should be one of {value}

Источник ошибки:

Ошибки сервиса Events

Описание ошибки:

Указано недопустимое значение в поле «target» в параметрах запроса. Допустимое значение должно быть одним из указанных.

Проверьте установленное значение для поля «target».

11.13.12 Вернулся код ошибки 23012

Текст ошибки:

Bad input data. Meta length exceeds 2Mb

Источник ошибки:

Ошибки сервиса Events

Описание ошибки:

Слишком большой размер метаинформации. Размер не должен превышать 2 МБ.

11.13.13 Вернулся код ошибки 23013

Текст ошибки:

Bad input data. Event content exceeds {value}

Источник ошибки:

Ошибки сервиса Events

Описание ошибки:

Слишком большой размер общего события. Размер не должен превышать 2 МБ.

11.14 Ошибки сервиса Configurator

11.14.1 Вернулся код ошибки 27001

Текст ошибки:

Object not found Setting with id «{value}» not found

Источник ошибки:

Ошибки сервиса Configurator

Описание ошибки:

Параметр с указанным ID не найден. Проверьте корректность указанного идентификатора и наличие настройки в сервисе Configurator.

11.14.2 Вернулся код ошибки 27002

Текст ошибки:

Integrity error Setting with the following fields already exists: name: {value} tag: {value}

Источник ошибки:

Ошибки сервиса Configurator

Описание ошибки:

Параметр с указанными полями уже существует. Пара «name» и «tag» в БД должна быть уникальна.

11.14.3 Вернулся код ошибки 27003

Текст ошибки:

Bad/incomplete input data Connection check to service is failed

Источник ошибки:

Ошибки сервиса Configurator

Описание ошибки:

Запрос к сервису выполнен с ошибкой. Убедитесь, что сервис запущен, в ее логах нет ошибок и сервис доступен по сети.

11.14.4 Вернулся код ошибки 27004

Текст ошибки:

Bad/incomplete input data Not allowed to change tags for default setting

Источник ошибки:

Ошибки сервиса Configurator

Описание ошибки:

Нельзя создавать tag для настройки по умолчанию. У настройки по умолчанию не может быть tag. Продублируйте параметр, чтобы изменить его и применить tag.

11.14.5 Вернулся код ошибки 27005

Текст ошибки:

Bad/incomplete input data Tag «{value}» for setting «{value}» not found

Источник ошибки:

Ошибки сервиса Configurator

Описание ошибки:

Для указанной настройки отсутствует указанный tag.

11.14.6 Вернулся код ошибки 27006

Текст ошибки:

Object not found Limitation named «{value}» not found

Источник ошибки:

Ошибки сервиса Configurator

Описание ошибки:

Новая созданная настройка должна соответствовать заданному для неё шаблону настройки (limitation).

Описание шаблона настройки приводится в пользовательском интерфейсе и dump-файле. Файл можно получить с помощью запроса «GET Dump file» к сервису Configurator.

11.14.7 Вернулся код ошибки 27007

Текст ошибки:

Integrity error Limitation named «{value}» already exists

Источник ошибки:

Ошибки сервиса Configurator

Описание ошибки:

При создании нового шаблона для настройки (limitation) использовано имя уже существующего шаблона настройки. Имя должно быть уникальным.

11.14.8 Вернулся код ошибки 27008

Текст ошибки:

Object not found Group named «{value}» not found

Источник ошибки:

Ошибки сервиса Configurator

Описание ошибки:

Объект Group с указанным именем не был найден.

Вы можете создать новую группу с помощью пользовательского интерфейса Configurator или запроса «New group» к сервису Configurator.

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.14.9 Вернулся код ошибки 27009

Текст ошибки:

Object not found One or more groups not found including group named «{value}»

Источник ошибки:

Ошибки сервиса Configurator

Описание ошибки:

Один или несколько объектов Group не были найдены, включая объект с указанным именем.

Вы можете создать новую группу с помощью пользовательского интерфейса Configurator или запроса «New group» к сервису Configurator.

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.14.10 Вернулся код ошибки 27010

Текст ошибки:

Integrity error Group named «{value}» already exists

Источник ошибки:

Ошибки сервиса Configurator

Описание ошибки:

Объект Group с указанным именем уже существует.

11.14.11 Вернулся код ошибки 27011

Текст ошибки:

Bad/incomplete input data Cannot remove default setting «{value}»

Источник ошибки:

Ошибки сервиса Configurator

Описание ошибки:

Невозможно удалить настройку по умолчанию из Configurator.

11.14.12 Вернулся код ошибки 27012

Текст ошибки:

Bad/incomplete input data Not allowed to change default setting name

Источник ошибки:

Ошибки сервиса Configurator

Описание ошибки:

Ошибка при попытке изменения имени настройки по умолчанию. Изменение имени настройки по умолчанию запрещено.

Продублируйте параметр, чтобы изменить его и применить новое имя.

11.15 Ошибки сервиса Tasks

11.15.1 Вернулся код ошибки 28000

Текст ошибки:

Network error Cannot send subtasks to task_workers. Reason: «{value}»

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Не удалось отправить подзадачи «рабочим процессам» сервиса Tasks. Причина указана в поле «Reason».

Убедитесь, что сервис «рабочих процессов» запущен и нет проблем с сетью.

11.15.2 Вернулся код ошибки 28001

Текст ошибки:

Object not found Task with id «{value}» not found

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Задача с указанным ID не найдена сервисом Tasks.

Задача была завершена, ID был задан некорректно или задача с заданным ID была удалена.

11.15.3 Вернулся код ошибки 28002

Текст ошибки:

Object not found One or more tasks not found including task with id {value}

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Сервис Tasks не обнаружил указанные задачи, в том числе задачи с указанным ID. ID задан неверно, либо задача была удалена.

11.15.4 Вернулся код ошибки 28003

Текст ошибки:

Object not found Task error with id «{value}» not found

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Ошибка с указанным в запросе ID не найдена сервисом Tasks. Ошибка не существует.

11.15.5 Вернулся код ошибки 28004

Текст ошибки:

Object not found One or more tasks not found including task with id «{value}»

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Одна или несколько ошибок не найдены сервисом Tasks, в том числе с указанным ID.

ID задан неверно, либо ошибка была удалена.

11.15.6 Вернулся код ошибки 28005

Текст ошибки:

Stop tasks worker On worker shutdown all active tasks on the worker are failed all active subtasks are cancelled

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Процесс выполнения задачи был прерван потому что «рабочий процесс» Tasks был остановлен. Все активные подзадачи были отменены.

Следует проверить статус всех «рабочих процессов» сервиса Tasks. Какой-то из «рабочих процессов» стал недоступен по некоторой причине. Могли возникнуть проблемы с сервисом, сетью или сервером.

11.15.7 Вернулся код ошибки 28006

Текст ошибки:

Internal server error Failed to update task {value} status. Desired status: {value}

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Сервису Tasks не удалось обновить статус указанной задачи. Показан предполагаемый статус.

11.15.8 Вернулся код ошибки 28007

Текст ошибки:

Internal server error Failed to update subtask {value} status. Desired status: {value}

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Сервису Tasks не удалось обновить статус подзадачи. Показан предполагаемый статус.

11.15.9 Вернулся код ошибки 28008

Текст ошибки:

Internal server error Failed to update task {value} progress. Desired progress: {value}

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Сервису Tasks не удалось обновить прогресс задачи. Показан предполагаемый статус.

11.15.10 Вернулся код ошибки 28009

Текст ошибки:

Attribute is not equal Event ({value}) attribute is not equal {value} corresponding face attribute

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

ID атрибута события и ID соответствующего атрибута лица не совпали (задача Linker).

11.15.11 Вернулся код ошибки 28010

Текст ошибки:

Objects not found Objects for clustering not found (empty set)

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Указанные для кластеризации объекты не найдены сервисом Tasks. Проверьте, что объекты существуют и их идентификаторы корректны.

11.15.12 Вернулся код ошибки 28011

Текст ошибки:

Internal server error Failed to save report to a csv-file

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Сервису Tasks не удалось создать CSV файл для отчёта.

11.15.13 Вернулся код ошибки 28012

Текст ошибки:

Internal server error Failed to create archive from a report

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Сервису Tasks не удалось создать архив для отчёта.

11.15.14 Вернулся код ошибки 28013

Текст ошибки:

Bad/incomplete input data Tasks with type «{value}» does not support a build report

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Для данного типа задачи нельзя создать отчёт (задача Reporter).

11.15.15 Вернулся код ошибки 28014

Текст ошибки:

Bad/incomplete input data Column «{value}» not allowed in report by {value}

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Указанный столбец не может быть добавлен в отчёт (задача Reporter). Столбец недоступен в задаче Reporter.

См. «create reporter task» в спецификации API сервиса API или Tasks.

> Последняя ссылка ведет на последнюю версию документации сервиса Tasks. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.15.16 Вернулся код ошибки 28015

Текст ошибки:

Bad/incomplete input data Tasks with status «{value}» does not support a build report

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Для задач с указанным статусом нельзя создать отчёт (задача Reporter). Следует дождаться выполнения задачи для получения отчёта.

Можно проверить статус задачи с помощью запроса «GET Task»: сервис API или Tasks.

> Последняя ссылка ведет на последнюю версию документации сервиса Tasks. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.15.17 Вернулся код ошибки 28016

Текст ошибки:

Bad/incomplete input data Clusterization task without result does not support a build report

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Отчёт не может быть сгенерирован для задачи кластеризации без результатов. Задача не была завершена, был передан некорректный индентификатор задачи.

11.15.18 Вернулся код ошибки 28017

Текст ошибки:

Object not found Result of the task «{value}» not found

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Результат выполнения указанной задачи не найден.

Возможные причины: результат задачи был удален из сервиса Image Store или произошла ошибка во время выполнения задачи.

11.15.19 Вернулся код ошибки 28018

Текст ошибки:

Object not found Task «{value}» does not have result yet

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Задача ещё не имеет результата. Следует дождаться выполнения задачи.

Можно проверить статус задачи с помощью запроса «GET Task»: сервис API или Tasks.

> Последняя ссылка ведет на последнюю версию документации сервиса Tasks. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.15.20 Вернулся код ошибки 28019

Текст ошибки:

Bad/incomplete input data Task «{value}» with status {value} cannot be canceled

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Задача с указанным статусом не может быть отменена.

В зависимости от статуса задача могла быть остановлена, завершена с ошибкой или успешно выполнена.

11.15.21 Вернулся код ошибки 28020

Текст ошибки:

Object not found Impossible get a result of task {value} with status {value}

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Невозможно получить результат выполнения задачи сервиса Tasks с указанным статусом.

11.15.22 Вернулся код ошибки 28021

Текст ошибки:

Forbidden Service does not support Luna Events as source for tasks

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Сервис не поддерживает Events как источник данных для задач.

11.15.23 Вернулся код ошибки 28022

Текст ошибки:

Objects not found {value} for cross-matching not found (empty set)

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Указанный объект для перекрёстного сравнения (cross-matching) не найден.

Объект был удален или в его идентификаторе допущена опечатка.

11.15.24 Вернулся код ошибки 28023

Текст ошибки:

Reference uuid is missing: {value}

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Объект с ID, указанным в качестве эталона для задачи Cross-matching, не был найден.

Объект был удален или в его идентификаторе допущена опечатка.

11.15.25 Вернулся код ошибки 28024

Текст ошибки:

Reference uuid has no extracted descriptor: {value}

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Отсутствует извлечённый биометрический шаблон для объекта с UUID, указанным для задачи Cross-matching.

11.15.26 Вернулся код ошибки 28025

Текст ошибки:

Task «{value}» has been cancelled

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Задача была отменена, и для нее нет результата.

11.15.27 Вернулся код ошибки 28026

Текст ошибки:

Failed to read archive {value}

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Не удалось прочесть указанный архив.

Проверьте доступность архива.

11.15.28 Вернулся код ошибки 28027

Текст ошибки:

Failed to read file {value}

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Не удалось прочесть указанный файл.

Проверьте доступность файла.

11.15.29 Вернулся код ошибки 28028

Текст ошибки:

Failed to read directory {value}

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Не удалось прочесть указанную директорию.

Проверьте доступность директории.

11.15.30 Вернулся код ошибки 28029

Текст ошибки:

Network disk is not available {value}

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Указанный сетевой диск недоступен.

Проверьте доступность сетевого диска.

11.15.31 Вернулся код ошибки 28030

Текст ошибки:

FTP server authorization error {value}

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Произошла ошибка авторизации при подключении к указанному FTP-серверу.

Проверьте параметры «user» и «password».

11.15.32 Вернулся код ошибки 28031

Текст ошибки:

FTP server is unreachable {value}

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

FTP-сервер недоступен.

Проверьте доступность FTP-сервера.

11.15.33 Вернулся код ошибки 28032

Текст ошибки:

FTP download error {value}

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Произошла ошибка при загрузке данных с FTP-сервера.

Проверьте введенные параметры запроса. ### Вернулся код ошибки 28033 {#code-28033-returned}

Текст ошибки:

FTP file listing error {value}

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Произошла ошибка получения списка файлов с FTP-сервера.

Проверьте введенные параметры запроса.

11.15.34 Вернулся код ошибки 28034

Текст ошибки:

Samba authorization error, {value}

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Ошибка авторизации указанной Samba.

Проверьте введенные параметры запроса.

11.15.35 Вернулся код ошибки 28035

Текст ошибки:

Samba is unreachable, {value}

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Samba по указанным данным недоступна.

Проверьте введенные параметры запроса.

11.15.36 Вернулся код ошибки 28036

Текст ошибки:

Samba download error, {value}

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Ошибка загрузки для указанной Samba.

Проверьте введенные параметры запроса.

11.15.37 Вернулся код ошибки 28037

Текст ошибки:

Samba unknown error, {value}

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Неизвестная ошибка указанной Samba.

11.15.38 Вернулся код ошибки 28038

Текст ошибки:

Exporter failed to download data, {value}

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Произошла ошибка при загрузке данных во время выполнения указанной задачи Exporter.

Данная ошибка может возникнуть, например, если сервис Faces или Events вернул ошибку.

11.15.39 Вернулся код ошибки 28039

Текст ошибки:

Exporter repeatedly failed to download data, {value}

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Произошли неоднократные ошибки при загрузке данных во время выполнения указанной задачи Exporter.

Данная ошибка может возникнуть, например, если при выполнении задачи Exporter периодически не удавалось вовремя соединиться с сервисом Faces или Events и экспортировались не все данные.

11.15.40 Вернулся код ошибки 28040

Текст ошибки:

Object not found, Schedule with id {value} not found

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Не найдено расписание с указанным идентификатором.

Проверьте корректность введенного идентификатора.

Можно получить список всех расписаний с помощью запроса «get tasks schedules».

11.15.41 Вернулся код ошибки 28041

Текст ошибки:

Encryption error, Authorization data is incorrect

Источник ошибки:

Ошибки сервиса Tasks

Описание ошибки:

Ошибка шифрования существующих паролей и токенов, передаваемых в задаче Estimator или в политике «notification_policy».

Убедитесь, что переменные окружения `FERNET_PASSPHRASE` и `SALT` были указаны во время миграции БД Tasks и во время запуска контейнера сервиса Tasks.

См. подробную информацию в разделе «Дополнительная защита паролей и токенов» в руководстве администратора.

11.16 Ошибки сервиса Sender

11.16.1 Вернулся код ошибки 29001

Текст ошибки:

Forbidden Cannot subscribe for events without «Luna-Account-Id» header set

Источник ошибки:

Ошибки сервиса Sender

Описание ошибки:

В запросе не указан заголовок «Luna-Account-Id», необходимый для подписки на события сервиса Sender.

Следует использовать тот же заголовок «Luna-Account-Id», что и для запроса на создание событий «generate events».

См. раздел «HEADER PARAMETERS» ресурса «/ws»:

- сервис API
- сервис Sender

> Последняя ссылка ведет на последнюю версию документации сервиса Sender. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.16.2 Вернулся код ошибки 29002

Текст ошибки:

Internal server error Failed to subscribe to chanel {value}

Источник ошибки:

Ошибки сервиса Sender

Описание ошибки:

Сервису Sender не удалось подписаться на канал.

11.16.3 Вернулся код ошибки 29003

Текст ошибки:

Internal server error Failed to process input message

Источник ошибки:

Ошибки сервиса Sender

Описание ошибки:

Сервису Sender не удалось обработать входящее сообщение.

Проверьте, что входящее изображение не повреждено и имеет правильный формат.

11.16.4 Вернулся код ошибки 29004

Текст ошибки:

Internal server error Failed to publish message to ws

Источник ошибки:

Ошибки сервиса Sender

Описание ошибки:

Сервису Sender не удалось опубликовать сообщение по веб-сокету.

11.16.5 Вернулся код ошибки 29005

Текст ошибки:

Service Unavailable Redis disconnected

Источник ошибки:

Ошибки сервиса Sender

Описание ошибки:

Служба Sender недоступна. БД Redis была отключена. Проверьте, что Redis запущен и работает, а также доступен через сеть.

11.17 Ошибки сервиса Python Matcher

11.17.1 Вернулся код ошибки 31000

Текст ошибки:

Bad/incomplete input data Account id from query parameters does not match the one from {value} filters

Источник ошибки:

Ошибки сервиса Python Matcher

Описание ошибки:

ID учетной записи, указанный в параметрах запроса, не соответствует ID учетной записи из указанных фильтров. Данные принадлежат разным ID учетной записи.

Измените ID учетной записи в запросе на требуемый идентификатор.

11.17.2 Вернулся код ошибки 31001

Текст ошибки:

Bad/incomplete input data Descriptor has the version ({value}) which does not match with version ({value}) which is supposed to use for the matching

Источник ошибки:

Ошибки сервиса Python Matcher

Описание ошибки:

Версия обработанного биометрического шаблона не соответствует версии, которая в настоящее время используется для сравнения.

См. версию биометрического шаблона по умолчанию в параметре «DEFAULT_FACE_DESCRIPTOR_-VERSION» в сервисе Configurator или в файлах конфигурации сервисов LP. Обработанные БШ должны быть одной и той же версии.

Необходимо извлечь БШ новой версии, используя исходное изображение, с помощью задачи Additional extraction или путем создания новых биометрических шаблонов.

11.17.3 Вернулся код ошибки 31002

Текст ошибки:

Bad/incomplete input data Cross match filters presume too many received objects. Current general limit — {value}

Источник ошибки:

Ошибки сервиса Python Matcher

Описание ошибки:

Для задачи Cross-matching было отправлено слишком много объектов. Доступный предел показан в описании ошибки.

11.17.4 Вернулся код ошибки 31003

Текст ошибки:

Internal server error Unknown cross matching error

Источник ошибки:

Ошибки сервиса Python Matcher

Описание ошибки:

Произошла неизвестная ошибка перекрёстного сравнения (cross-matching).

11.17.5 Вернулся код ошибки 31005

Текст ошибки:

Bad/incomplete input data Matching between different versions {value} is not allowed

Источник ошибки:

Ошибки сервиса Python Matcher

Описание ошибки:

Сравнение между разными версиями биометрических шаблонов не допускается.

Необходимо извлечь БШ новой версии, используя исходное изображение, с помощью задачи Additional extraction или путем создания новых биометрических шаблонов.

11.17.6 Вернулся код ошибки 31006

Текст ошибки:

Internal server error Unexpected behavior of the «{value}» matcher: {value}

Источник ошибки:

Ошибки сервиса Python Matcher

Описание ошибки:

Непредвиденное поведение заданного матчера.

Ошибка возникает при использовании плагинов. В качестве второго значения отображается ошибка плагина, возникающая из-за того, что автор плагина не учел требования к использованию плагинов.

11.17.7 Вернулся код ошибки 31007

Текст ошибки:

Bad/incomplete input data. Amount of specified/required references exceeds limit: {value}

Источник ошибки:

Ошибки сервиса Python Matcher

Описание ошибки:

Превышено количество заданных/требуемых эталонов.

Проверьте соответствующую настройку в сервисе Python Matcher.

11.17.8 Вернулся код ошибки 31008

Текст ошибки:

Bad/incomplete input data. User with account type user is not allowed to use different account id

Источник ошибки:

Ошибки сервиса Python Matcher

Описание ошибки:

Ошибка входных данных. Аккаунты с типом «user» не могут использовать «account_id», отличный от текущего.

Для возможности использования других «account_id» необходим тип аккаунта «advanced_user» или «admin».

11.17.9 Вернулся код ошибки 31010

Текст ошибки:

Bad input data. Fail to decrypt descriptor.

Источник ошибки:

Ошибки сервиса Python Matcher

Описание ошибки:

Ошибка возникает, когда сервис не смог расшифровать биометрический шаблон из-за неправильных входных данных.

Это может произойти, если пользователь самостоятельно зашифровал биометрический шаблон, но сделал это некорректно, что привело к невозможности биометрический шаблон.

Сервис не осуществляет проверку корректности шифрования биометрического шаблона в момент сохранения, поэтому любые ошибки в самом зашифрованном биометрическом шаблоне могут привести к этой ошибке при попытке его дешифрации.

11.18 Ошибки сервиса Licenses

11.18.1 Вернулся код ошибки 33001

Текст ошибки:

License problem Failed to check HASP License feature {value}

Источник ошибки:

Ошибки сервиса Licenses

Описание ошибки:

Возникла проблема с лицензией. Не удалось получить информацию о доступности одной или нескольких лицензируемых функций LP. Проверьте свою лицензию.

Возможные причины проблемы: лицензия не была добавлена в систему, вы используете неправильную лицензию, не удалось установить соединение с сервером лицензий.

11.18.2 Вернулся код ошибки 33002

Текст ошибки:

License problem Failed to get value of HASP License feature {value}

Источник ошибки:

Ошибки сервиса Licenses

Описание ошибки:

Возникла проблема с лицензией. Не удалось получить значение одной или нескольких лицензируемых функций LP. Проверьте свою лицензию. Возможные причины проблемы: лицензия не была добавлена в систему, вы используете неправильную лицензию, не удалось установить соединение с сервером лицензий.

11.18.3 Вернулся код ошибки 33003

Текст ошибки:

License problem No value found for required HASP License feature {value}

Источник ошибки:

Ошибки сервиса Licenses

Описание ошибки:

Возникла проблема с лицензией. Лицензия не включает в себя требуемую функцию.

Возможные причины проблемы:

- необходимая функция не добавлена в лицензию.
- используется неправильная лицензия. Была применена неправильная лицензия.
- новая лицензия не была применена.
- не удалось установить соединение с сервером лицензий.

11.18.4 Вернулся код ошибки 33004

Текст ошибки:

License problem Failed to consume {value}

Источник ошибки:

Ошибки сервиса Licenses

Описание ошибки:

Невозможно использовать указанную лицензируемую функцию LP, т.к. превышен лимит на количество транзакций. Проверьте свою лицензию.

11.18.5 Вернулся код ошибки 33005

Текст ошибки:

License problem Failed to consume {value}: license expired

Источник ошибки:

Ошибки сервиса Licenses

Описание ошибки:

Невозможно использовать указанную лицензируемую функцию LP, т.к. срок действия лицензии истёк.

Проверьте свою лицензию.

11.18.6 Вернулся код ошибки 33006

Текст ошибки:

Bad input data. «{value}» is not valid target to get license features. Valid target should be one of {values}.

Источник ошибки:

Ошибки сервиса Licenses

Описание ошибки:

Указанное в теле запроса значение не принадлежит ни одному из доступных значений.

См. документацию API для запроса «get license».

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.18.7 Вернулся код ошибки 33007

Текст ошибки:

License problem. Failed to initialize license client {value}.

Источник ошибки:

Ошибки сервиса Licenses

Описание ошибки:

Ошибка лицензии. Не удалось подключиться, либо истекла лицензия.

Проверьте свою лицензию.

11.19 Ошибки сервиса Handlers

11.19.1 Вернулся код ошибки 34000

Текст ошибки:

Object not found Handler with id {value} not found

Источник ошибки:

Ошибки сервиса Handlers

Описание ошибки:

Обработчик с указанным ID не найден.

Возможные причины:

- Неправильный идентификатор учетной записи в запрос. Для указанного идентификатора учетной записи нет обработчика с таким идентификатором.
- Обработчик был удален.
- В идентификаторе обработчика допущена опечатка.

Вы можете создать новый обработчик с помощью запроса «create handler».

11.19.2 Вернулся код ошибки 34001

Текст ошибки:

Forbidden Descriptor version {value} is not supported

Источник ошибки:

Ошибки сервиса Handlers

Описание ошибки:

Биометрический шаблон указанной версии не поддерживается.

Проверьте, что используется корректная версия биометрического шаблона. См. доступные версии биометрических шаблонов в разделе «Нейронные сети».

11.19.3 Вернулся код ошибки 34002

Текст ошибки:

Internal server error Cannot load handler with id «{value}». Handler is corrupted

Источник ошибки:

Ошибки сервиса Handlers

Описание ошибки:

Система не может загрузить обработчик с указанным ID. Обработчик поврежден.

Вы можете создать новый обработчик с помощью запроса «create handler».

11.19.4 Вернулся код ошибки 34003

Текст ошибки:

Bad/incomplete input data Aggregation is not supported for raw descriptors

Источник ошибки:

Ошибки сервиса Handlers

Описание ошибки:

Агрегация не поддерживается для необработанных биометрических шаблонов.

Ошибка возникает, когда в запросе выполняется агрегация, но отправляются биометрические шаблоны вместо изображений. Агрегирование выполняется с использованием изображений.

См. раздел «Агрегация».

11.19.5 Вернулся код ошибки 34004

Текст ошибки:

Object not found Verifier with id {value} not found

Источник ошибки:

Ошибки сервиса Handlers

Описание ошибки:

Верификатор с указанным ID не найден.

Возможные причины:

- В запросе задан неправильный идентификатор учетной записи. Для указанного идентификатора учетной записи не существует верификатора с таким идентификатором.
- Верификатор был удален.
- Была допущена опечатка в идентификаторе верификатора.

Вы можете создать новый верификатор с помощью запроса «create verifier».

11.19.6 Вернулся код ошибки 34005

Текст ошибки:

Bad/incomplete input data Candidates limit exceeded: {value} > {value}

Источник ошибки:

Ошибки сервиса Handlers

Описание ошибки:

Превышен лимит кандидатов. Можно указать ограниченное количество кандидатов.

См. настройку «PLATFORM_LIMITS» в сервисе Configurator или в конфигурационных файлах сервисов.

11.19.7 Вернулся код ошибки 34006

Текст ошибки:

Bad/incomplete input data No candidates specified

Источник ошибки:

Ошибки сервиса Handlers

Описание ошибки:

В запросе не были указаны кандидаты.

См. документацию API для запроса.

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.19.8 Вернулся код ошибки 34007

Текст ошибки:

Forbidden Allowed use only dynamic handler

Источник ошибки:

Ошибки сервиса Handlers

Описание ошибки:

Разрешено использовать только динамический обработчик. Используемый обработчик таковым не является.

Получите текущий обработчик по его идентификатору с помощью запроса «get handlers».

11.19.9 Вернулся код ошибки 34008

Текст ошибки:

Multiple bodies found in image {value} , detect {value}

Источник ошибки:

Ошибки сервиса Handlers

Описание ошибки:

На изображении обнаружено более одного тела и предпринята попытка агрегирования со значением «multiface_policy», отличным от «2». При выполнении агрегирования убедитесь, что значение параметра «multiface_policy» в обработчике установлено равным «2».

11.20 Ошибки сервиса Backport 4

11.20.1 Вернулся код ошибки 35000

Текст ошибки:

Bad/incomplete input data Attributes gc is not available

Источник ошибки:

Ошибки сервиса Backport 4

Описание ошибки:

Атрибуты задачи Garbage collection недоступны.

11.20.2 Вернулся код ошибки 35001

Текст ошибки:

Forbidden Luna Sender service is disabled

Источник ошибки:

Ошибки сервиса Backport 4

Описание ошибки:

Сервис Sender отключён.

Проверьте настройки «ADDITIONAL_SERVICES_USAGE» в сервисе Configurator или в конфигурационном файле.

11.20.3 Вернулся код ошибки 35002

Текст ошибки:

Handler is not supported Invalid handler with id «{value}»

Источник ошибки:

Ошибки сервиса Backport 4

Описание ошибки:

Обработчик не поддерживается. Например, в Backport 4 использовался обработчик с расширенными политиками. Он не может быть обработан сервисом.

См. запросы на создание обработчиков:

- Для сервиса API см. ресурс «/handlers»
- Для сервиса Backport 4 см. ресурс «/handlers»
- Для сервиса Backport 3 см. раздел «Handlers»

> Последние две ссылки ведут на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.21 Ошибки сервиса Backport 3

11.21.1 Вернулся код ошибки 4003

Текст ошибки:

No faces found

Источник ошибки:

Ошибки сервиса Backport 3

Описание ошибки:

Лица не найдены.

11.21.2 Вернулся код ошибки 11002

Текст ошибки:

Authorization failed Account corresponding login/password not found

Источник ошибки:

Ошибки сервиса Backport 3

Описание ошибки:

Для сервиса Backport 3 не была найдена учетная запись. В системе нет учетной записи с указанным логином/паролем.

11.21.3 Вернулся код ошибки 11004

Текст ошибки:

Account is suspended

Источник ошибки:

Ошибки сервиса Backport 3

Описание ошибки:

Учетная запись не активна в сервисе Backport 3.

11.21.4 Вернулся код ошибки 11011

Текст ошибки:

Unique constraint error An account with given email already exists

Источник ошибки:

Ошибки сервиса Backport 3

Описание ошибки:

Учетная запись с указанным адресом электронной почты уже существует.

11.21.5 Вернулся код ошибки 11012

Текст ошибки:

Extract policy error {value}

Источник ошибки:

Ошибки сервиса Backport 3

Описание ошибки:

Во время извлечения произошла ошибка. На изображении есть несколько лиц.

11.21.6 Вернулся код ошибки 11018

Текст ошибки:

Object not found Face descriptor of version {value} is not found for object with id «{value}».

Источник ошибки:

Ошибки сервиса Backport 3

Описание ошибки:

Биометрический шаблон с заданным ID не найден.

Проверьте, что идентификатор введен корректно.

11.21.7 Вернулся код ошибки 11022

Текст ошибки:

Object not found Token not found

Источник ошибки:

Ошибки сервиса Backport 3

Описание ошибки:

Для сервиса Backport 3 не был найден токен.

11.21.8 Вернулся код ошибки 12001

Текст ошибки:

Bad/incomplete input data Object in query is not UUID4 format: xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx

Источник ошибки:

Ошибки сервиса Backport 3

Описание ошибки:

Указанный идентификатор объекта не в формате UUID4. Убедитесь, в корректности переданного идентификатора.

11.21.9 Вернулся код ошибки 12018

Текст ошибки:

Bad/incomplete input data Unsupported param «{value}»

Источник ошибки:

Ошибки сервиса Backport 3

Описание ошибки:

Параметр запроса не поддерживается.

См. документацию API для запроса. Сравните ваши параметры запроса с параметрами из раздела «QUERY PARAMETERS».

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

11.21.10 Вернулся код ошибки 22007

Текст ошибки:

Object not found Person with id «{value}» not found

Источник ошибки:

Ошибки сервиса Backport 3

Описание ошибки:

Человек с данным ID не найден в базе данных.

11.21.11 Вернулся код ошибки 22008

Текст ошибки:

Unique constraint error This face is already attached to the person

Источник ошибки:

Ошибки сервиса Backport 3

Описание ошибки:

Лицо уже прикреплено к человеку.

11.21.12 Вернулся код ошибки 36001

Текст ошибки:

Bad/incomplete input data Expected list of {value}. Got list of {value}. List id «{value}»

Источник ошибки:

Ошибки сервиса Backport 3

Описание ошибки:

Указан неверный тип списка. Убедитесь, что используется список с правильным типом данных. Ошибка обычно возвращается при использовании сервиса Backport 3.

11.21.13 Вернулся код ошибки 36002

Текст ошибки:

Bad/incomplete input data Bad format basic authorization header format: Basic base64(login:password)

Источник ошибки:

Ошибки сервиса Backport 3

Описание ошибки:

Неверный формат основного заголовка авторизации. В заголовке авторизации указан неверный логин или пароль. Ошибка обычно возвращается при использовании сервиса Backport 3.

11.21.14 Вернулся код ошибки 36003

Текст ошибки:

Face does not satisfy thresholds limits No faces found.

Источник ошибки:

Ошибки сервиса Backport 3

Описание ошибки:

Не было найдено лиц, удовлетворяющих указанным пороговым ограничениям. Проверьте пороговые значения, указанные в запросе.

11.21.15 Вернулся код ошибки 36004

Текст ошибки:

Bad/incomplete input data Face was not linked to the person

Источник ошибки:

Ошибки сервиса Backport 3

Описание ошибки:

Лицо не связано с человеком. Ошибка обычно возвращается при использовании сервиса Backport 3.

Вы должны связать лицо с человеком, чтобы выполнить запрос.

11.21.16 Вернулся код ошибки 36005

Текст ошибки:

Forbidden. WS subscription is disabled

Источник ошибки:

Ошибки сервиса Backport 3

Описание ошибки:

Отключена подписка WS.

Проверьте вашу подписку.

11.22 Ошибки сервера приложений

11.22.1 Вернулся код ошибки 37001

Текст ошибки:

Internal server error Service did not process the request within {value} seconds

Источник ошибки:

Ошибки сервера приложений

Описание ошибки:

Сервис не обработал запрос в указанный срок.

Ошибка может возникать из-за:

- высокой загрузки системы.
- нехватки ресурсов сервера.
- проблем с сетью.

11.22.2 Вернулся код ошибки 37002

Текст ошибки:

Request timeout Service did not receive the complete request message within {value} seconds

Источник ошибки:

Ошибки сервера приложений

Описание ошибки:

Сервис не получил полное сообщение с запросом в указанный срок.

Ошибка может возникать из-за:

- высокой загрузки системы.
- нехватки ресурсов сервера.
- проблем с сетью.

11.22.3 Вернулся код ошибки 37003

Текст ошибки:

Request payload is too large

Источник ошибки:

Ошибки сервера приложений

Описание ошибки:

Полезная нагрузка (payload) слишком высока.

11.22.4 Вернулся код ошибки 37004

Текст ошибки:

Internal server error Unknown web application error: {value}

Источник ошибки:

Ошибки сервера приложений

Описание ошибки:

Неизвестная ошибка веб-приложения.

11.22.5 Вернулся код ошибки 37005

Текст ошибки:

Client closed request

Источник ошибки:

Ошибки сервера приложений

Описание ошибки:

Клиентская сторона была отключена.

11.23 Ошибка Healthcheck

11.23.1 Вернулся код ошибки 38001

Текст ошибки:

Health check error {value}

Источник ошибки:

Ошибка Healthcheck

Описание ошибки:

Ошибка работоспособности сервиса.

11.24 Ошибка Cached Matcher

11.24.1 Вернулся код ошибки 40001

Текст ошибки:

Object not found List with id «{value}» not found in the cache

Источник ошибки:

Ошибка Cached Matcher

Описание ошибки:

Список с заданным идентификатором не найден в кэше.

11.25 Ошибки сервиса Accounts

11.25.1 Вернулся код ошибки 41001

Текст ошибки:

Unique constraint error, Account with login «{value}» already exists

Источник ошибки:

Ошибки сервиса Accounts

Описание ошибки:

Аккаунт с указанной электронной почтой уже существует.

11.25.2 Вернулся код ошибки 41002

Текст ошибки:

Unique constraint error, Account with account_id «{value}» already exists

Источник ошибки:

Ошибки сервиса Accounts

Описание ошибки:

Аккаунт с указанным `account_id` уже существует.

11.25.3 Вернулся код ошибки 41003

Текст ошибки:

Object not found, Token with id «{value}» not found

Источник ошибки:

Ошибки сервиса Accounts

Описание ошибки:

Токен с указанным `token_id` не существует.

11.25.4 Вернулся код ошибки 41004

Текст ошибки:

Forbidden, Account with account_id «{value}» has admin type and its type can’t be changed

Источник ошибки:

Ошибки сервиса Accounts

Описание ошибки:

Аккаунт с указанным `account_id` имеет тип аккаунта «admin». Недостаточно прав на изменение такого типа аккаунта.

11.25.5 Вернулся код ошибки 41005

Текст ошибки:

JWT token not found, Specified JWT token doesn’t exist

Источник ошибки:

Ошибки сервиса Accounts

Описание ошибки:

Указанный JWT токен не существует.

11.25.6 Вернулся код ошибки 41006

Текст ошибки:

Account credentials wrong, Account login or password are incorrect

Источник ошибки:

Ошибки сервиса Accounts

Описание ошибки:

Ошибка при верификации учетных данных. Логин или пароль учетной записи неверны.

11.25.7 Вернулся код ошибки 41007

Текст ошибки:

Account credentials wrong, Token has been expired

Источник ошибки:

Ошибки сервиса Accounts

Описание ошибки:

Ошибка при верификации учетных данных. Истекло время действия токена.

Проверьте поле «expiration_time» токена.

11.25.8 Вернулся код ошибки 41008

Текст ошибки:

Bad/incomplete input data, Specified token corrupted

Источник ошибки:

Ошибки сервиса Accounts

Описание ошибки:

Указанный токен поврежден. Проверьте корректность введенного токена.

11.25.9 Вернулся код ошибки 41009

Текст ошибки:

Bad/incomplete input data, Specified permissions («{values}») are unacceptable valid for «{value}» account type

Источник ошибки:

Ошибки сервиса Accounts

Описание ошибки:

Невозможно выдать указанные разрешения аккаунту с указанным типом.

11.25.10 Вернулся код ошибки 41010

Текст ошибки:

Bad input data, «{values}» is not valid target to get account(s). Valid target should be one of {value}.

Источник ошибки:

Ошибки сервиса Accounts

Описание ошибки:

Перечисленные значения для параметра запроса «targets» недоступны для указания в запросе «get account» или «get accounts».

См. список возможных значений в описании ошибки.

11.26 Ошибки сервиса Lambda

11.26.1 Вернулся код ошибки 42001

Текст ошибки:

Object not found, Lambda with id «{value}» not found

Источник ошибки:

Ошибки сервиса Lambda

Описание ошибки:

Lambda с указанным идентификатором не найдена.

Убедитесь, что Lambda была создана.

11.26.2 Вернулся код ошибки 42002

Текст ошибки:

Lambda image creation not found, Lambda image creation pod not found. Lambda id «{value}»

Источник ошибки:

Ошибки сервиса Lambda

Описание ошибки:

Docker-образ для указанной Lambda не найден.

Проверьте логи создания образа с помощью запроса «get lambda image creation logs».

11.26.3 Вернулся код ошибки 42003

Текст ошибки:

Unique constraint error, Lambda with name «{value}» already exists

Источник ошибки:

Ошибки сервиса Lambda

Описание ошибки:

Lambda с указанными именем уже существует. У каждой lambda должно быть уникальное имя.

Удалите старую lambda с помощью запроса «delete lambda» или обновите существующую lambda с помощью запросов «py lambda» или «patch lambda».

11.26.4 Вернулся код ошибки 42004

Текст ошибки:

Lambda exception, «{value}»

Источник ошибки:

Ошибки сервиса Lambda

Описание ошибки:

Ошибка, возникшая в пользовательском модуле.

11.26.5 Вернулся код ошибки 42005

Текст ошибки:

Lambda validation exception, «{value}»

Источник ошибки:

Ошибки сервиса Lambda

Описание ошибки:

Ошибка при валидации Lambda.

Убедитесь, что архив с модулем соответствует требованиям.

11.26.6 Вернулся код ошибки 42006

Текст ошибки:

Lambda update exception, «{value}»

Источник ошибки:

Ошибки сервиса Lambda

Описание ошибки:

Ошибка при обновлении Lambda.

Убедитесь, что архив с модулем соответствует требованиям.

11.26.7 Вернулся код ошибки 42007

Текст ошибки:

Lambda wrong type exception, Lambda type must be «{}» but it is «{}»

Источник ошибки:

Ошибки сервиса Lambda

Описание ошибки:

Неправильный тип lambda.

См. доступные типы в описании ошибки и в описании параметра «parameters» > «lambda_type» запроса «create lambda».

11.26.8 Вернулся код ошибки 42008

Текст ошибки:

Kubernetes exception, {}

Источник ошибки:

Ошибки сервиса Lambda

Описание ошибки:

Указанная ошибка Kubernetes.

11.27 Ошибки сервиса Remote SDK

11.27.1 Вернулся код ошибки 43001

Текст ошибки:

External request failed. Failed to download video by url «{value}», detail «{value}»

Источник ошибки:

Ошибки сервиса Remote SDK

Описание ошибки:

Внешний запрос к указанному видеофайлу не был успешным. Детали ошибки доступны в поле «detail».

Эта ошибка может возникнуть по различным причинам, включая:

- Проблемы с подключением к интернету или серверу, на котором расположено видео. Если сервер недоступен или имеет проблемы с производительностью, запрос может не пройти.
- Ограничения на стороне сервера. Некоторые серверы могут блокировать или ограничивать доступ к видеофайлам для определенных пользователей, IP-адресов или географических регионов.
- Проблемы с правами доступа. Если видео защищено правами доступа и требует авторизации для просмотра или скачивания, запрос может не пройти.

11.27.2 Вернулся код ошибки 43002

Текст ошибки:

Bad/incomplete configuration. Failed to initialize video decoder «{value}»

Источник ошибки:

Ошибки сервиса Remote SDK

Описание ошибки:

Ошибка инициализации видеодекодера на GPU.

11.27.3 Вернулся код ошибки 43003

Текст ошибки:

Bad/incomplete input data. Unsupported video format

Источник ошибки:

Ошибки сервиса Remote SDK

Описание ошибки:

Неподдерживаемый формат видео.

См. полный перечень форматов на CPU в официальной документации FFmpeg.

11.27.4 Вернулся код ошибки 43004

Текст ошибки:

Bad/incomplete input data. Error occurred while decoding video by url «{value}» around «{value}» second

Источник ошибки:

Ошибки сервиса Remote SDK

Описание ошибки:

Ошибка декодирования видео по указанному URL за указанное время.

Эта ошибка может возникнуть по различным причинам, включая неправильные, неполные или поврежденные данные видео.

11.27.5 Вернулся код ошибки 43005

Текст ошибки:

Bad/incomplete input data. Video file by url «{value}» has too large size ({value})

Источник ошибки:

Ошибки сервиса Remote SDK

Описание ошибки:

Видеофайл, расположенный по указанному URL, слишком большой.

Максимальный размер файла задается в настройке «max_size» группы параметров «LUNA_-REMOTE_SDK_VIDEO_SETTINGS» сервиса Remote SDK.

11.27.6 Вернулся код ошибки 43006

Текст ошибки:

Forbidden. Unsupported analytics {value}

Источник ошибки:

Ошибки сервиса Remote SDK

Описание ошибки:

Указанная видеоаналитика не поддерживается.

См. описание запроса «videosdk» в спецификации OpenAPI для получения перечня возможных видеоаналитик.

11.27.7 Вернулся код ошибки 43007

Текст ошибки:

Internal server error. Video analytics processing failed: {value}

Источник ошибки:

Ошибки сервиса Remote SDK

Описание ошибки:

Внутренняя ошибка при выполнении видеоанализа. См. содержание ошибки.

11.27.8 Вернулся код ошибки 43008

Текст ошибки:

Bad/incomplete input data«.»{value}”

Источник ошибки:

Ошибки сервиса Remote SDK

Описание ошибки:

Ошибка валидации параметров, указываемых в теле запроса, либо ошибка при загрузке модуля аналитики из-за того, что модуль не найден или не соответствует протоколу.

Проверьте тело запроса в спецификации OpenAPI.

11.27.9 Вернулся код ошибки 43009

Текст ошибки:

Timeout error. Failed to open video by url {value}, {value}

Источник ошибки:

Ошибки сервиса Remote SDK

Описание ошибки:

Сервис Remote SDK не смог открыть видео по указанному URL из-за истечения времени ожидания.

11.28 Ошибки сервиса Video Manager

11.28.1 Вернулся код ошибки 44001

Текст ошибки:

Object not found. Stream with id {value} not found

Источник ошибки:

Ошибки сервиса Video Manager

Описание ошибки:

Поток с указанным идентификатором не найден.

Можно получить список всех потоков с помощью запроса «get streams».

11.28.2 Вернулся код ошибки 44002

Текст ошибки:

Bad input data. «{value}» is not valid stream status; permitted: {value}.

Источник ошибки:

Ошибки сервиса Video Manager

Описание ошибки:

Указанный статус не является подходящим. В ответе предоставлен список возможных статусов.

11.28.3 Вернулся код ошибки 44003

Текст ошибки:

Unique constraint error. Group named «{value}» already exists

Источник ошибки:

Ошибки сервиса Video Manager

Описание ошибки:

Группа с указанным именем уже существует.

Список всех созданных групп можно получить с помощью запроса «get groups».

11.28.4 Вернулся код ошибки 44004

Текст ошибки:

Object not found. Group with id {value} not found

Источник ошибки:

Ошибки сервиса Video Manager

Описание ошибки:

Группа с указанным идентификатором не найдена.

Список всех созданных групп можно получить с помощью запроса «get groups».

11.28.5 Вернулся код ошибки 44005

Текст ошибки:

Object not found. Group named «{value}» not found

Источник ошибки:

Ошибки сервиса Video Manager

Описание ошибки:

Группа с указанным именем не найдена.

Список всех созданных групп можно получить с помощью запроса «get groups».

11.28.6 Вернулся код ошибки 44006

Текст ошибки:

Bad input data. «{value}» is not valid stream log target; permitted: {value}.

Источник ошибки:

Ошибки сервиса Video Manager

Описание ошибки:

Указанное значение в поле «targets» в запросе на получение логов не подходит под перечень доступных значений. В ответе предоставлен список возможных значений.

11.28.7 Вернулся код ошибки 44007

Текст ошибки:

Object not found. Agent with id {value} not found

Источник ошибки:

Ошибки сервиса Video Manager

Описание ошибки:

Агент с указанным идентификатором не найден.

Список всех созданных агентов можно получить с помощью запроса «get agents» к сервису Video Manager. См. подробную информацию в спецификации сервиса Video Manager.

11.28.8 Вернулся код ошибки 44008

Текст ошибки:

Object not found. Analytic with id {value} not found

Источник ошибки:

Ошибки сервиса Video Manager

Описание ошибки:

Аналитика с указанным идентификатором не найдена.

Список всех созданных аналитик можно получить с помощью запроса «get analytics».

11.28.9 Вернулся код ошибки 44009

Текст ошибки:

Unique constraint error. Analytic with name «{value}» already exists

Источник ошибки:

Ошибки сервиса Video Manager

Описание ошибки:

Аналитика с указанным именем уже существует.

Список всех созданных аналитик можно получить с помощью запроса «get analytics».

11.28.10 Вернулся код ошибки 44010

Текст ошибки:

Unable to stop processing. Unable to stop processing, because stream processing is not in progress

Источник ошибки:

Ошибки сервиса Video Manager

Описание ошибки:

Невозможно остановить обработку потока, т.к. его обработка еще не была начата.

Получить статус обработки потока можно с помощью запроса «get stream».

Дождитесь начала обработки потока и повторите попытку.

11.28.11 Вернулся код ошибки 44011

Текст ошибки:

Unable to resume processing. Unable to resume processing, because stream processing is not stopped

Источник ошибки:

Ошибки сервиса Video Manager

Описание ошибки:

Невозможно восстановить обработку потока, т.к. его обработка еще не была остановлена.

Получить статус обработки потока можно с помощью запроса «get stream».

Дождитесь остановки обработки потока и повторите попытку.

11.28.12 Вернулся код ошибки 44012

Текст ошибки:

Unable to stop processing. Stream processing does not stopped yet

Источник ошибки:

Ошибки сервиса Video Manager

Описание ошибки:

Невозможно остановить обработку потока, т.к. данные об остановке еще не попали к агенту.

Попробуйте выполнить запрос позднее.

11.28.13 Вернулся код ошибки 44013

Текст ошибки:

Stream status is unsuitable. Stream with id «{value}» is not currently being processed

Источник ошибки:

Ошибки сервиса Video Manager

Описание ошибки:

Невозможно определить статус потока. Поток с указанным идентификатором не находится в обработке.

11.28.14 Вернулся код ошибки 44014

Текст ошибки:

Unprocessable content. Failed to find agent that supports all analytics associated with the stream

Источник ошибки:

Ошибки сервиса Video Manager

Описание ошибки:

Невозможно найти агента, котороый мог бы найти все аналитики, указываемые в потоке.

Необходимо создавать отдельные потоки для разных агентов.

11.29 Ошибки сервиса Video Agent

11.29.1 Вернулся код ошибки 45001

Текст ошибки:

Stream processing error. Stream processing error: {value}

Источник ошибки:

Ошибки сервиса Video Agent

Описание ошибки:

Указанная ошибка обработки потока.

11.30 Ошибки сервиса Streams Retranslator

11.30.1 Вернулся код ошибки 46001

Текст ошибки:

Unprocessable Content. Failed to process stream by url {value}

Источник ошибки:

Ошибки сервиса Streams Retranslator

Описание ошибки:

Возникает, если сервис не смог обработать поток по указанному URL. Это может быть связано с неверным форматом ссылки или недоступностью ресурса.

11.30.2 Вернулся код ошибки 46002

Текст ошибки:

Mediamtx error. Mediamtx returned error: {value}

Источник ошибки:

Ошибки сервиса Streams Retranslator

Описание ошибки:

Возникает, если в процессе обработки потока Mediamtx возвращает ошибку. Обычно связано с внутренними проблемами конфигурации или некорректной работой Mediamtx.

11.30.3 Вернулся код ошибки 46003

Текст ошибки:

Unprocessable Content. The {value} retranslation is not supported at the moment

Источник ошибки:

Ошибки сервиса Streams Retranslator

Описание ошибки:

Возникает, если запрошенный тип ретрансляции (например, формат потока или метод передачи) на данный момент не поддерживается сервисом или ссылка указана некорректно.

11.30.4 Вернулся код ошибки 46004

Текст ошибки:

Authorization failed. Authorization failed

Источник ошибки:

Ошибки сервиса Streams Retranslator

Описание ошибки:

Возникает, если авторизация по токену не удалась. Проверьте токен.

11.30.5 Вернулся код ошибки 46005

Текст ошибки:

Object not found. Restream with id {value} not found

Источник ошибки:

Ошибки сервиса Streams Retranslator

Описание ошибки:

Возникает, если запрошенный поток не найден в системе. Это может произойти, если указан неверный идентификатор или ретрансляция была завершена.

11.30.6 Вернулся код ошибки 46006

Текст ошибки:

Unprocessable Content. Failed to open stream by url {value}

Источник ошибки:

Ошибки сервиса Streams Retranslator

Описание ошибки:

Возникает, если сервис не смог открыть поток по указанному URL. Возможные причины: недоступность потока, проблемы с сетью или некорректный формат ссылки.

11.31 Ошибки SDK

11.31.1 Вернулся код ошибки 99999

Текст ошибки:

Unknown fsdk core error, {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Неизвестная ошибка ядра FSDK.

11.31.2 Вернулся код ошибки 100001

Текст ошибки:

Buffer is empty {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Буфер пуст.

11.31.3 Вернулся код ошибки 100002

Текст ошибки:

Buffer is null {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Буфер равен Null.

11.31.4 Вернулся код ошибки 100003

Текст ошибки:

Buffer is full {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Буфер переполнен.

11.31.5 Вернулся код ошибки 100004

Текст ошибки:

Descriptors are incompatible {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Указанные биометрические шаблоны несовместимы. Биометрические шаблоны имеют разные версии и не могут быть сравнены.

11.31.6 Вернулся код ошибки 100005

Текст ошибки:

Internal error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Возникла внутренняя ошибка.

11.31.7 Вернулся код ошибки 100006

Текст ошибки:

Invalid buffer size {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Неверный размер буфера.

11.31.8 Вернулся код ошибки 100007

Текст ошибки:

Invalid descriptor {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Неверный биометрический шаблон. Проверьте файл биометрического шаблона.

11.31.9 Вернулся код ошибки 100008

Текст ошибки:

Invalid descriptor batch {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Неверный пакет биометрических шаблонов. Проверьте пакет биометрических шаблонов.

11.31.10 Вернулся код ошибки 100009

Текст ошибки:

Invalid detection {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Недопустимая детекция.

11.31.11 Вернулся код ошибки 100010

Текст ошибки:

Invalid image {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Неверное изображение. Проверьте, что файл изображения не поврежден и может быть обработан системой.

11.31.12 Вернулся код ошибки 100011

Текст ошибки:

Invalid image format {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Неверный формат изображения.

11.31.13 Вернулся код ошибки 100012

Текст ошибки:

Invalid image size {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Неверный размер изображения.

11.31.14 Вернулся код ошибки 100013

Текст ошибки:

Invalid input {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Неверный ввод.

11.31.15 Вернулся код ошибки 100014

Текст ошибки:

Invalid landmarks 5 {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Неверные landmarks5.

11.31.16 Вернулся код ошибки 100015

Текст ошибки:

Invalid landmarks 68 {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Неверные landmarks68.

11.31.17 Вернулся код ошибки 100016

Текст ошибки:

Invalid rectangle {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Неверный bbox.

11.31.18 Вернулся код ошибки 100017

Текст ошибки:

Invalid settings provider {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Неверный поставщик настроек.

11.31.19 Вернулся код ошибки 100018

Текст ошибки:

Licensing issue {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Проблема с лицензированием.

11.31.20 Вернулся код ошибки 100019

Текст ошибки:

Module is not initialized {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Модуль не инициализирован.

11.31.21 Вернулся код ошибки 100020

Текст ошибки:

Module is not ready {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Модуль не готов.

11.31.22 Вернулся код ошибки 100021

Текст ошибки:

Error during initialization fdsk image {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка инициализации FSDK изображения.

11.31.23 Вернулся код ошибки 100022

Текст ошибки:

Error during image loadin {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка загрузки изображения.

11.31.24 Вернулся код ошибки 100023

Текст ошибки:

Error during image saving {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка сохранения изображения.

11.31.25 Вернулся код ошибки 100024

Текст ошибки:

Archive image error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка архивации изображения.

11.31.26 Вернулся код ошибки 100025

Текст ошибки:

Invalid detection {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Недопустимая детекция.

11.31.27 Вернулся код ошибки 100026

Текст ошибки:

Image conversion not implemented {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Преобразование изображений не выполнено.

11.31.28 Вернулся код ошибки 100027

Текст ошибки:

Bad input image data pointer {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Неверный указатель данных входного изображения.

11.31.29 Вернулся код ошибки 100028

Текст ошибки:

Bad input image data size {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Неверный размер данных входного изображения.

11.31.30 Вернулся код ошибки 100029

Текст ошибки:

Unsupported image format {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Неподдерживаемый формат изображения.

11.31.31 Вернулся код ошибки 100030

Текст ошибки:

Invalid image height {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Неверная высота изображения.

11.31.32 Вернулся код ошибки 100031

Текст ошибки:

Bad path for image saving / loading {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Неверный путь для сохранения/загрузки изображений.

11.31.33 Вернулся код ошибки 100032

Текст ошибки:

Error at image memory opening {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка при открытии памяти изображений.

11.31.34 Вернулся код ошибки 100033

Текст ошибки:

Unsupported image type {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Неподдерживаемый тип изображения.

11.31.35 Вернулся код ошибки 100034

Текст ошибки:

Invalid image width {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Неверная ширина изображения.

11.31.36 Вернулся код ошибки 100035

Текст ошибки:

Batching error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка пакетной обработки.

11.31.37 Вернулся код ошибки 110001

Текст ошибки:

Creation descriptor error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка создания биометрического шаблона.

11.31.38 Вернулся код ошибки 110002

Текст ошибки:

Creation descriptor error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка создания пакета биометрических шаблонов.

11.31.39 Вернулся код ошибки 110003

Текст ошибки:

Creation core image error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка создания основного изображения.

11.31.40 Вернулся код ошибки 110004

Текст ошибки:

Estimation descriptor error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка оценки биометрического шаблона.

11.31.41 Вернулся код ошибки 110005

Текст ошибки:

Estimation descriptor error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка оценки пакета биометрических шаблонов.

11.31.42 Вернулся код ошибки 110006

Текст ошибки:

Estimation basic attributes error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка оценки основных атрибутов.

11.31.43 Вернулся код ошибки 110007

Текст ошибки:

Batch estimation basic attributes error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка оценки пакета основных атрибутов.

11.31.44 Вернулся код ошибки 110008

Текст ошибки:

Estimation AGS error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка оценки AGS.

11.31.45 Вернулся код ошибки 110009

Текст ошибки:

Estimation head pose error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка оценки положения головы.

11.31.46 Вернулся код ошибки 110011

Текст ошибки:

Estimation eyes gase error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка оценки взгляда.

11.31.47 Вернулся код ошибки 110012

Текст ошибки:

Estimation emotions error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка оценки эмоций.

11.31.48 Вернулся код ошибки 110013

Текст ошибки:

Estimation warp quality error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка оценки качества биометрического образца.

11.31.49 Вернулся код ошибки 110014

Текст ошибки:

Estimation mouth state error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка оценки состояния рта.

11.31.50 Вернулся код ошибки 110015

Текст ошибки:

Estimation eyes error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка оценки глаз.

11.31.51 Вернулся код ошибки 110016

Текст ошибки:

Creation warped image error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка создания биометрического образца.

11.31.52 Вернулся код ошибки 110017

Текст ошибки:

Landmarks transformation error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка преобразования контрольных точек лица (landmarks).

11.31.53 Вернулся код ошибки 110018

Текст ошибки:

Detect one face error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка обнаружения одного лица.

11.31.54 Вернулся код ошибки 110019

Текст ошибки:

Detect faces error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка обнаружения нескольких лиц.

11.31.55 Вернулся код ошибки 110020

Текст ошибки:

High memory usage {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка чрезмерного использования памяти.

11.31.56 Вернулся код ошибки 110021

Текст ошибки:

Detect one human body error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка обнаружения одного человеческого тела.

11.31.57 Вернулся код ошибки 110022

Текст ошибки:

Detect humans bodies error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка обнаружения человеческих тел.

11.31.58 Вернулся код ошибки 110023

Текст ошибки:

Estimation mask error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Произошла ошибка оценки маски.

11.31.59 Вернулся код ошибки 110024

Текст ошибки:

Filtered aggregation error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка агрегирования, связанная с заданным порогом.

11.31.60 Вернулся код ошибки 1100010

Текст ошибки:

Estimation ethnities error {value}

Источник ошибки:

Ошибки SDK

Описание ошибки:

Ошибка оценки этнической принадлежности.

11.32 Ошибки RPC

11.32.1 Вернулся код ошибки 120004

Текст ошибки:

The command does not exist {value}

Источник ошибки:

Ошибки RPC

Описание ошибки:

Команды не существует.

11.32.2 Вернулся код ошибки 120005

Текст ошибки:

Internal RPC error {value}

Источник ошибки:

Ошибки RPC

Описание ошибки:

Внутренняя ошибка RPC.

11.32.3 Вернулся код ошибки 120006

Текст ошибки:

The command is not valid {value}

Источник ошибки:

Ошибки RPC

Описание ошибки:

Команда недействительна.

11.33 Ошибки выполнения оценок

11.33.1 Вернулся код ошибки 200003

Текст ошибки:

Bad/incomplete input data. Not supported descriptor version {value}

Источник ошибки:

Ошибки выполнения оценок

Описание ошибки:

Биометрический шаблон указанной версии не поддерживается.

Убедитесь, что в настройках «DEFAULT_FACE_DESCRIPTOR_VERSION» или «DEFAULT_HUMAN_-DESCRIPTOR_VERSION» указана поддерживаемая версия биометрического шаблона.

11.33.2 Вернулся код ошибки 200004

Текст ошибки:

Internal error. Estimator {value} initialization fail

Источник ошибки:

Ошибки выполнения оценок

Описание ошибки:

Ошибка инициализации указанного эстиматора.

Проверьте наличие нейронной сети для эстиматора в контейнере Handlers.

11.33.3 Вернулся код ошибки 200005

Текст ошибки:

Internal error. {value} estimator was not initialized

Источник ошибки:

Ошибки выполнения оценок

Описание ошибки:

Указанный эстиматор не был инициализирован при старте контейнера Handlers.

Перезапустите контейнер с аргументом включения данного эстиматора. См. раздел «Включение/отключение некоторых эстиматоров и детекторов».

> Ссылка ведет на последнюю версию документации. Убедитесь, что вы читаете документацию для вашей текущей версии LUNA PLATFORM.

12 Описание параметров сервисов

12.1 Настройки сервиса Configurator

Данный раздел описывает параметры сервиса Configurator

Сервис Configurator настраивается только с помощью его локального конфигурационного файла «./example-docker/luna_configurator/configs/».

12.1.1 Группа параметров LUNA_CONFIGURATOR_DB

В данной группе параметров задаются настройки подключения к базе данных сервиса Configurator.

12.1.1.1 db_type

Параметр задает тип базы данных. Доступны следующие типы:

- «postgres» — тип СУБД PostgreSQL
- «oracle» — тип СУБД Oracle

Формат задания настройки — `string`.

Значение по умолчанию — `postgres`.

12.1.1.2 db_host

Параметр задает имя сервера (хост) базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.1.1.3 db_port

Параметр задает порт базы данных.

Порт по умолчанию для типа «postgres»- `5432`.

Порт по умолчанию для типа «oracle» — `1521`.

Формат задания настройки — `string`.

Значение по умолчанию — `5432`.

12.1.1.4 db_user

Параметр задает имя пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.1.1.5 db_password

Параметр задает пароль пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.1.1.6 db_name

Параметр задает имя базы данных для типа «postgres» или имя SID для типа «oracle».

Формат задания настройки — `string`.

Значение по умолчанию — `luna_configurator`.

12.1.1.7 connection_pool_size

Параметр задает размер пула соединений к БД. Реальное количество подключений может быть больше значения данной настройки на 1.

При необходимости в настройке «max_connections» конфигурационного файла PostgreSQL можно задать максимальное количество одновременных подключений к серверу БД. См. раздел «Продвинутая настройка PostrgeSQL» для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `10`.

12.1.1.8 dsn

Параметр задает строку подключения DSN для соединения с БД.

> DSN — это строка подключения, которая идентифицирует и указывает на источник данных (базу данных), к которому нужно установить соединение.

В строке DSN могут задаваться такие настройки, как множественные хосты, аутентификационные данные, порт и другие параметры.

> Настройки зависят от типа БД. Множественные хосты поддерживаются только с PostgreSQL.

> По умолчанию параметр «dsn» не отображается во вкладке «Settings» в Configurator. Проверить список всех доступных параметров для группы настроек можно на вкладке «Limitations».

Пример:

```
{
"dsn": "luna:luna@postgres01:5432,postgres02:5432/luna_configurator?
    some_option=some_value"
"db_settings": {
    "connection_pool_size": 5
    }
}
```

Здесь:

- «luna:luna» — имя пользователя и пароль для подключения к PostgreSQL;
- «@postgres01:5432,postgres02:5432» — список хостов и портов, разделенных запятыми. Это означает, что сервис будет пытаться подключиться к первому хосту («postgres01») на порту 5432. Если это не удастся, она попытается подключиться ко второму хосту («postgres02») также на порту 5432;
- «/luna_configurator» — название базы данных;
- «?some_option=some_value» — опциональные параметры для подключения.

При необходимости можно комбинировать строку DSN и классические настройки, однако строка DSN является более приоритетной. Можно частично заполнить строку DSN (например, «postgres01,postgres02/luna_configurator»), и тогда недостающие параметры будут заполнены из значений параметров «db_host», «db_port», «db_name», «db_user» и «db_password».

При запуске сервис создаст пул подключений к одному из доступных хостов DSN. В случае возникновения проблем с установлением соединения после нескольких неудачных попыток, сервис снова попытается настроить пул подключений к любому из доступных хостов DSN.

12.1.2 Группа параметров LUNA_CONFIGURATOR_LOGGER

Данная группа параметров задает настройки логирования.

12.1.2.1 log_level

Параметр задает уровень отладочной печати, по приоритету: «ERROR», «WARNING», «INFO», «DEBUG».

Формат задания настройки — `string`.

Значение по умолчанию — `INFO`.

12.1.2.2 log_time

Параметр задает формат времени, используемый в записях лога. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;

- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.1.2.3 log_to_stdout

Параметр позволяет отправлять логи в стандартный вывод (stdout).

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`

12.1.2.4 log_to_file

Параметр позволяет сохранять логи в файл. Директория с файлами логов указывается в параметре «folder_with_logs».

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.1.2.5 folder_with_logs

Параметр задает директорию, в которой хранятся логи. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить параметр «log_to_file».

Формат задания настройки — `string`.

Значение по умолчанию — `./`

Пример:

```
  "folder_with_logs": "/srv/logs"
```

12.1.2.6 max_log_file_size

Параметр задает максимальный размер файла лога в МБ перед выполнением его ротации (0 — не использовать ротацию).

Для использования данного параметра требуется включить параметр «log_to_file».

> При необходимости можно настроить ротацию логов Docker. См. раздел «Настройка ротации логов Docker» в руководстве по установке LUNA PLATFORM.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024`

12.1.2.7 multiline_stack_trace

Параметр включает многострочную трассировку стека в логах. Когда параметр включен, информация о стеке вызовов записывается в логах так, что каждый фрейм стека помещается на отдельной строке, что улучшает читаемость. Если параметр выключен, информация о стеке вызовов записывается на одной строке, что может сделать логи менее удобными для анализа.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.1.2.8 format

Параметр определяет формат выводимых логов. Доступны следующие значения:

- «default» — стандартный формат вывода логов LUNA PLATFORM
- «json» — вывод логов в формате json
- «ecs» — вывод логов в формате ECS (Elastic Common Schema)

При использовании значения «ecs» будут использоваться следующие поля:

- «http.response.status_code» — содержит код состояния ответа HTTP (200, 404, 500 и т.д.);
- «http.response.execution_time» — содержит информацию о времени, затраченном на выполнение запроса и получение ответа;
- «http.request.method» — содержит метод HTTP-запроса (GET, POST, PUT и т.д.);
- «url.path» — содержит путь в URL-адресе запроса;
- «error.code» — содержит код ошибки, если запрос завершается с ошибкой.

Формат задания настройки — `string`.

Значение по умолчанию — `default`.

12.1.3 Группа параметров LUNA_CONFIGURATOR_HTTP_SETTINGS

В данной группе параметров содержатся настройки, отвечающие за обработку HTTP-подключений. См. подробную информацию по следующей ссылке: https://sanic.dev/en/guide/deployment/configuration.html#builtin-values

12.1.3.1 request_timeout

Параметр задает продолжительность времени между моментом, когда новое открытое TCP-соединение передается на сервер, и моментом, когда получен весь HTTP-запрос.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.1.3.2 response_timeout

Параметр задает продолжительность времени между моментом, когда сервер передает HTTP-запрос приложению, и моментом, когда HTTP-ответ отправляется клиенту.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `600`.

12.1.3.3 request_max_size

Параметр задает максимальный размер запроса.

Формат задания настройки — `integer` (байты).

Значение по умолчанию — `1073741824`.

12.1.3.4 keep_alive_timeout

Параметр задает тайм-аут поддержания активности HTTP.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `15`.

12.1.4 Группа параметров LUNA_MONITORING

Данная группа параметров задает настройки мониторинга.

См. подробную информацию о мониторинге в разделе «Мониторинг».

12.1.4.1 storage_type

Тип хранилища для хранения данных мониторинга.

В настоящее время доступно только БД Influx.

Формат задания настройки — `string`.

Значение по умолчанию — `influx`.

12.1.4.2 send_data_for_monitoring

Параметр позволяет включить или отключить отправку данных для мониторинга в InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.1.4.3 use_ssl

Параметр позволяет использовать HTTPS для подключения к InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.1.4.4 organization

Параметр задает рабочую область InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.1.4.5 token

Параметр задает аутентификации InfluxDB 2.x.

Формат задания настройки — `string`.

12.1.4.6 bucket

Параметр задаёт имя бакета InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_monitoring`.

12.1.4.7 host

Параметр задаёт IP-адрес InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.1.4.8 port

Параметр задает порт InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `8086`.

12.1.4.9 flushing_period

Параметр задает частоту отправки данных мониторинга в InfluxDB.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `1`.

12.1.5 Группа параметров LUNA_SERVICE_METRICS

Данная группа параметров включает и настраивает сбор метрик в формате Prometheus.

См. подробную информацию в разделе «Мониторинг».

12.1.5.1 enabled

Параметр включает сбор метрик.

Если сбор метрик отключен, то запрос к ресурсу `/metrics` вернет соответствующее сообщение.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.1.5.2 metrics_format

Параметр задает формат метрик.

На данный момент поддерживается только формат Prometheus.

См. официальную документацию Prometheus для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `prometheus`.

12.1.5.3 extra_labels

Параметр задает пользовательские типы лейблов.

Формат задания настройки — `label_name=label_value`.

Значение по умолчанию не задано.

12.1.6 Прочие

12.1.6.1 storage_time

Параметр задает формат времени, используемый для записей в базе данных. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.2 Настройки сервиса API

Данный раздел описывает параметры сервиса API.

Настройку сервиса можно выполнить с помощью сервиса Configurator.

12.2.1 Группа параметров LUNA_CONFIGURATOR

Данная группа параметров задаёт настройки подключения сервиса API к сервису Configurator.

Данная группа параметров не будет видна в пользовательском интерфейсе сервиса Configurator. Параметры можно изменить только в конфигурационном файле «config.conf», расположенном в директории «/srv/luna_api/configs/» соответствующего контейнера.

12.2.1.1 use_configurator

Параметр позволяет включить использование сервиса Configurator.

> С помощью сервиса Configurator упрощается настройка сервисов LP. Сервис сохраняет все необходимые конфигурации для всех сервисов LP в одном месте.

При отключенном параметре будут использованы настройки из файла «config.conf», расположенного в директории «/srv/luna_api/configs/» соответствующего контейнера.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.2.1.2 luna_configurator_origin

Параметр задает протокол, IP адрес и порт сервиса Configurator.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5070`.

12.2.1.3 luna_configurator_api

Параметр задает версию API сервиса Configurator. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.2.2 Группа параметров LUNA_MONITORING

Данная группа параметров задает настройки мониторинга.

См. подробную информацию о мониторинге в разделе «Мониторинг».

12.2.2.1 storage_type

Тип хранилища для хранения данных мониторинга.

В настоящее время доступно только БД Influx.

Формат задания настройки — `string`.

Значение по умолчанию — `influx`.

12.2.2.2 send_data_for_monitoring

Параметр позволяет включить или отключить отправку данных для мониторинга в InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.2.2.3 use_ssl

Параметр позволяет использовать HTTPS для подключения к InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.2.2.4 organization

Параметр задает рабочую область InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.2.2.5 token

Параметр задает аутентификации InfluxDB 2.x.

Формат задания настройки — `string`.

12.2.2.6 bucket

Параметр задаёт имя бакета InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_monitoring`.

12.2.2.7 host

Параметр задаёт IP-адрес InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.2.2.8 port

Параметр задает порт InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `8086`.

12.2.2.9 flushing_period

Параметр задает частоту отправки данных мониторинга в InfluxDB.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `1`.

12.2.3 Группа параметров LUNA_API_LOGGER

Данная группа параметров задает настройки логирования.

12.2.3.1 log_level

Параметр задает уровень отладочной печати, по приоритету: «ERROR», «WARNING», «INFO», «DEBUG».

Формат задания настройки — `string`.

Значение по умолчанию — `INFO`.

12.2.3.2 log_time

Параметр задает формат времени, используемый в записях лога. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.2.3.3 log_to_stdout

Параметр позволяет отправлять логи в стандартный вывод (stdout).

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`

12.2.3.4 log_to_file

Параметр позволяет сохранять логи в файл. Директория с файлами логов указывается в параметре «folder_with_logs».

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.2.3.5 folder_with_logs

Параметр задает директорию, в которой хранятся логи. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить параметр «log_to_file».

Формат задания настройки — `string`.

Значение по умолчанию — `./`

Пример:

```
  "folder_with_logs": "/srv/logs"
```

12.2.3.6 max_log_file_size

Параметр задает максимальный размер файла лога в МБ перед выполнением его ротации (0 — не использовать ротацию).

Для использования данного параметра требуется включить параметр «log_to_file».

> При необходимости можно настроить ротацию логов Docker. См. раздел «Настройка ротации логов Docker» в руководстве по установке LUNA PLATFORM.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024`

12.2.3.7 multiline_stack_trace

Параметр включает многострочную трассировку стека в логах. Когда параметр включен, информация о стеке вызовов записывается в логах так, что каждый фрейм стека помещается на отдельной строке, что улучшает читаемость. Если параметр выключен, информация о стеке вызовов записывается на одной строке, что может сделать логи менее удобными для анализа.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.2.3.8 format

Параметр определяет формат выводимых логов. Доступны следующие значения:

- «default» — стандартный формат вывода логов LUNA PLATFORM
- «json» — вывод логов в формате json
- «ecs» — вывод логов в формате ECS (Elastic Common Schema)

При использовании значения «ecs» будут использоваться следующие поля:

- «http.response.status_code» — содержит код состояния ответа HTTP (200, 404, 500 и т.д.);
- «http.response.execution_time» — содержит информацию о времени, затраченном на выполнение запроса и получение ответа;
- «http.request.method» — содержит метод HTTP-запроса (GET, POST, PUT и т.д.);
- «url.path» — содержит путь в URL-адресе запроса;
- «error.code» — содержит код ошибки, если запрос завершается с ошибкой.

Формат задания настройки — `string`.

Значение по умолчанию — `default`.

12.2.4 Группа параметров LUNA_FACES_ADDRESS

Данная группа параметров задает настройки подключения к сервису Faces.

12.2.4.1 origin

Параметр задает протокол, IP адрес и порт сервиса Faces.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Faces, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Faces.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5030`.

12.2.4.2 api_version

Параметр задает версию API сервиса Faces. Доступная версия API — «3».

Формат задания настройки — `integer`.

Значение по умолчанию — `3`.

12.2.5 Группа параметров LUNA_FACES_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Faces.

12.2.5.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Faces. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.2.5.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.5.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.2.5.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.6 Группа параметров LUNA_VIDEO_AGENT_ADDRESS

Данная группа параметров задает настройки подключения к сервису Video Agent.

12.2.6.1 origin

Параметр задает протокол, IP адрес и порт сервиса Video Agent.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Video Agent, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Video Agent.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5240`.

12.2.6.2 api_version

Параметр задает версию API сервиса Video Agent. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.2.7 Группа параметров LUNA_VIDEO_AGENT_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Video Agent.

12.2.7.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Video Agent. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.2.7.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.7.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.2.7.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.8 Группа параметров LUNA_IMAGE_STORE_FACES_SAMPLES_ADDRESS

В данной группе параметров задаются настройки бакета для хранения БО лиц.

12.2.8.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.2.8.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.2.8.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-samples`.

12.2.9 Группа параметров LUNA_IMAGE_STORE_FACES_SAMPLES_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к бакету, хранящему БО лиц.

12.2.9.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к бакету, хранящему БО лиц. Это время ожидания, в течение которого клиент пытается установить соединение с бакетом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `30`.

12.2.9.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.10 Группа параметров LUNA_IMAGE_STORE_BODIES_SAMPLES_ADDRESS

В данной группе параметров задаются настройки бакета для хранения БО тел.

12.2.10.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.2.10.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.2.10.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-bodies-samples`.

12.2.11 Группа параметров LUNA_IMAGE_STORE_BODIES_SAMPLES_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к бакету, хранящему БО тел.

12.2.11.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к бакету, хранящему БО тел. Это время ожидания, в течение которого клиент пытается установить соединение с бакетом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `30`.

12.2.11.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.12 Группа параметров LUNA_IMAGE_STORE_IMAGES_ADDRESS

В данной группе параметров задаются настройки бакета для хранения исходных изображений.

12.2.12.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.2.12.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.2.12.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-image-origin`.

12.2.13 Группа параметров LUNA_IMAGE_STORE_IMAGES_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к бакету, хранящему исходных изображений.

12.2.13.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к бакету, хранящему исходных изображений. Это время ожидания, в течение которого клиент пытается установить соединение с бакетом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `30`.

12.2.13.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.14 Группа параметров LUNA_IMAGE_STORE_OBJECTS_ADDRESS

В данной группе параметров задаются настройки бакета для хранения объектов.

12.2.14.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.2.14.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.2.14.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-objects`.

12.2.15 Группа параметров LUNA_IMAGE_STORE_OBJECTS_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к бакету, хранящему объектов.

12.2.15.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к бакету, хранящему объектов. Это время ожидания, в течение которого клиент пытается установить соединение с бакетом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `30`.

12.2.15.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.16 Группа параметров LUNA_SENDER_ADDRESS

Данная группа параметров задает настройки подключения к сервису Sender.

12.2.16.1 origin

Параметр задает протокол, IP адрес и порт сервиса Sender.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Sender, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Sender.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5080`.

12.2.16.2 api_version

Параметр задает версию API сервиса Sender. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.2.17 Группа параметров LUNA_STREAMS_RETRANSLATOR_ADDRESS

Данная группа параметров задает настройки подключения к сервису Streams Reatranslator.

12.2.17.1 origin

Параметр задает протокол, IP адрес и порт сервиса Streams Reatranslator.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Streams Reatranslator, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Streams Reatranslator.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5250`.

12.2.17.2 api_version

Параметр задает версию API сервиса Streams Reatranslator. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.2.18 Группа параметров ADDITIONAL_SERVICES_USAGE

12.2.18.1 luna_events

Параметр задает возможность использования сервиса Events.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.2.18.2 luna_tasks

Параметр задает возможность использования сервиса Tasks.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.2.18.3 luna_faces

Параметр задает возможность использования сервиса Faces.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.2.18.4 luna_handlers

Параметр задает возможность использования сервиса Handlers.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.2.18.5 luna_sender

Параметр задает возможность использования сервиса Sender.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.2.18.6 luna_matcher_proxy

Параметр задает возможность использования сервиса Python Matcher Proxy.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.2.18.7 luna_image_store

Параметр задает возможность использования сервиса Image Store.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.2.18.8 luna_lambda

Параметр задает возможность использования сервиса Lambda.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.2.18.9 luna_video_manager

Параметр задает возможность использования сервиса Video Manager.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.2.18.10 luna_video_agent

Параметр задает возможность использования сервиса Video Agent.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.2.18.11 luna_streams_retranslator

Параметр задает возможность использования сервиса Streams Retranslator.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.2.19 Группа параметров LUNA_EVENTS_ADDRESS

Данная группа параметров задает настройки подключения к сервису Events.

12.2.19.1 origin

Параметр задает протокол, IP адрес и порт сервиса Events.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Events, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Events.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5040`.

12.2.19.2 api_version

Параметр задает версию API сервиса Events. Доступная версия API — «2».

Формат задания настройки — `integer`.

Значение по умолчанию — `2`.

12.2.20 Группа параметров LUNA_EVENTS_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Events.

12.2.20.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Events. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.2.20.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.20.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.2.20.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.21 Группа параметров LUNA_HANDLERS_ADDRESS

Данная группа параметров задает настройки подключения к сервису Handlers.

12.2.21.1 origin

Параметр задает протокол, IP адрес и порт сервиса Handlers.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Handlers, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Handlers.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5090`.

12.2.21.2 api_version

Параметр задает версию API сервиса Handlers. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.2.22 Группа параметров LUNA_HANDLERS_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Handlers.

12.2.22.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Handlers. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.2.22.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.22.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.2.22.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.23 Группа параметров LUNA_PYTHON_MATCHER_ADDRESS

Данная группа параметров задает настройки подключения к сервису Python Matcher.

12.2.23.1 origin

Параметр задает протокол, IP адрес и порт сервиса Python Matcher.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Python Matcher, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Python Matcher.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5100`.

12.2.23.2 api_version

Параметр задает версию API сервиса Python Matcher. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.2.24 Группа параметров LUNA_PYTHON_MATCHER_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Python Matcher.

12.2.24.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Python Matcher. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.2.24.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.24.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.2.24.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.25 Группа параметров LUNA_MATCHER_PROXY_ADDRESS

Данная группа параметров задает настройки подключения к сервису Python Matcher Proxy.

12.2.25.1 origin

Параметр задает протокол, IP адрес и порт сервиса Python Matcher Proxy.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Python Matcher Proxy, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Python Matcher Proxy.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5110`.

12.2.25.2 api_version

Параметр задает версию API сервиса Python Matcher Proxy. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.2.26 Группа параметров LUNA_PYTHON_MATCHER_PROXY_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Python Matcher Proxy.

12.2.26.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Python Matcher Proxy. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.2.26.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.26.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.2.26.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.27 Группа параметров LUNA_TASKS_ADDRESS

Данная группа параметров задает настройки подключения к сервису Tasks.

12.2.27.1 origin

Параметр задает протокол, IP адрес и порт сервиса Tasks.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Tasks, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Tasks.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5050`.

12.2.27.2 api_version

Параметр задает версию API сервиса Tasks. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.2.28 Группа параметров LUNA_TASKS_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Tasks.

12.2.28.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Tasks. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.2.28.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.28.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.2.28.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.29 Группа параметров LUNA_LICENSES_ADDRESS

Данная группа параметров задает настройки подключения к сервису Licenses.

12.2.29.1 origin

Параметр задает протокол, IP адрес и порт сервиса Licenses.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Licenses, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Licenses.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5120`.

12.2.29.2 api_version

Параметр задает версию API сервиса Licenses. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.2.30 Группа параметров LUNA_ACCOUNTS_ADDRESS

Данная группа параметров задает настройки подключения к сервису Accounts.

12.2.30.1 origin

Параметр задает протокол, IP адрес и порт сервиса Accounts.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Accounts, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Accounts.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5170`.

12.2.30.2 api_version

Параметр задает версию API сервиса Accounts. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.2.31 Группа параметров LUNA_ACCOUNTS_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Accounts.

12.2.31.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Accounts. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.2.31.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.31.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.2.31.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.32 Группа параметров LUNA_REMOTE_SDK_ADDRESS

Данная группа параметров задает настройки подключения к сервису Remote SDK.

12.2.32.1 origin

Параметр задает протокол, IP адрес и порт сервиса Remote SDK.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Remote SDK, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Remote SDK.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5220`.

12.2.32.2 api_version

Параметр задает версию API сервиса Remote SDK. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.2.33 Группа параметров LUNA_REMOTE_SDK_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Remote SDK.

12.2.33.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Remote SDK. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.2.33.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.33.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.2.33.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.34 Группа параметров LUNA_LAMBDA_ADDRESS

Данная группа параметров задает настройки подключения к сервису Lambda.

12.2.34.1 origin

Параметр задает протокол, IP адрес и порт сервиса Lambda.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Lambda, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Lambda.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5210`.

12.2.34.2 api_version

Параметр задает версию API сервиса Lambda. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.2.35 Группа параметров LUNA_LAMBDA_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Lambda.

12.2.35.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Lambda. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.2.35.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.35.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.2.35.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.36 Группа параметров EXTERNAL_LUNA_API_ADDRESS

Данная группа параметров задает адрес сервиса API, который будет отображаться в параметре «external_url» в теле ответа на создание различных объектов. В результате можно будет получить абсолютную ссылку до созданного объекта.

Пример тела ответа на создание списка:

```
{
    "list_id": "3b94d026-2434-4a52-b1e7-da4a97fc0398",
    "url": "/6/lists/3b94d026-2434-4a52-b1e7-da4a97fc0398",
    "external_url": "http://127.0.0.1:5000/6/lists/3b94d026-2434-4a52-b1e7-
       da4a97fc0398"
}
```

Здесь:

- «url» — относительная ссылка до списка
- «external_url» — абсолютная ссылка до списка, состоящая из параметров «origin», «api_-version» и «url»

Это позволяет использовать ссылки из ответов сервиса API в своих целях, не зная точного адреса сервиса. Также это позволяет передавать ссылки в удобном формате, по которому можно сразу получить их содержимое.

12.2.36.1 origin

Параметр задает протокол, IP адрес и порт сервиса API, отображаемый в параметре «external_url» в теле ответа на создание различных объектов.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5000`.

12.2.36.2 api_version

Параметр задает версию API сервиса API, отображаемую в параметре «external_url» в теле ответа на создание различных объектов.

Формат задания настройки — `integer`.

Значение по умолчанию — `6`.

12.2.37 Группа параметров LUNA_API_HTTP_SETTINGS

В данной группе параметров содержатся настройки, отвечающие за обработку HTTP-подключений. См. подробную информацию по следующей ссылке: https://sanic.dev/en/guide/deployment/configuration.html#builtin-values

12.2.37.1 request_timeout

Параметр задает продолжительность времени между моментом, когда новое открытое TCP-соединение передается на сервер, и моментом, когда получен весь HTTP-запрос.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.2.37.2 response_timeout

Параметр задает продолжительность времени между моментом, когда сервер передает HTTP-запрос приложению, и моментом, когда HTTP-ответ отправляется клиенту.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `600`.

12.2.37.3 request_max_size

Параметр задает максимальный размер запроса.

Формат задания настройки — `integer` (байты).

Значение по умолчанию — `1073741824`.

12.2.37.4 keep_alive_timeout

Параметр задает тайм-аут поддержания активности HTTP.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `15`.

12.2.38 Группа параметров LUNA_SERVICE_METRICS

Данная группа параметров включает и настраивает сбор метрик в формате Prometheus.

См. подробную информацию в разделе «Мониторинг».

12.2.38.1 enabled

Параметр включает сбор метрик.

Если сбор метрик отключен, то запрос к ресурсу `/metrics` вернет соответствующее сообщение.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.2.38.2 metrics_format

Параметр задает формат метрик.

На данный момент поддерживается только формат Prometheus.

См. официальную документацию Prometheus для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `prometheus`.

12.2.38.3 extra_labels

Параметр задает пользовательские типы лейблов.

Формат задания настройки — `label_name=label_value`.

Значение по умолчанию не задано.

12.2.39 Прочие

12.2.39.1 luna_api_active_plugins

Параметр задает список плагинов, которые должен использовать сервис.

Имена задаются в следующем формате:

```
[
   "plugin_1",
   "plugin_2",
   "plugin_3"
]
```

> Список должен содержать имена файлов без расширения (.py).

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.2.39.2 luna_api_plugins_settings

Параметр задает настройки для плагинов.

> На данный момент доступны настройки только для плагина «luna-streams.py». См. подробную информацию в разделе «Проксирование запросов в LUNA Streams через LUNA API» руководства администратора FaceStream.

Важно! Плагины должны быть включены в настройке «luna_api_active_plugins».

Формат задания настройки — `object`.

Значение по умолчанию не задано.

Настройки плагина luna-streams

Для плагина «luna-streams.py» доступны следующие настройки:

- «luna-streams-address» > «origin» — протокол, IP адрес и порт сервиса LUNA Streams
- «luna-streams-address» > «api_version» — версия API сервиса LUNA Streams
- «luna-streams-timeouts» > «connect» — таймаут для установки соединения при отправке HTTP-запроса к сервису LUNA Streams
- «luna-streams-timeouts» > «request» — общий таймаут для выполнения всего HTTP-запроса
- «luna-streams-timeouts» > «sock_connect» — таймаут для установки соединения на уровне сокетов
- «luna-streams-timeouts» > «sock_read» — таймаут на чтение данных с сокета после успешного соединения

Пример задания настройки «luna_api_plugins_settings» для плагина «luna-streams.py»:

```
{
    "luna-streams": {
        "luna-streams-address": {
            "origin": "http://127.0.0.1:5160/1",
            "api_version": 1
        },
        "luna-streams-timeouts": {
            "request": 60,
            "connect": 20,
            "sock_connect": 10,
            "sock_read": 60
        }
    }
}
```

Если настройка «luna_api_plugins_settings» не задана или не заданы какие-то из вышеописанных настроек для плагина, будут применены значения по умолчанию. Значения по умолчанию указаны в примере выше.

12.2.39.3 allow_luna_account_auth_header

Параметр включает возможность авторизации по заголовку «Luna-Account-Id» (**LunaAccountIdAuth**), в котором указывается сгенерированный после создания аккаунта «account_id».

Такая авторизация была принята за основу до версии 5.30.0 и считается устаревшей.

В спецификации OpenAPI заголовок «Luna-Account-Id» помечен словом **Deprecated**.

См. подробную информацию об авторизации в разделе «Аккаунты, токены и способы авторизации».

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `0`.

12.2.39.4 storage_time

Параметр задает формат времени, используемый для записей в базе данных. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.2.39.5 default_face_descriptor_version

Параметр задает используемую версию биометрического шаблона лица.

Формат задания настройки — `string`.

Значение по умолчанию — `59`.

Примечание. См. подробную информацию о версиях биометрических шаблонах в разделе «Нейронные сети».

12.3 Настройки сервиса Admin

Данный раздел описывает параметры сервиса Admin.

Настройку сервиса можно выполнить с помощью сервиса Configurator.

12.3.1 Группа параметров LUNA_CONFIGURATOR

Данная группа параметров задаёт настройки подключения сервиса Admin к сервису Configurator.

Данная группа параметров не будет видна в пользовательском интерфейсе сервиса Configurator. Параметры можно изменить только в конфигурационном файле «config.conf», расположенном в директории «/srv/luna_admin/configs/» соответствующего контейнера.

12.3.1.1 use_configurator

Параметр позволяет включить использование сервиса Configurator.

> С помощью сервиса Configurator упрощается настройка сервисов LP. Сервис сохраняет все необходимые конфигурации для всех сервисов LP в одном месте.

При отключенном параметре будут использованы настройки из файла «config.conf», расположенного в директории «/srv/luna_admin/configs/» соответствующего контейнера.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.3.1.2 luna_configurator_origin

Параметр задает протокол, IP адрес и порт сервиса Configurator.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5070`.

12.3.1.3 luna_configurator_api

Параметр задает версию API сервиса Configurator. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.3.2 Группа параметров LUNA_MONITORING

Данная группа параметров задает настройки мониторинга.

См. подробную информацию о мониторинге в разделе «Мониторинг».

12.3.2.1 storage_type

Тип хранилища для хранения данных мониторинга.

В настоящее время доступно только БД Influx.

Формат задания настройки — `string`.

Значение по умолчанию — `influx`.

12.3.2.2 send_data_for_monitoring

Параметр позволяет включить или отключить отправку данных для мониторинга в InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.3.2.3 use_ssl

Параметр позволяет использовать HTTPS для подключения к InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.3.2.4 organization

Параметр задает рабочую область InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.3.2.5 token

Параметр задает аутентификации InfluxDB 2.x.

Формат задания настройки — `string`.

12.3.2.6 bucket

Параметр задаёт имя бакета InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_monitoring`.

12.3.2.7 host

Параметр задаёт IP-адрес InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.3.2.8 port

Параметр задает порт InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `8086`.

12.3.2.9 flushing_period

Параметр задает частоту отправки данных мониторинга в InfluxDB.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `1`.

12.3.3 Группа параметров LUNA_ADMIN_LOGGER

Данная группа параметров задает настройки логирования.

12.3.3.1 log_level

Параметр задает уровень отладочной печати, по приоритету: «ERROR», «WARNING», «INFO», «DEBUG».

Формат задания настройки — `string`.

Значение по умолчанию — `INFO`.

12.3.3.2 log_time

Параметр задает формат времени, используемый в записях лога. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.3.3.3 log_to_stdout

Параметр позволяет отправлять логи в стандартный вывод (stdout).

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`

12.3.3.4 log_to_file

Параметр позволяет сохранять логи в файл. Директория с файлами логов указывается в параметре «folder_with_logs».

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.3.3.5 folder_with_logs

Параметр задает директорию, в которой хранятся логи. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить параметр «log_to_file».

Формат задания настройки — `string`.

Значение по умолчанию — `./`

Пример:

```
  "folder_with_logs": "/srv/logs"
```

12.3.3.6 max_log_file_size

Параметр задает максимальный размер файла лога в МБ перед выполнением его ротации (0 — не использовать ротацию).

Для использования данного параметра требуется включить параметр «log_to_file».

> При необходимости можно настроить ротацию логов Docker. См. раздел «Настройка ротации логов Docker» в руководстве по установке LUNA PLATFORM.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024`

12.3.3.7 multiline_stack_trace

Параметр включает многострочную трассировку стека в логах. Когда параметр включен, информация о стеке вызовов записывается в логах так, что каждый фрейм стека помещается на отдельной строке, что улучшает читаемость. Если параметр выключен, информация о стеке вызовов записывается на одной строке, что может сделать логи менее удобными для анализа.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.3.3.8 format

Параметр определяет формат выводимых логов. Доступны следующие значения:

- «default» — стандартный формат вывода логов LUNA PLATFORM
- «json» — вывод логов в формате json
- «ecs» — вывод логов в формате ECS (Elastic Common Schema)

При использовании значения «ecs» будут использоваться следующие поля:

- «http.response.status_code» — содержит код состояния ответа HTTP (200, 404, 500 и т.д.);
- «http.response.execution_time» — содержит информацию о времени, затраченном на выполнение запроса и получение ответа;
- «http.request.method» — содержит метод HTTP-запроса (GET, POST, PUT и т.д.);
- «url.path» — содержит путь в URL-адресе запроса;
- «error.code» — содержит код ошибки, если запрос завершается с ошибкой.

Формат задания настройки — `string`.

Значение по умолчанию — `default`.

12.3.4 Группа параметров LUNA_ACCOUNTS_ADDRESS

Данная группа параметров задает настройки подключения к сервису Accounts.

12.3.4.1 origin

Параметр задает протокол, IP адрес и порт сервиса Accounts.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Accounts, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Accounts.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5170`.

12.3.4.2 api_version

Параметр задает версию API сервиса Accounts. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.3.5 Группа параметров LUNA_API_ADDRESS

Данная группа параметров задает настройки подключения к сервису API.

12.3.5.1 origin

Параметр задает протокол, IP адрес и порт сервиса API.

IP адрес «127.0.0.1» означает, что будет использоваться сервис API, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом API.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5000`.

12.3.5.2 api_version

Параметр задает версию API сервиса API. Доступная версия API — «6».

Формат задания настройки — `integer`.

Значение по умолчанию — `6`.

12.3.6 Группа параметров LUNA_FACES_ADDRESS

Данная группа параметров задает настройки подключения к сервису Faces.

12.3.6.1 origin

Параметр задает протокол, IP адрес и порт сервиса Faces.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Faces, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Faces.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5030`.

12.3.6.2 api_version

Параметр задает версию API сервиса Faces. Доступная версия API — «3».

Формат задания настройки — `integer`.

Значение по умолчанию — `3`.

12.3.7 Группа параметров LUNA_VIDEO_AGENT_ADDRESS

Данная группа параметров задает настройки подключения к сервису Video Agent.

12.3.7.1 origin

Параметр задает протокол, IP адрес и порт сервиса Video Agent.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Video Agent, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Video Agent.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5240`.

12.3.7.2 api_version

Параметр задает версию API сервиса Video Agent. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.3.8 Группа параметров LUNA_IMAGE_STORE_FACES_SAMPLES_ADDRESS

В данной группе параметров задаются настройки бакета для хранения БО лиц.

12.3.8.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.3.8.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.3.8.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-samples`.

12.3.9 Группа параметров LUNA_IMAGE_STORE_BODIES_SAMPLES_ADDRESS

В данной группе параметров задаются настройки бакета для хранения БО тел.

12.3.9.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.3.9.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.3.9.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-bodies-samples`.

12.3.10 Группа параметров LUNA_IMAGE_STORE_IMAGES_ADDRESS

В данной группе параметров задаются настройки бакета для хранения исходных изображений.

12.3.10.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.3.10.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.3.10.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-image-origin`.

12.3.11 Группа параметров LUNA_IMAGE_STORE_TASK_RESULT_ADDRESS

В данной группе параметров задаются настройки бакета для хранения результатов задач.

12.3.11.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.3.11.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.3.11.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `task-result`.

12.3.12 Группа параметров LUNA_SENDER_ADDRESS

Данная группа параметров задает настройки подключения к сервису Sender.

12.3.12.1 origin

Параметр задает протокол, IP адрес и порт сервиса Sender.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Sender, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Sender.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5080`.

12.3.12.2 api_version

Параметр задает версию API сервиса Sender. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.3.13 Группа параметров LUNA_EVENTS_ADDRESS

Данная группа параметров задает настройки подключения к сервису Events.

12.3.13.1 origin

Параметр задает протокол, IP адрес и порт сервиса Events.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Events, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Events.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5040`.

12.3.13.2 api_version

Параметр задает версию API сервиса Events. Доступная версия API — «2».

Формат задания настройки — `integer`.

Значение по умолчанию — `2`.

12.3.14 Группа параметров LUNA_TASKS_ADDRESS

Данная группа параметров задает настройки подключения к сервису Tasks.

12.3.14.1 origin

Параметр задает протокол, IP адрес и порт сервиса Tasks.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Tasks, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Tasks.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5050`.

12.3.14.2 api_version

Параметр задает версию API сервиса Tasks. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.3.15 Группа параметров LUNA_HANDLERS_ADDRESS

Данная группа параметров задает настройки подключения к сервису Handlers.

12.3.15.1 origin

Параметр задает протокол, IP адрес и порт сервиса Handlers.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Handlers, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Handlers.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5090`.

12.3.15.2 api_version

Параметр задает версию API сервиса Handlers. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.3.16 Группа параметров LUNA_REMOTE_SDK_ADDRESS

Данная группа параметров задает настройки подключения к сервису Remote SDK.

12.3.16.1 origin

Параметр задает протокол, IP адрес и порт сервиса Remote SDK.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Remote SDK, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Remote SDK.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5220`.

12.3.16.2 api_version

Параметр задает версию API сервиса Remote SDK. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.3.17 Группа параметров LUNA_PYTHON_MATCHER_ADDRESS

Данная группа параметров задает настройки подключения к сервису Python Matcher.

12.3.17.1 origin

Параметр задает протокол, IP адрес и порт сервиса Python Matcher.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Python Matcher, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Python Matcher.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5100`.

12.3.17.2 api_version

Параметр задает версию API сервиса Python Matcher. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.3.18 Группа параметров LUNA_MATCHER_PROXY_ADDRESS

Данная группа параметров задает настройки подключения к сервису Python Matcher Proxy.

12.3.18.1 origin

Параметр задает протокол, IP адрес и порт сервиса Python Matcher Proxy.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Python Matcher Proxy, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Python Matcher Proxy.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5110`.

12.3.18.2 api_version

Параметр задает версию API сервиса Python Matcher Proxy. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.3.19 Группа параметров LUNA_LICENSES_ADDRESS

Данная группа параметров задает настройки подключения к сервису Licenses.

12.3.19.1 origin

Параметр задает протокол, IP адрес и порт сервиса Licenses.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Licenses, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Licenses.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5120`.

12.3.19.2 api_version

Параметр задает версию API сервиса Licenses. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.3.20 Группа параметров LUNA_LAMBDA_ADDRESS

Данная группа параметров задает настройки подключения к сервису Lambda.

12.3.20.1 origin

Параметр задает протокол, IP адрес и порт сервиса Lambda.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Lambda, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Lambda.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5210`.

12.3.20.2 api_version

Параметр задает версию API сервиса Lambda. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.3.21 Группа параметров LUNA_ADMIN_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Admin.

12.3.21.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Admin. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.3.21.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.3.21.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.3.21.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.3.22 Группа параметров LUNA_STREAMS_RETRANSLATOR_ADDRESS

Данная группа параметров задает настройки подключения к сервису Streams Reatranslator.

12.3.22.1 origin

Параметр задает протокол, IP адрес и порт сервиса Streams Reatranslator.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Streams Reatranslator, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Streams Reatranslator.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5250`.

12.3.22.2 api_version

Параметр задает версию API сервиса Streams Reatranslator. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.3.23 Группа параметров ADDITIONAL_SERVICES_USAGE

12.3.23.1 luna_events

Параметр задает возможность использования сервиса Events.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.3.23.2 luna_tasks

Параметр задает возможность использования сервиса Tasks.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.3.23.3 luna_faces

Параметр задает возможность использования сервиса Faces.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.3.23.4 luna_handlers

Параметр задает возможность использования сервиса Handlers.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.3.23.5 luna_sender

Параметр задает возможность использования сервиса Sender.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.3.23.6 luna_matcher_proxy

Параметр задает возможность использования сервиса Python Matcher Proxy.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.3.23.7 luna_image_store

Параметр задает возможность использования сервиса Image Store.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.3.23.8 luna_lambda

Параметр задает возможность использования сервиса Lambda.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.3.23.9 luna_video_manager

Параметр задает возможность использования сервиса Video Manager.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.3.23.10 luna_video_agent

Параметр задает возможность использования сервиса Video Agent.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.3.23.11 luna_streams_retranslator

Параметр задает возможность использования сервиса Streams Retranslator.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.3.24 Группа параметров LUNA_ADMIN_HTTP_SETTINGS

В данной группе параметров содержатся настройки, отвечающие за обработку HTTP-подключений. См. подробную информацию по следующей ссылке: https://sanic.dev/en/guide/deployment/configuration.html#builtin-values

12.3.24.1 request_timeout

Параметр задает продолжительность времени между моментом, когда новое открытое TCP-соединение передается на сервер, и моментом, когда получен весь HTTP-запрос.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.3.24.2 response_timeout

Параметр задает продолжительность времени между моментом, когда сервер передает HTTP-запрос приложению, и моментом, когда HTTP-ответ отправляется клиенту.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `600`.

12.3.24.3 request_max_size

Параметр задает максимальный размер запроса.

Формат задания настройки — `integer` (байты).

Значение по умолчанию — `1073741824`.

12.3.24.4 keep_alive_timeout

Параметр задает тайм-аут поддержания активности HTTP.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `15`.

12.3.25 Группа параметров LUNA_SERVICE_METRICS

Данная группа параметров включает и настраивает сбор метрик в формате Prometheus.

См. подробную информацию в разделе «Мониторинг».

12.3.25.1 enabled

Параметр включает сбор метрик.

Если сбор метрик отключен, то запрос к ресурсу `/metrics` вернет соответствующее сообщение.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.3.25.2 metrics_format

Параметр задает формат метрик.

На данный момент поддерживается только формат Prometheus.

См. официальную документацию Prometheus для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `prometheus`.

12.3.25.3 extra_labels

Параметр задает пользовательские типы лейблов.

Формат задания настройки — `label_name=label_value`.

Значение по умолчанию не задано.

12.3.26 Прочие

12.3.26.1 luna_admin_active_plugins

Параметр задает список плагинов, которые должен использовать сервис.

Имена задаются в следующем формате:

```
[
   "plugin_1",
   "plugin_2",
   "plugin_3"
]
```

> Список должен содержать имена файлов без расширения (.py).

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.4 Настройки сервиса Faces

Данный раздел описывает параметры сервиса Faces.

Настройку сервиса можно выполнить с помощью сервиса Configurator.

12.4.1 Группа параметров LUNA_CONFIGURATOR

Данная группа параметров задаёт настройки подключения сервиса Faces к сервису Configurator.

Данная группа параметров не будет видна в пользовательском интерфейсе сервиса Configurator. Параметры можно изменить только в конфигурационном файле «config.conf», расположенном в директории «/srv/luna_faces/configs/» соответствующего контейнера.

12.4.1.1 use_configurator

Параметр позволяет включить использование сервиса Configurator.

> С помощью сервиса Configurator упрощается настройка сервисов LP. Сервис сохраняет все необходимые конфигурации для всех сервисов LP в одном месте.

При отключенном параметре будут использованы настройки из файла «config.conf», расположенного в директории «/srv/luna_faces/configs/» соответствующего контейнера.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.4.1.2 luna_configurator_origin

Параметр задает протокол, IP адрес и порт сервиса Configurator.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5070`.

12.4.1.3 luna_configurator_api

Параметр задает версию API сервиса Configurator. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.4.2 Группа параметров LUNA_FACES_DB

В данной группе параметров задаются настройки подключения к базе данных сервиса Faces.

12.4.2.1 db_type

Параметр задает тип базы данных. Доступны следующие типы:

- «postgres» — тип СУБД PostgreSQL
- «oracle» — тип СУБД Oracle

Формат задания настройки — `string`.

Значение по умолчанию — `postgres`.

12.4.2.2 db_host

Параметр задает имя сервера (хост) базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.4.2.3 db_port

Параметр задает порт базы данных.

Порт по умолчанию для типа «postgres»- `5432`.

Порт по умолчанию для типа «oracle» — `1521`.

Формат задания настройки — `string`.

Значение по умолчанию — `5432`.

12.4.2.4 db_user

Параметр задает имя пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.4.2.5 db_password

Параметр задает пароль пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.4.2.6 db_name

Параметр задает имя базы данных для типа «postgres» или имя SID для типа «oracle».

Формат задания настройки — `string`.

Значение по умолчанию — `luna_faces`.

12.4.2.7 connection_pool_size

Параметр задает размер пула соединений к БД. Реальное количество подключений может быть больше значения данной настройки на 1.

При необходимости в настройке «max_connections» конфигурационного файла PostgreSQL можно задать максимальное количество одновременных подключений к серверу БД. См. раздел «Продвинутая настройка PostrgeSQL» для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `10`.

12.4.2.8 dsn

Параметр задает строку подключения DSN для соединения с БД.

> DSN — это строка подключения, которая идентифицирует и указывает на источник данных (базу данных), к которому нужно установить соединение.

В строке DSN могут задаваться такие настройки, как множественные хосты, аутентификационные данные, порт и другие параметры.

> Настройки зависят от типа БД. Множественные хосты поддерживаются только с PostgreSQL.

> По умолчанию параметр «dsn» не отображается во вкладке «Settings» в Configurator. Проверить список всех доступных параметров для группы настроек можно на вкладке «Limitations».

Пример:

```
{
"dsn": "luna:luna@postgres01:5432,postgres02:5432/luna_faces?some_option=
    some_value"
"db_settings": {
     "connection_pool_size": 5
     }
}
```

Здесь:

- «luna:luna» — имя пользователя и пароль для подключения к PostgreSQL;
- «@postgres01:5432,postgres02:5432» — список хостов и портов, разделенных запятыми. Это означает, что сервис будет пытаться подключиться к первому хосту («postgres01») на порту 5432. Если это не удастся, она попытается подключиться ко второму хосту («postgres02») также на порту 5432;
- «/luna_faces» — название базы данных;
- «?some_option=some_value» — опциональные параметры для подключения.

При необходимости можно комбинировать строку DSN и классические настройки, однако строка DSN является более приоритетной. Можно частично заполнить строку DSN (например, «postgres01,postgres02/luna_faces»), и тогда недостающие параметры будут заполнены из значений параметров «db_host», «db_port», «db_name», «db_user» и «db_password».

При запуске сервис создаст пул подключений к одному из доступных хостов DSN. В случае возникновения проблем с установлением соединения после нескольких неудачных попыток, сервис снова попытается настроить пул подключений к любому из доступных хостов DSN.

12.4.3 Группа параметров LUNA_ATTRIBUTES_DB

12.4.3.1 user

Параметр задает имя пользователя БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.4.3.2 password

Параметр задает пароль БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию не указывается.

12.4.3.3 host

Параметр задает IP-адрес БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.4.3.4 port

Параметр задает номер порта, на котором Redis ожидает входящие сетевые соединения и прослушивает их для выполнения команд от клиентов.

Формат задания настройки — `integer`.

Значение по умолчанию — `6379`.

12.4.3.5 sentinel > master_name

Параметр задает имя мастера базы данных Redis, который контролируется и управляется системой Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_attributes`.

12.4.3.6 sentinel > sentinels

Параметр задает список адресов и портов серверов Sentinel, которые будут использоваться клиентами для обнаружения и мониторинга БД Redis.

Формат задания настройки — `list > string`.

Значение по умолчанию — `[]`.

12.4.3.7 sentinel > user

Параметр задает имя пользователя сервера Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.4.3.8 sentinel > password

Параметр задает пароль пользователя сервера Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.4.4 Группа параметров ATTRIBUTES_STORAGE_POLICY

В данной группе параметров задаются настройки для сервиса Faces, связанные с хранением временных атрибутов.

12.4.4.1 default_ttl

Параметр задает время существования временных атрибутов по умолчанию.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `300`.

12.4.4.2 max_ttl

Параметр задает максимальное время существования временных атрибутов.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `86400`.

12.4.5 Группа параметров LUNA_LICENSES_ADDRESS

Данная группа параметров задает настройки подключения к сервису Licenses.

12.4.5.1 origin

Параметр задает протокол, IP адрес и порт сервиса Licenses.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Licenses, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Licenses.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5120`.

12.4.5.2 api_version

Параметр задает версию API сервиса Licenses. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.4.6 Группа параметров LUNA_FACES_LOGGER

Данная группа параметров задает настройки логирования.

12.4.6.1 log_level

Параметр задает уровень отладочной печати, по приоритету: «ERROR», «WARNING», «INFO», «DEBUG».

Формат задания настройки — `string`.

Значение по умолчанию — `INFO`.

12.4.6.2 log_time

Параметр задает формат времени, используемый в записях лога. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.4.6.3 log_to_stdout

Параметр позволяет отправлять логи в стандартный вывод (stdout).

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`

12.4.6.4 log_to_file

Параметр позволяет сохранять логи в файл. Директория с файлами логов указывается в параметре «folder_with_logs».

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.4.6.5 folder_with_logs

Параметр задает директорию, в которой хранятся логи. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить параметр «log_to_file».

Формат задания настройки — `string`.

Значение по умолчанию — `./`

Пример:

```
  "folder_with_logs": "/srv/logs"
```

12.4.6.6 max_log_file_size

Параметр задает максимальный размер файла лога в МБ перед выполнением его ротации (0 — не использовать ротацию).

Для использования данного параметра требуется включить параметр «log_to_file».

> При необходимости можно настроить ротацию логов Docker. См. раздел «Настройка ротации логов Docker» в руководстве по установке LUNA PLATFORM.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024`

12.4.6.7 multiline_stack_trace

Параметр включает многострочную трассировку стека в логах. Когда параметр включен, информация о стеке вызовов записывается в логах так, что каждый фрейм стека помещается на отдельной строке, что улучшает читаемость. Если параметр выключен, информация о стеке вызовов записывается на одной строке, что может сделать логи менее удобными для анализа.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.4.6.8 format

Параметр определяет формат выводимых логов. Доступны следующие значения:

- «default» — стандартный формат вывода логов LUNA PLATFORM
- «json» — вывод логов в формате json
- «ecs» — вывод логов в формате ECS (Elastic Common Schema)

При использовании значения «ecs» будут использоваться следующие поля:

- «http.response.status_code» — содержит код состояния ответа HTTP (200, 404, 500 и т.д.);
- «http.response.execution_time» — содержит информацию о времени, затраченном на выполнение запроса и получение ответа;
- «http.request.method» — содержит метод HTTP-запроса (GET, POST, PUT и т.д.);
- «url.path» — содержит путь в URL-адресе запроса;
- «error.code» — содержит код ошибки, если запрос завершается с ошибкой.

Формат задания настройки — `string`.

Значение по умолчанию — `default`.

12.4.7 Группа параметров LUNA_MONITORING

Данная группа параметров задает настройки мониторинга.

См. подробную информацию о мониторинге в разделе «Мониторинг».

12.4.7.1 storage_type

Тип хранилища для хранения данных мониторинга.

В настоящее время доступно только БД Influx.

Формат задания настройки — `string`.

Значение по умолчанию — `influx`.

12.4.7.2 send_data_for_monitoring

Параметр позволяет включить или отключить отправку данных для мониторинга в InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.4.7.3 use_ssl

Параметр позволяет использовать HTTPS для подключения к InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.4.7.4 organization

Параметр задает рабочую область InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.4.7.5 token

Параметр задает аутентификации InfluxDB 2.x.

Формат задания настройки — `string`.

12.4.7.6 bucket

Параметр задаёт имя бакета InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_monitoring`.

12.4.7.7 host

Параметр задаёт IP-адрес InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.4.7.8 port

Параметр задает порт InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `8086`.

12.4.7.9 flushing_period

Параметр задает частоту отправки данных мониторинга в InfluxDB.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `1`.

12.4.8 Группа параметров LUNA_FACES_HTTP_SETTINGS

В данной группе параметров содержатся настройки, отвечающие за обработку HTTP-подключений. См. подробную информацию по следующей ссылке: https://sanic.dev/en/guide/deployment/configuration.html#builtin-values

12.4.8.1 request_timeout

Параметр задает продолжительность времени между моментом, когда новое открытое TCP-соединение передается на сервер, и моментом, когда получен весь HTTP-запрос.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.4.8.2 response_timeout

Параметр задает продолжительность времени между моментом, когда сервер передает HTTP-запрос приложению, и моментом, когда HTTP-ответ отправляется клиенту.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `600`.

12.4.8.3 request_max_size

Параметр задает максимальный размер запроса.

Формат задания настройки — `integer` (байты).

Значение по умолчанию — `1073741824`.

12.4.8.4 keep_alive_timeout

Параметр задает тайм-аут поддержания активности HTTP.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `15`.

12.4.9 Группа параметров LUNA_SERVICE_METRICS

Данная группа параметров включает и настраивает сбор метрик в формате Prometheus.

См. подробную информацию в разделе «Мониторинг».

12.4.9.1 enabled

Параметр включает сбор метрик.

Если сбор метрик отключен, то запрос к ресурсу `/metrics` вернет соответствующее сообщение.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.4.9.2 metrics_format

Параметр задает формат метрик.

На данный момент поддерживается только формат Prometheus.

См. официальную документацию Prometheus для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `prometheus`.

12.4.9.3 extra_labels

Параметр задает пользовательские типы лейблов.

Формат задания настройки — `label_name=label_value`.

Значение по умолчанию не задано.

12.4.10 Группа параметров DESCRIPTOR_ENCRYPTION

Данная группа параметров задает настройки шифрования биометрических шаблонов для повышения безопасности и предотвращения несанкционированного использования.

См. подробную информацию в разделе «Шифрование биометрических шаблонов».

12.4.10.1 enabled

Параметр включает шифрование биометрических шаблонов.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.4.10.2 algorithm

Параметр задает название используемого алгоритма шифрования.

Формат задания настройки — `string`.

Значение по умолчанию — `aes256-gcm`.

12.4.10.3 params > source

Параметр задает название источника ключа шифрования. Поддерживаются два типа источников: `raw` и `vaultKV`.

Формат задания настройки — `string`.

Значение по умолчанию — `raw`.

12.4.10.4 params > key

Параметр задает ключ шифрования или учетные данные для его получения из указанного источника.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.4.11 Прочие

12.4.11.1 database_number

Параметр задает номер базы данных Faces в контексте настройки репликации.

> Репликация базы данных — это процесс создания и поддержания копий данных из одной базы данных в другой.

Значение «0» означает, что текущая база данных Faces не участвует в репликации. Иными словами, она не является источником данных для какой-либо другой базы данных, и данные в ней не реплицируются на другие базы данных.

См. подробную информацию о репликации данных в разделе «Replication» руководства разработчика сервиса Faces.

Формат задания настройки — `integer` («0», «1» или «2»).

Значение по умолчанию — `0`.

12.4.11.2 default_face_descriptor_version

Параметр задает используемую версию биометрического шаблона лица.

Формат задания настройки — `string`.

Значение по умолчанию — `59`.

Примечание. См. подробную информацию о версиях биометрических шаблонах в разделе «Нейронные сети».

12.4.11.3 use_material_views

Параметр позволяет использовать материализованные представления, что позволяет ускорить запрос к ресурсу «/lists/linkkeys» сервиса Faces.

> Материализованные представления — это объекты в базе данных, которые содержат результаты выполнения запроса к одной или нескольким таблицам. В отличие от обычных (виртуальных) представлений, материализованные представления хранят данные фактически в таблице, что позволяет ускорить выполнение запросов, особенно при сложных агрегирующих или вычислительных операциях.

См. принцип использования данного параметра в разделе «Materialized» руководства разработчика сервиса Faces.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `0`.

12.4.11.4 luna_faces_db_ping_max_count

Параметр задает максимальное количество проверок соединения с базой данных для каждого запроса.

Если БД недоступна, то возвращается ошибка. Если выставлено значение <= 0, то проверки не выполняются.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.4.11.5 storage_time

Параметр задает формат времени, используемый для записей в базе данных. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.4.11.6 luna_faces_active_plugins

Параметр задает список плагинов, которые должен использовать сервис.

Имена задаются в следующем формате:

```
[
   "plugin_1",
   "plugin_2",
   "plugin_3"
]
```

> Список должен содержать имена файлов без расширения (.py).

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.5 Настройки сервиса Image Store

Данный раздел описывает параметры сервиса Image Store.

Настройку сервиса можно выполнить с помощью сервиса Configurator.

12.5.1 Группа параметров LUNA_CONFIGURATOR

Данная группа параметров задаёт настройки подключения сервиса Image Store к сервису Configurator.

Данная группа параметров не будет видна в пользовательском интерфейсе сервиса Configurator. Параметры можно изменить только в конфигурационном файле «config.conf», расположенном в директории «/srv/luna_image_store/configs/» соответствующего контейнера.

12.5.1.1 use_configurator

Параметр позволяет включить использование сервиса Configurator.

> С помощью сервиса Configurator упрощается настройка сервисов LP. Сервис сохраняет все необходимые конфигурации для всех сервисов LP в одном месте.

При отключенном параметре будут использованы настройки из файла «config.conf», расположенного в директории «/srv/luna_image_store/configs/» соответствующего контейнера.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.5.1.2 luna_configurator_origin

Параметр задает протокол, IP адрес и порт сервиса Configurator.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5070`.

12.5.1.3 luna_configurator_api

Параметр задает версию API сервиса Configurator. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.5.2 Группа параметров LUNA_IMAGE_STORE_LOGGER

Данная группа параметров задает настройки логирования.

12.5.2.1 log_level

Параметр задает уровень отладочной печати, по приоритету: «ERROR», «WARNING», «INFO», «DEBUG».

Формат задания настройки — `string`.

Значение по умолчанию — `INFO`.

12.5.2.2 log_time

Параметр задает формат времени, используемый в записях лога. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.5.2.3 log_to_stdout

Параметр позволяет отправлять логи в стандартный вывод (stdout).

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`

12.5.2.4 log_to_file

Параметр позволяет сохранять логи в файл. Директория с файлами логов указывается в параметре «folder_with_logs».

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.5.2.5 folder_with_logs

Параметр задает директорию, в которой хранятся логи. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить параметр «log_to_file».

Формат задания настройки — `string`.

Значение по умолчанию — `./`

Пример:

```
  "folder_with_logs": "/srv/logs"
```

12.5.2.6 max_log_file_size

Параметр задает максимальный размер файла лога в МБ перед выполнением его ротации (0 — не использовать ротацию).

Для использования данного параметра требуется включить параметр «log_to_file».

> При необходимости можно настроить ротацию логов Docker. См. раздел «Настройка ротации логов Docker» в руководстве по установке LUNA PLATFORM.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024`

12.5.2.7 multiline_stack_trace

Параметр включает многострочную трассировку стека в логах. Когда параметр включен, информация о стеке вызовов записывается в логах так, что каждый фрейм стека помещается на отдельной строке, что улучшает читаемость. Если параметр выключен, информация о стеке вызовов записывается на одной строке, что может сделать логи менее удобными для анализа.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.5.2.8 format

Параметр определяет формат выводимых логов. Доступны следующие значения:

- «default» — стандартный формат вывода логов LUNA PLATFORM
- «json» — вывод логов в формате json
- «ecs» — вывод логов в формате ECS (Elastic Common Schema)

При использовании значения «ecs» будут использоваться следующие поля:

- «http.response.status_code» — содержит код состояния ответа HTTP (200, 404, 500 и т.д.);
- «http.response.execution_time» — содержит информацию о времени, затраченном на выполнение запроса и получение ответа;
- «http.request.method» — содержит метод HTTP-запроса (GET, POST, PUT и т.д.);
- «url.path» — содержит путь в URL-адресе запроса;
- «error.code» — содержит код ошибки, если запрос завершается с ошибкой.

Формат задания настройки — `string`.

Значение по умолчанию — `default`.

12.5.3 Группа параметров LUNA_MONITORING

Данная группа параметров задает настройки мониторинга.

См. подробную информацию о мониторинге в разделе «Мониторинг».

12.5.3.1 storage_type

Тип хранилища для хранения данных мониторинга.

В настоящее время доступно только БД Influx.

Формат задания настройки — `string`.

Значение по умолчанию — `influx`.

12.5.3.2 send_data_for_monitoring

Параметр позволяет включить или отключить отправку данных для мониторинга в InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.5.3.3 use_ssl

Параметр позволяет использовать HTTPS для подключения к InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.5.3.4 organization

Параметр задает рабочую область InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.5.3.5 token

Параметр задает аутентификации InfluxDB 2.x.

Формат задания настройки — `string`.

12.5.3.6 bucket

Параметр задаёт имя бакета InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_monitoring`.

12.5.3.7 host

Параметр задаёт IP-адрес InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.5.3.8 port

Параметр задает порт InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `8086`.

12.5.3.9 flushing_period

Параметр задает частоту отправки данных мониторинга в InfluxDB.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `1`.

12.5.4 Группа параметров LUNA_IMAGE_STORE_HTTP_SETTINGS

В данной группе параметров содержатся настройки, отвечающие за обработку HTTP-подключений. См. подробную информацию по следующей ссылке: https://sanic.dev/en/guide/deployment/configuration.html#builtin-values

12.5.4.1 request_timeout

Параметр задает продолжительность времени между моментом, когда новое открытое TCP-соединение передается на сервер, и моментом, когда получен весь HTTP-запрос.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.5.4.2 response_timeout

Параметр задает продолжительность времени между моментом, когда сервер передает HTTP-запрос приложению, и моментом, когда HTTP-ответ отправляется клиенту.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `600`.

12.5.4.3 request_max_size

Параметр задает максимальный размер запроса.

Формат задания настройки — `integer` (байты).

Значение по умолчанию — `1073741824`.

12.5.4.4 keep_alive_timeout

Параметр задает тайм-аут поддержания активности HTTP.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `15`.

12.5.5 S3

Данная группа параметров содержит параметры для взаимодействия с S3-хранилищем.

Для использования данных параметров требуется включить использование S3-хранилища в параметре «storage_type».

Для использования S3 нужно обязательно указать следующие параметры:

- «host»
- «aws_public_access_key»
- «aws_secret_access_key»
- «authorization_signature»

12.5.5.1 host

Параметр задает URL-адрес, по которому будет установлено соединение с S3-хранилищем.

Формат задания настройки — `string`.

Значение по умолчанию — `http://localhost:7480`.

12.5.5.2 region

Параметр задает регион, который будет использоваться при взаимодействии с S3-хранилищем.

> Регион может влиять на доступность и производительность различных ресурсов в AWS S3.

Формат задания настройки — `string`.

Значение по умолчанию не указано.

12.5.5.3 aws_public_access_key

Параметр задает публичный ключ доступа, который используется для аутентификации при доступе к S3-хранилищу. Этот ключ предоставляется AWS и используется для идентификации клиента.

Формат задания настройки — `string`.

Значение по умолчанию не указано.

12.5.5.4 aws_secret_access_key

Параметр задает секретный ключа доступа, который совместно с публичным ключом обеспечивает аутентификацию при доступе к S3-хранилищу.

Формат задания настройки — `string`.

Значение по умолчанию не указано.

12.5.5.5 authorization_signature

Параметр определяет метод, используемый для создания подписи аутентификации при выполнении операций с S3.

Можно указать два значения:

- «s3v4» — использование алгоритма подписи AWS S3 Version 4
- «s3v2» — использование алгоритма подписи AWS S3 Version 2

Формат задания настройки — `string`.

Значение по умолчанию — `s3v4`.

12.5.5.6 request_timeout

Параметр задает максимальное время, в течение которого запрос к S3-хранилищу должен быть выполнен. Если запрос не завершится в пределах этого времени, он будет отменен.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.5.5.7 connect_timeout

Параметр задает максимальное время ожидания для установления соединения с S3-хранилищем. Если соединение не устанавливается в пределах этого времени, то оно будет считаться неудачным.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `30`.

12.5.5.8 verify_ssl

Параметр определяет, следует ли выполнять проверку SSL-сертификата при установлении защищенного (HTTPS) соединения с S3-хранилищем. Если значение равно `true`, то SSL-сертификат будет проверен. Если значение равно `false`, то проверка будет отключена, что может вести к проблемам с безопасностью.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.5.6 Группа параметров LUNA_SERVICE_METRICS

Данная группа параметров включает и настраивает сбор метрик в формате Prometheus.

См. подробную информацию в разделе «Мониторинг».

12.5.6.1 enabled

Параметр включает сбор метрик.

Если сбор метрик отключен, то запрос к ресурсу `/metrics` вернет соответствующее сообщение.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.5.6.2 metrics_format

Параметр задает формат метрик.

На данный момент поддерживается только формат Prometheus.

См. официальную документацию Prometheus для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `prometheus`.

12.5.6.3 extra_labels

Параметр задает пользовательские типы лейблов.

Формат задания настройки — `label_name=label_value`.

Значение по умолчанию не задано.

12.5.7 Прочие

12.5.7.1 storage_type

Параметр задает тип хранилища Image Store. Доступно два типа:

- «S3» — использование облачного хранилища S3
- «LOCAL» — использование локального хранилища (SSD или HDD)

При использовании S3 необходимо дополнительно задать настройки из группы параметров «S3».

При использовании локального хранилища необходимо дополнительно путь до директории с изображениями в настройке «local_storage».

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.5.7.2 local_storage

Параметр задает путь и имя директории для хранения изображений. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить использование локального хранилище в параметре «storage_type».

Формат задания настройки — `string`.

Значение по умолчанию — `./local_storage`.

12.5.7.3 default_image_extension

Параметр задает формат изображения биометрического образца, получаемого после нормализации исходного изображения. Доступно два значения:

- «jpg» — сохранять биометрический образец в формате JPG
- «png» — сохранять биометрический образец в формате PNG

Обратите внимание, что формат передаваемого исходного изображения не зависит от формата биометрического образца, полученного после обработки. Если значение данной настройки равно «jpg», то для исходного изображения формата PNG все равно будет сохранен биометрический образец в формате JPG.

Формат задания настройки — `string`.

Значение по умолчанию — `jpg`.

12.5.7.4 luna_image_store_active_plugins

Параметр задает список плагинов, которые должен использовать сервис.

Имена задаются в следующем формате:

```
[
   "plugin_1",
   "plugin_2",
   "plugin_3"
]
```

> Список должен содержать имена файлов без расширения (.py).

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.6 Настройки сервиса Tasks

Данный раздел описывает параметры сервиса Tasks.

Настройку сервиса можно выполнить с помощью сервиса Configurator.

12.6.1 Группа параметров LUNA_CONFIGURATOR

Данная группа параметров задаёт настройки подключения сервиса Tasks к сервису Configurator.

Данная группа параметров не будет видна в пользовательском интерфейсе сервиса Configurator. Параметры можно изменить только в конфигурационном файле «config.conf», расположенном в директории «/srv/luna_tasks/configs/» соответствующего контейнера.

12.6.1.1 use_configurator

Параметр позволяет включить использование сервиса Configurator.

> С помощью сервиса Configurator упрощается настройка сервисов LP. Сервис сохраняет все необходимые конфигурации для всех сервисов LP в одном месте.

При отключенном параметре будут использованы настройки из файла «config.conf», расположенного в директории «/srv/luna_tasks/configs/» соответствующего контейнера.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.6.1.2 luna_configurator_origin

Параметр задает протокол, IP адрес и порт сервиса Configurator.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5070`.

12.6.1.3 luna_configurator_api

Параметр задает версию API сервиса Configurator. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.6.2 Группа параметров LUNA_TASKS_DB

В данной группе параметров задаются настройки подключения к базе данных сервиса Tasks.

12.6.2.1 db_type

Параметр задает тип базы данных. Доступны следующие типы:

- «postgres» — тип СУБД PostgreSQL
- «oracle» — тип СУБД Oracle

Формат задания настройки — `string`.

Значение по умолчанию — `postgres`.

12.6.2.2 db_host

Параметр задает имя сервера (хост) базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.6.2.3 db_port

Параметр задает порт базы данных.

Порт по умолчанию для типа «postgres»- `5432`.

Порт по умолчанию для типа «oracle» — `1521`.

Формат задания настройки — `string`.

Значение по умолчанию — `5432`.

12.6.2.4 db_user

Параметр задает имя пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.6.2.5 db_password

Параметр задает пароль пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.6.2.6 db_name

Параметр задает имя базы данных для типа «postgres» или имя SID для типа «oracle».

Формат задания настройки — `string`.

Значение по умолчанию — `luna_tasks`.

12.6.2.7 connection_pool_size

Параметр задает размер пула соединений к БД. Реальное количество подключений может быть больше значения данной настройки на 1.

При необходимости в настройке «max_connections» конфигурационного файла PostgreSQL можно задать максимальное количество одновременных подключений к серверу БД. См. раздел «Продвинутая настройка PostrgeSQL» для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `10`.

12.6.2.8 dsn

Параметр задает строку подключения DSN для соединения с БД.

> DSN — это строка подключения, которая идентифицирует и указывает на источник данных (базу данных), к которому нужно установить соединение.

В строке DSN могут задаваться такие настройки, как множественные хосты, аутентификационные данные, порт и другие параметры.

> Настройки зависят от типа БД. Множественные хосты поддерживаются только с PostgreSQL.

> По умолчанию параметр «dsn» не отображается во вкладке «Settings» в Configurator. Проверить список всех доступных параметров для группы настроек можно на вкладке «Limitations».

Пример:

```
{
"dsn": "luna:luna@postgres01:5432,postgres02:5432/luna_tasks?some_option=
    some_value"
"db_settings": {
     "connection_pool_size": 5
     }
}
```

Здесь:

- «luna:luna» — имя пользователя и пароль для подключения к PostgreSQL;
- «@postgres01:5432,postgres02:5432» — список хостов и портов, разделенных запятыми. Это означает, что сервис будет пытаться подключиться к первому хосту («postgres01») на порту 5432. Если это не удастся, она попытается подключиться ко второму хосту («postgres02») также на порту 5432;
- «/luna_tasks» — название базы данных;
- «?some_option=some_value» — опциональные параметры для подключения.

При необходимости можно комбинировать строку DSN и классические настройки, однако строка DSN является более приоритетной. Можно частично заполнить строку DSN (например, «postgres01,postgres02/luna_tasks»), и тогда недостающие параметры будут заполнены из значений параметров «db_host», «db_port», «db_name», «db_user» и «db_password».

При запуске сервис создаст пул подключений к одному из доступных хостов DSN. В случае возникновения проблем с установлением соединения после нескольких неудачных попыток, сервис снова попытается настроить пул подключений к любому из доступных хостов DSN.

12.6.3 Группа параметров LUNA_TASKS_LOGGER

Данная группа параметров задает настройки логирования.

12.6.3.1 log_level

Параметр задает уровень отладочной печати, по приоритету: «ERROR», «WARNING», «INFO», «DEBUG».

Формат задания настройки — `string`.

Значение по умолчанию — `INFO`.

12.6.3.2 log_time

Параметр задает формат времени, используемый в записях лога. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.6.3.3 log_to_stdout

Параметр позволяет отправлять логи в стандартный вывод (stdout).

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`

12.6.3.4 log_to_file

Параметр позволяет сохранять логи в файл. Директория с файлами логов указывается в параметре «folder_with_logs».

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.6.3.5 folder_with_logs

Параметр задает директорию, в которой хранятся логи. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить параметр «log_to_file».

Формат задания настройки — `string`.

Значение по умолчанию — `./`

Пример:

```
  "folder_with_logs": "/srv/logs"
```

12.6.3.6 max_log_file_size

Параметр задает максимальный размер файла лога в МБ перед выполнением его ротации (0 — не использовать ротацию).

Для использования данного параметра требуется включить параметр «log_to_file».

> При необходимости можно настроить ротацию логов Docker. См. раздел «Настройка ротации логов Docker» в руководстве по установке LUNA PLATFORM.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024`

12.6.3.7 multiline_stack_trace

Параметр включает многострочную трассировку стека в логах. Когда параметр включен, информация о стеке вызовов записывается в логах так, что каждый фрейм стека помещается на отдельной строке, что улучшает читаемость. Если параметр выключен, информация о стеке вызовов записывается на одной строке, что может сделать логи менее удобными для анализа.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.6.3.8 format

Параметр определяет формат выводимых логов. Доступны следующие значения:

- «default» — стандартный формат вывода логов LUNA PLATFORM
- «json» — вывод логов в формате json

- «ecs» — вывод логов в формате ECS (Elastic Common Schema)

При использовании значения «ecs» будут использоваться следующие поля:

- «http.response.status_code» — содержит код состояния ответа HTTP (200, 404, 500 и т.д.);
- «http.response.execution_time» — содержит информацию о времени, затраченном на выполнение запроса и получение ответа;
- «http.request.method» — содержит метод HTTP-запроса (GET, POST, PUT и т.д.);
- «url.path» — содержит путь в URL-адресе запроса;
- «error.code» — содержит код ошибки, если запрос завершается с ошибкой.

Формат задания настройки — `string`.

Значение по умолчанию — `default`.

12.6.4 Группа параметров LUNA_MONITORING

Данная группа параметров задает настройки мониторинга.

См. подробную информацию о мониторинге в разделе «Мониторинг».

12.6.4.1 storage_type

Тип хранилища для хранения данных мониторинга.

В настоящее время доступно только БД Influx.

Формат задания настройки — `string`.

Значение по умолчанию — `influx`.

12.6.4.2 send_data_for_monitoring

Параметр позволяет включить или отключить отправку данных для мониторинга в InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.6.4.3 use_ssl

Параметр позволяет использовать HTTPS для подключения к InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.6.4.4 organization

Параметр задает рабочую область InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.6.4.5 token

Параметр задает аутентификации InfluxDB 2.x.

Формат задания настройки — `string`.

12.6.4.6 bucket

Параметр задаёт имя бакета InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_monitoring`.

12.6.4.7 host

Параметр задаёт IP-адрес InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.6.4.8 port

Параметр задает порт InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `8086`.

12.6.4.9 flushing_period

Параметр задает частоту отправки данных мониторинга в InfluxDB.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `1`.

12.6.5 Группа параметров LUNA_IMAGE_STORE_FACES_SAMPLES_ADDRESS

В данной группе параметров задаются настройки бакета для хранения БО лиц.

12.6.5.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.6.5.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.6.5.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-samples`.

12.6.6 Группа параметров LUNA_IMAGE_STORE_BODIES_SAMPLES_ADDRESS

В данной группе параметров задаются настройки бакета для хранения БО тел.

12.6.6.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.6.6.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — 1.

12.6.6.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-bodies-samples`.

12.6.7 Группа параметров LUNA_IMAGE_STORE_IMAGES_ADDRESS

В данной группе параметров задаются настройки бакета для хранения исходных изображений.

12.6.7.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.6.7.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.6.7.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-image-origin`.

12.6.8 Группа параметров LUNA_IMAGE_STORE_FACES_SAMPLES_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к бакету, хранящему БО лиц.

12.6.8.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к бакету, хранящему БО лиц. Это время ожидания, в течение которого клиент пытается установить соединение с бакетом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `30`.

12.6.8.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.6.9 Группа параметров LUNA_IMAGE_STORE_BODIES_SAMPLES_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к бакету, хранящему БО тел.

12.6.9.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к бакету, хранящему БО тел. Это время ожидания, в течение которого клиент пытается установить соединение с бакетом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `30`.

12.6.9.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.6.10 Группа параметров LUNA_IMAGE_STORE_IMAGES_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к бакету, хранящему исходных изображений.

12.6.10.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к бакету, хранящему исходных изображений. Это время ожидания, в течение которого клиент пытается установить соединение с бакетом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `30`.

12.6.10.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.6.11 Группа параметров LUNA_IMAGE_STORE_TASK_RESULT_ADDRESS

В данной группе параметров задаются настройки бакета для хранения результатов задач.

12.6.11.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.6.11.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.6.11.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `task-result`.

12.6.12 Группа параметров LUNA_IMAGE_STORE_TASK_RESULT_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к бакету, хранящему результатов задач.

12.6.12.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к бакету, хранящему результатов задач. Это время ожидания, в течение которого клиент пытается установить соединение с бакетом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `30`.

12.6.12.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.6.13 Группа параметров LUNA_IMAGE_STORE_OBJECTS_ADDRESS

В данной группе параметров задаются настройки бакета для хранения объектов.

12.6.13.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.6.13.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.6.13.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-objects`.

12.6.14 Группа параметров LUNA_TASKS_LOAD_EXTERNAL_ARCHIVE_TIMEOUTS

Данная группа параметров задает таймауты выполнения HTTP-запросов к внешним ресурсам для загрузки архивов. Каждый параметр определяет максимальное количество времени, которое ожидается для выполнения определенной операции в рамках HTTP-запроса.

12.6.14.1 connect

Параметр задает таймаут для установки соединения с внешним ресурсом. Если соединение не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.6.14.2 request

Параметр задает таймаут для ожидания ответа на HTTP-запрос, который отправляется для получения архива с внешнего ресурса.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `100`.

12.6.14.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.6.14.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `100`.

12.6.15 Группа параметров LUNA_FACES_ADDRESS

Данная группа параметров задает настройки подключения к сервису Faces.

12.6.15.1 origin

Параметр задает протокол, IP адрес и порт сервиса Faces.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Faces, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Faces.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5030`.

12.6.15.2 api_version

Параметр задает версию API сервиса Faces. Доступная версия API — «3».

Формат задания настройки — `integer`.

Значение по умолчанию — `3`.

12.6.16 Группа параметров LUNA_FACES_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Faces.

12.6.16.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Faces. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.6.16.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.6.16.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.6.16.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.6.17 Группа параметров LUNA_PYTHON_MATCHER_ADDRESS

Данная группа параметров задает настройки подключения к сервису Python Matcher.

12.6.17.1 origin

Параметр задает протокол, IP адрес и порт сервиса Python Matcher.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Python Matcher, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Python Matcher.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5100`.

12.6.17.2 api_version

Параметр задает версию API сервиса Python Matcher. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.6.18 Группа параметров LUNA_MATCHER_PROXY_ADDRESS

Данная группа параметров задает настройки подключения к сервису Python Matcher Proxy.

12.6.18.1 origin

Параметр задает протокол, IP адрес и порт сервиса Python Matcher Proxy.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Python Matcher Proxy, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Python Matcher Proxy.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5110`.

12.6.18.2 api_version

Параметр задает версию API сервиса Python Matcher Proxy. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.6.19 Группа параметров LUNA_TASKS_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Tasks.

12.6.19.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Tasks. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.6.19.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.6.19.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.6.19.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.6.20 Группа параметров LUNA_EVENTS_ADDRESS

Данная группа параметров задает настройки подключения к сервису Events.

12.6.20.1 origin

Параметр задает протокол, IP адрес и порт сервиса Events.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Events, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Events.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5040`.

12.6.20.2 api_version

Параметр задает версию API сервиса Events. Доступная версия API — «2».

Формат задания настройки — `integer`.

Значение по умолчанию — `2`.

12.6.21 Группа параметров LUNA_EVENTS_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Events.

12.6.21.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Events. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.6.21.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.6.21.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.6.21.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.6.22 Группа параметров LUNA_HANDLERS_ADDRESS

Данная группа параметров задает настройки подключения к сервису Handlers.

12.6.22.1 origin

Параметр задает протокол, IP адрес и порт сервиса Handlers.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Handlers, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Handlers.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5090`.

12.6.22.2 api_version

Параметр задает версию API сервиса Handlers. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.6.23 Группа параметров LUNA_HANDLERS_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Handlers.

12.6.23.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Handlers. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.6.23.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.6.23.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.6.23.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.6.24 Группа параметров PLATFORM_LIMITS

Данная группа параметров определяет ограничения, связанные с операциями сравнения лиц, событий или атрибутов. Эти параметры управляют количеством объектов, которые могут быть включены в операцию сравнения, а также ограничивают количество результатов, возвращаемых в ответе.

Эти параметры полезны для балансировки производительности и использования ресурсов.

Параметры из подгруппы «match» определяют ограничения, связанные с операциями сравнения.

Параметры из подгруппы «cross_match» определяют ограничения, связанные с операциями перекрестного сравнения (см. описание задачи «Cross-matching»).

> Для сервиса Tasks подгруппа «match» не используется. Настройки ограничения сравнения операций обычного сравнения задаются в группе параметров «PLATFORM_LIMITS» сервиса Python Matcher.

12.6.24.1 cross_match > short_array_filter_limit

Параметр задает максимальное количество объектов (максимальный размер массива), указанных во всех фильтрах, кроме фильтров «face_ids», «event_ids» и «attribute_ids» в задаче Cross-matching.

> Например, максимальный размер массива для фильтров «sources», «tags», «cities» и др. В спецификации OpenAPI такие фильтры помечены примером `[ 1 .. 1000 ] items`.

Формат задания настройки — `integer`.

Значение по умолчанию — `1000`.

12.6.24.2 cross_match > array_filter_limit

Параметр задает максимальное количество объектов (максимальный размер массива), указанных в фильтрах «face_ids», «event_ids» и «attribute_ids» в задаче Cross-matching.

> В спецификации OpenAPI такие фильтры помечены примером `[ 1 .. 20000 ] items`.

Формат задания настройки — `integer`.

Значение по умолчанию — `20000`.

12.6.24.3 cross_match > result_candidate_limit

Параметр задает максимальное количество результатов сравнения (максимальный размер массива) для списка кандидатов, возвращаемых в ответе запроса на получение результата задачи Cross-matching.

В теле запроса на сравнение также можно ограничить количество возвращаемых кандидатов в ответе с помощью параметра «limit», однако параметр «result_candidate_limit» имеет более высокий приоритет. Это означает, что при значении «result_candidate_limit» = 4 и «limit» = 6, будет возвращена ошибка с кодом «31007».

Формат задания настройки — `integer`.

Значение по умолчанию — `20000`.

12.6.24.4 cross_match > general_limit

Параметр задает общее ограничение на количество перекрестных сравнений. Он применяется к произведению количества кандидатов и эталонов, которые участвуют в операции сравнения с учетом фильтров. Если произведение превышает значение «general_limit», запрос завершится ошибкой.

Например, если значение «general_limit» равно «20», количество кандидатов-лиц в массиве «face_ids» равно «7», количество эталонов-лиц в массиве «face_ids» равно «5» и все остальные фильтры не сокращают реальное количество кандидатов-лиц, то запрос на перекрестное сравнение не будет выполнен, т.к. общее количество перекрестных сравнений будет равно «35» (5 x 7). Если же какой-то фильтр сокращает общее количество кандидатов-лиц на «3», то запрос будет выполнен успешно, т.к. общее количество перекрестных сравнений будет равно «20» (5 x (7 - 3)).

Формат задания настройки — `integer`.

Значение по умолчанию — `100000000`.

12.6.25 Группа параметров EXTERNAL_LUNA_API_ADDRESS

Данная группа параметров предназначена для корректной обработки ссылок на объекты, созданные с помощью ресурсов «/images» и «/objects» в сервисе API. В данной секции задается адрес и версия API сервиса API.

Если параметре «content» > «source» > «reference» ресурса «/tasks/estimator» указывается URL-адрес и версия сервиса API до объекта типа «object» (ZIP-архив), совпадающие с адресом и версией API из секции «EXTERNAL_LUNA_API_ADDRESS» сервиса Tasks, то данный объект будет загружаться с помощью сервиса Image Store напрямую, а не отправлять запрос в сервис API с последующим перенаправлением в сервис Image Store.

Пример формата: `http://10.15.3.144:5000/6/images/141d2706-8baf-433b-82eb-8c7fada847da`, где значение `http://10.15.3.144:5000` должно совпадать со значением из

настройки «origin», а значение 6 должно совпадать со значением настройки `api_version` секции «EXTERNAL_LUNA_API_ADDRESS».

Для избежания ошибок необходимо настроить данную секцию в настройках Handlers перед использованием URL-адресов до объектов типа «objects» или «images» в качестве источника входных данных.

12.6.25.1 origin

Параметр задает протокол, IP адрес и порт сервиса API.

См. описание логики работы в разделе «Группа параметров EXTERNAL_LUNA_API_ADDRESS».

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5000`.

12.6.25.2 api_version

Параметр задает версию API сервиса API.

См. описание логики работы в разделе «Группа параметров EXTERNAL_LUNA_API_ADDRESS».

Формат задания настройки — `integer`.

Значение по умолчанию — `6`.

12.6.26 Группа параметров LUNA_TASKS_HTTP_SETTINGS

В данной группе параметров содержатся настройки, отвечающие за обработку HTTP-подключений. См. подробную информацию по следующей ссылке: https://sanic.dev/en/guide/deployment/configuration.html#builtin-values

12.6.26.1 request_timeout

Параметр задает продолжительность времени между моментом, когда новое открытое TCP-соединение передается на сервер, и моментом, когда получен весь HTTP-запрос.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.6.26.2 response_timeout

Параметр задает продолжительность времени между моментом, когда сервер передает HTTP-запрос приложению, и моментом, когда HTTP-ответ отправляется клиенту.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `600`.

12.6.26.3 request_max_size

Параметр задает максимальный размер запроса.

Формат задания настройки — `integer` (байты).

Значение по умолчанию — `1073741824`.

12.6.26.4 keep_alive_timeout

Параметр задает тайм-аут поддержания активности HTTP.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `15`.

12.6.27 Группа параметров TASKS_REDIS_DB_ADDRESS

12.6.27.1 user

Параметр задает имя пользователя БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.6.27.2 password

Параметр задает пароль БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию не указывается.

12.6.27.3 host

Параметр задает IP-адрес БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.6.27.4 port

Параметр задает номер порта, на котором Redis ожидает входящие сетевые соединения и прослушивает их для выполнения команд от клиентов.

Формат задания настройки — `integer`.

Значение по умолчанию — `6379`.

12.6.27.5 sentinel > master_name

Параметр задает имя мастера базы данных Redis, который контролируется и управляется системой Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_tasks`.

12.6.27.6 sentinel > sentinels

Параметр задает список адресов и портов серверов Sentinel, которые будут использоваться клиентами для обнаружения и мониторинга БД Redis.

Формат задания настройки — `list > string`.

Значение по умолчанию — `[]`.

12.6.27.7 sentinel > user

Параметр задает имя пользователя сервера Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.6.27.8 sentinel > password

Параметр задает пароль пользователя сервера Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.6.28 Группа параметров ADDITIONAL_SERVICES_USAGE

12.6.28.1 luna_events

Параметр задает возможность использования сервиса Events.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.6.28.2 luna_faces

Параметр задает возможность использования сервиса Faces.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.6.28.3 luna_handlers

Параметр задает возможность использования сервиса Handlers.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.6.28.4 luna_sender

Параметр задает возможность использования сервиса Sender.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.6.28.5 luna_matcher_proxy

Параметр задает возможность использования сервиса Python Matcher Proxy.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.6.28.6 luna_image_store

Параметр задает возможность использования сервиса Image Store.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.6.28.7 luna_lambda

Параметр задает возможность использования сервиса Lambda.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.6.28.8 luna_video_manager

Параметр задает возможность использования сервиса Video Manager.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.6.28.9 luna_video_agent

Параметр задает возможность использования сервиса Video Agent.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.6.28.10 luna_streams_retranslator

Параметр задает возможность использования сервиса Streams Retranslator.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.6.29 Группа параметров LUNA_SERVICE_METRICS

Данная группа параметров включает и настраивает сбор метрик в формате Prometheus.

См. подробную информацию в разделе «Мониторинг».

12.6.29.1 enabled

Параметр включает сбор метрик.

Если сбор метрик отключен, то запрос к ресурсу `/metrics` вернет соответствующее сообщение.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.6.29.2 metrics_format

Параметр задает формат метрик.

На данный момент поддерживается только формат Prometheus.

См. официальную документацию Prometheus для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `prometheus`.

12.6.29.3 extra_labels

Параметр задает пользовательские типы лейблов.

Формат задания настройки — `label_name=label_value`.

Значение по умолчанию не задано.

12.6.30 Прочие

12.6.30.1 storage_time

Параметр задает формат времени, используемый для записей в базе данных. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.6.30.2 max_error_count_per_task

Параметр задает максимальное количество ошибок для каждой сохраняемой задачи.

Формат задания настройки — `integer`.

Значение по умолчанию — `100000`.

12.6.30.3 tasks_to_faces_requests_concurrency

Параметр задает максимальное количество одновременных запросов (параллельных операций) из «рабочего(чих) процесса(ов)» Tasks в сервис Faces в один момент времени.

Это ограничение на количество запросов, которые могут быть обработаны в один и тот же момент времени.

Формат задания настройки — `integer`.

Значение по умолчанию — `5`.

12.6.30.4 tasks_to_image_store_requests_concurrency

Параметр задает максимальное количество одновременных запросов (параллельных операций) из «рабочего(чих) процесса(ов)» Tasks в сервис Image-Store в один момент времени.

Это ограничение на количество запросов, которые могут быть обработаны в один и тот же момент времени.

Формат задания настройки — `integer`.

Значение по умолчанию — `100`.

12.6.30.5 tasks_to_handlers_requests_concurrency

Параметр задает максимальное количество одновременных запросов (параллельных операций) из «рабочего(чих) процесса(ов)» Tasks в сервис Handlers в один момент времени.

Это ограничение на количество запросов, которые могут быть обработаны в один и тот же момент времени.

Формат задания настройки — `integer`.

Значение по умолчанию — `8`.

12.6.30.6 luna_tasks_active_plugins

Параметр задает список плагинов, которые должен использовать сервис.

Имена задаются в следующем формате:

```
[
   "plugin_1",
   "plugin_2",
   "plugin_3"
]
```

> Список должен содержать имена файлов без расширения (.py).

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.7 Настройки сервиса Events

Данный раздел описывает параметры сервиса Events.

Настройку сервиса можно выполнить с помощью сервиса Configurator.

12.7.1 Группа параметров LUNA_CONFIGURATOR

Данная группа параметров задаёт настройки подключения сервиса Events к сервису Configurator.

Данная группа параметров не будет видна в пользовательском интерфейсе сервиса Configurator. Параметры можно изменить только в конфигурационном файле «config.conf», расположенном в директории «/srv/luna_events/configs/» соответствующего контейнера.

12.7.1.1 use_configurator

Параметр позволяет включить использование сервиса Configurator.

> С помощью сервиса Configurator упрощается настройка сервисов LP. Сервис сохраняет все необходимые конфигурации для всех сервисов LP в одном месте.

При отключенном параметре будут использованы настройки из файла «config.conf», расположенного в директории «/srv/luna_events/configs/» соответствующего контейнера.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.7.1.2 luna_configurator_origin

Параметр задает протокол, IP адрес и порт сервиса Configurator.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5070`.

12.7.1.3 luna_configurator_api

Параметр задает версию API сервиса Configurator. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.7.2 Группа параметров LUNA_EVENTS_LOGGER

Данная группа параметров задает настройки логирования.

12.7.2.1 log_level

Параметр задает уровень отладочной печати, по приоритету: «ERROR», «WARNING», «INFO», «DEBUG».

Формат задания настройки — `string`.

Значение по умолчанию — `INFO`.

12.7.2.2 log_time

Параметр задает формат времени, используемый в записях лога. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.7.2.3 log_to_stdout

Параметр позволяет отправлять логи в стандартный вывод (stdout).

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`

12.7.2.4 log_to_file

Параметр позволяет сохранять логи в файл. Директория с файлами логов указывается в параметре «folder_with_logs».

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.7.2.5 folder_with_logs

Параметр задает директорию, в которой хранятся логи. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить параметр «log_to_file».

Формат задания настройки — `string`.

Значение по умолчанию — `./`

Пример:

```
  "folder_with_logs": "/srv/logs"
```

12.7.2.6 max_log_file_size

Параметр задает максимальный размер файла лога в МБ перед выполнением его ротации (0 — не использовать ротацию).

Для использования данного параметра требуется включить параметр «log_to_file».

> При необходимости можно настроить ротацию логов Docker. См. раздел «Настройка ротации логов Docker» в руководстве по установке LUNA PLATFORM.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024`

12.7.2.7 multiline_stack_trace

Параметр включает многострочную трассировку стека в логах. Когда параметр включен, информация о стеке вызовов записывается в логах так, что каждый фрейм стека помещается на отдельной строке, что улучшает читаемость. Если параметр выключен, информация о стеке вызовов записывается на одной строке, что может сделать логи менее удобными для анализа.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.7.2.8 format

Параметр определяет формат выводимых логов. Доступны следующие значения:

- «default» — стандартный формат вывода логов LUNA PLATFORM
- «json» — вывод логов в формате json
- «ecs» — вывод логов в формате ECS (Elastic Common Schema)

При использовании значения «ecs» будут использоваться следующие поля:

- «http.response.status_code» — содержит код состояния ответа HTTP (200, 404, 500 и т.д.);
- «http.response.execution_time» — содержит информацию о времени, затраченном на выполнение запроса и получение ответа;
- «http.request.method» — содержит метод HTTP-запроса (GET, POST, PUT и т.д.);
- «url.path» — содержит путь в URL-адресе запроса;
- «error.code» — содержит код ошибки, если запрос завершается с ошибкой.

Формат задания настройки — `string`.

Значение по умолчанию — `default`.

12.7.3 Группа параметров LUNA_EVENTS_DB

В данной группе параметров задаются настройки подключения к базе данных сервиса Events.

12.7.3.1 db_type

Параметр задает тип базы данных. Доступны следующие типы:

- «postgres» — тип СУБД PostgreSQL
- «oracle» — тип СУБД Oracle

Формат задания настройки — `string`.

Значение по умолчанию — `postgres`.

12.7.3.2 db_host

Параметр задает имя сервера (хост) базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.7.3.3 db_port

Параметр задает порт базы данных.

Порт по умолчанию для типа «postgres»- `5432`.

Порт по умолчанию для типа «oracle» — `1521`.

Формат задания настройки — `string`.

Значение по умолчанию — `5432`.

12.7.3.4 db_user

Параметр задает имя пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.7.3.5 db_password

Параметр задает пароль пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.7.3.6 db_name

Параметр задает имя базы данных для типа «postgres» или имя SID для типа «oracle».

Формат задания настройки — `string`.

Значение по умолчанию — `luna_events`.

12.7.3.7 connection_pool_size

Параметр задает размер пула соединений к БД. Реальное количество подключений может быть больше значения данной настройки на 1.

При необходимости в настройке «max_connections» конфигурационного файла PostgreSQL можно задать максимальное количество одновременных подключений к серверу БД. См. раздел «Продвинутая настройка PostrgeSQL» для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `10`.

12.7.3.8 dsn

Параметр задает строку подключения DSN для соединения с БД.

> DSN — это строка подключения, которая идентифицирует и указывает на источник данных (базу данных), к которому нужно установить соединение.

В строке DSN могут задаваться такие настройки, как множественные хосты, аутентификационные данные, порт и другие параметры.

> Настройки зависят от типа БД. Множественные хосты поддерживаются только с PostgreSQL.

> По умолчанию параметр «dsn» не отображается во вкладке «Settings» в Configurator. Проверить список всех доступных параметров для группы настроек можно на вкладке «Limitations».

Пример:

```
  {
  "dsn": "luna:luna@postgres01:5432,postgres02:5432/luna_events?some_option=
     some_value"
  "db_settings": {
      "connection_pool_size": 5
      }
  }
```

Здесь:

- «luna:luna» — имя пользователя и пароль для подключения к PostgreSQL;

- «@postgres01:5432,postgres02:5432» — список хостов и портов, разделенных запятыми. Это означает, что сервис будет пытаться подключиться к первому хосту («postgres01») на порту 5432. Если это не удастся, она попытается подключиться ко второму хосту («postgres02») также на порту 5432;
- «/luna_events» — название базы данных;
- «?some_option=some_value» — опциональные параметры для подключения.

При необходимости можно комбинировать строку DSN и классические настройки, однако строка DSN является более приоритетной. Можно частично заполнить строку DSN (например, «postgres01,postgres02/luna_events»), и тогда недостающие параметры будут заполнены из значений параметров «db_host», «db_port», «db_name», «db_user» и «db_password».

При запуске сервис создаст пул подключений к одному из доступных хостов DSN. В случае возникновения проблем с установлением соединения после нескольких неудачных попыток, сервис снова попытается настроить пул подключений к любому из доступных хостов DSN.

12.7.4 Группа параметров LUNA_MONITORING

Данная группа параметров задает настройки мониторинга.

См. подробную информацию о мониторинге в разделе «Мониторинг».

12.7.4.1 storage_type

Тип хранилища для хранения данных мониторинга.

В настоящее время доступно только БД Influx.

Формат задания настройки — `string`.

Значение по умолчанию — `influx`.

12.7.4.2 send_data_for_monitoring

Параметр позволяет включить или отключить отправку данных для мониторинга в InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.7.4.3 use_ssl

Параметр позволяет использовать HTTPS для подключения к InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.7.4.4 organization

Параметр задает рабочую область InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.7.4.5 token

Параметр задает аутентификации InfluxDB 2.x.

Формат задания настройки — `string`.

12.7.4.6 bucket

Параметр задаёт имя бакета InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_monitoring`.

12.7.4.7 host

Параметр задаёт IP-адрес InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.7.4.8 port

Параметр задает порт InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `8086`.

12.7.4.9 flushing_period

Параметр задает частоту отправки данных мониторинга в InfluxDB.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `1`.

12.7.5 Группа параметров LUNA_EVENTS_HTTP_SETTINGS

В данной группе параметров содержатся настройки, отвечающие за обработку HTTP-подключений. См. подробную информацию по следующей ссылке: https://sanic.dev/en/guide/deployment/configuration.html#builtin-values

12.7.5.1 request_timeout

Параметр задает продолжительность времени между моментом, когда новое открытое TCP-соединение передается на сервер, и моментом, когда получен весь HTTP-запрос.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.7.5.2 response_timeout

Параметр задает продолжительность времени между моментом, когда сервер передает HTTP-запрос приложению, и моментом, когда HTTP-ответ отправляется клиенту.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `600`.

12.7.5.3 request_max_size

Параметр задает максимальный размер запроса.

Формат задания настройки — `integer` (байты).

Значение по умолчанию — `1073741824`.

12.7.5.4 keep_alive_timeout

Параметр задает тайм-аут поддержания активности HTTP.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `15`.

12.7.6 Группа параметров LUNA_SERVICE_METRICS

Данная группа параметров включает и настраивает сбор метрик в формате Prometheus.

См. подробную информацию в разделе «Мониторинг».

12.7.6.1 enabled

Параметр включает сбор метрик.

Если сбор метрик отключен, то запрос к ресурсу `/metrics` вернет соответствующее сообщение.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.7.6.2 metrics_format

Параметр задает формат метрик.

На данный момент поддерживается только формат Prometheus.

См. официальную документацию Prometheus для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `prometheus`.

12.7.6.3 extra_labels

Параметр задает пользовательские типы лейблов.

Формат задания настройки — `label_name=label_value`.

Значение по умолчанию не задано.

12.7.7 Группа параметров DESCRIPTOR_ENCRYPTION

Данная группа параметров задает настройки шифрования биометрических шаблонов для повышения безопасности и предотвращения несанкционированного использования.

См. подробную информацию в разделе «Шифрование биометрических шаблонов».

12.7.7.1 enabled

Параметр включает шифрование биометрических шаблонов.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.7.7.2 algorithm

Параметр задает название используемого алгоритма шифрования.

Формат задания настройки — `string`.

Значение по умолчанию — `aes256-gcm`.

12.7.7.3 params > source

Параметр задает название источника ключа шифрования. Поддерживаются два типа источников: `raw` и `vaultKV`.

Формат задания настройки — `string`.

Значение по умолчанию — `raw`.

12.7.7.4 params > key

Параметр задает ключ шифрования или учетные данные для его получения из указанного источника.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.7.8 Прочие

12.7.8.1 storage_time

Параметр задает формат времени, используемый для записей в базе данных. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.7.8.2 default_face_descriptor_version

Параметр задает используемую версию биометрического шаблона лица.

Формат задания настройки — `string`.

Значение по умолчанию — `59`.

Примечание. См. подробную информацию о версиях биометрических шаблонах в разделе «Нейронные сети».

12.7.8.3 default_human_descriptor_version

Параметр задает используемую версию биометрического шаблона тела.

См. подробную информацию о версиях биометрических шаблонах в разделе «Нейронные сети».

Формат задания настройки — `string`.

Значение по умолчанию — `116`.

Примечание. См. подробную информацию о версиях биометрических шаблонах в разделе «Нейронные сети».

12.7.8.4 save_events_timeout

Параметр задает время ожидания для сохранения событий в базе данных Events.

Значения «0» или отрицательные означают, что сервис будет сохранять события без каких-либо ограничений по времени ожидания.

Использование данного параметра предотвращает избыточное использование вычислительных ресурсов.

Формат задания настройки — `float` (секунды).

Значение по умолчанию — `2`.

12.7.8.5 luna_events_active_plugins

Параметр задает список плагинов, которые должен использовать сервис.

Имена задаются в следующем формате:

```
[
   "plugin_1",
   "plugin_2",
   "plugin_3"
]
```

> Список должен содержать имена файлов без расширения (.py).

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.8 Настройки сервиса Sender

Данный раздел описывает параметры сервиса Sender.

Настройку сервиса можно выполнить с помощью сервиса Configurator.

12.8.1 Группа параметров LUNA_CONFIGURATOR

Данная группа параметров задаёт настройки подключения сервиса Sender к сервису Configurator.

Данная группа параметров не будет видна в пользовательском интерфейсе сервиса Configurator. Параметры можно изменить только в конфигурационном файле «config.conf», расположенном в директории «/srv/luna_sender/configs/» соответствующего контейнера.

12.8.1.1 use_configurator

Параметр позволяет включить использование сервиса Configurator.

> С помощью сервиса Configurator упрощается настройка сервисов LP. Сервис сохраняет все необходимые конфигурации для всех сервисов LP в одном месте.

При отключенном параметре будут использованы настройки из файла «config.conf», расположенного в директории «/srv/luna_sender/configs/» соответствующего контейнера.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.8.1.2 luna_configurator_origin

Параметр задает протокол, IP адрес и порт сервиса Configurator.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5070`.

12.8.1.3 luna_configurator_api

Параметр задает версию API сервиса Configurator. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.8.2 Группа параметров LUNA_SENDER_LOGGER

Данная группа параметров задает настройки логирования.

12.8.2.1 log_level

Параметр задает уровень отладочной печати, по приоритету: «ERROR», «WARNING», «INFO», «DEBUG».

Формат задания настройки — `string`.

Значение по умолчанию — `INFO`.

12.8.2.2 log_time

Параметр задает формат времени, используемый в записях лога. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.8.2.3 log_to_stdout

Параметр позволяет отправлять логи в стандартный вывод (stdout).

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`

12.8.2.4 log_to_file

Параметр позволяет сохранять логи в файл. Директория с файлами логов указывается в параметре «folder_with_logs».

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.8.2.5 folder_with_logs

Параметр задает директорию, в которой хранятся логи. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить параметр «log_to_file».

Формат задания настройки — `string`.

Значение по умолчанию — `./`

Пример:

```
  "folder_with_logs": "/srv/logs"
```

12.8.2.6 max_log_file_size

Параметр задает максимальный размер файла лога в МБ перед выполнением его ротации (0 — не использовать ротацию).

Для использования данного параметра требуется включить параметр «log_to_file».

> При необходимости можно настроить ротацию логов Docker. См. раздел «Настройка ротации логов Docker» в руководстве по установке LUNA PLATFORM.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024`

12.8.2.7 multiline_stack_trace

Параметр включает многострочную трассировку стека в логах. Когда параметр включен, информация о стеке вызовов записывается в логах так, что каждый фрейм стека помещается на отдельной строке, что улучшает читаемость. Если параметр выключен, информация о стеке вызовов записывается на одной строке, что может сделать логи менее удобными для анализа.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.8.2.8 format

Параметр определяет формат выводимых логов. Доступны следующие значения:

- «default» — стандартный формат вывода логов LUNA PLATFORM
- «json» — вывод логов в формате json
- «ecs» — вывод логов в формате ECS (Elastic Common Schema)

При использовании значения «ecs» будут использоваться следующие поля:

- «http.response.status_code» — содержит код состояния ответа HTTP (200, 404, 500 и т.д.);
- «http.response.execution_time» — содержит информацию о времени, затраченном на выполнение запроса и получение ответа;
- «http.request.method» — содержит метод HTTP-запроса (GET, POST, PUT и т.д.);
- «url.path» — содержит путь в URL-адресе запроса;
- «error.code» — содержит код ошибки, если запрос завершается с ошибкой.

Формат задания настройки — `string`.

Значение по умолчанию — `default`.

12.8.3 Группа параметров REDIS_DB_ADDRESS

12.8.3.1 user

Параметр задает имя пользователя БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.8.3.2 password

Параметр задает пароль БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию не указывается.

12.8.3.3 host

Параметр задает IP-адрес БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.8.3.4 port

Параметр задает номер порта, на котором Redis ожидает входящие сетевые соединения и прослушивает их для выполнения команд от клиентов.

Формат задания настройки — `integer`.

Значение по умолчанию — `6379`.

12.8.3.5 channel

Параметр задает канал Redis-канал, на который сервис подписывается сервис.

Формат задания настройки — `integer`.

Значение по умолчанию — `luna-sender`.

12.8.3.6 sentinel > master_name

Параметр задает имя мастера базы данных Redis, который контролируется и управляется системой Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_sender_master`.

12.8.4 Группа параметров LUNA_SERVICE_METRICS

Данная группа параметров включает и настраивает сбор метрик в формате Prometheus.

См. подробную информацию в разделе «Мониторинг».

12.8.4.1 enabled

Параметр включает сбор метрик.

Если сбор метрик отключен, то запрос к ресурсу `/metrics` вернет соответствующее сообщение.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.8.4.2 metrics_format

Параметр задает формат метрик.

На данный момент поддерживается только формат Prometheus.

См. официальную документацию Prometheus для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `prometheus`.

12.8.4.3 extra_labels

Параметр задает пользовательские типы лейблов.

Формат задания настройки — `label_name=label_value`.

Значение по умолчанию не задано.

12.8.4.4 sentinel > sentinels

Параметр задает список адресов и портов серверов Sentinel, которые будут использоваться клиентами для обнаружения и мониторинга БД Redis.

Формат задания настройки — `list > string`.

Значение по умолчанию — `[]`.

12.8.4.5 sentinel > user

Параметр задает имя пользователя сервера Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.8.4.6 sentinel > password

Параметр задает пароль пользователя сервера Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.8.5 Группа параметров LUNA_MONITORING

Данная группа параметров задает настройки мониторинга.

См. подробную информацию о мониторинге в разделе «Мониторинг».

12.8.5.1 storage_type

Тип хранилища для хранения данных мониторинга.

В настоящее время доступно только БД Influx.

Формат задания настройки — `string`.

Значение по умолчанию — `influx`.

12.8.5.2 send_data_for_monitoring

Параметр позволяет включить или отключить отправку данных для мониторинга в InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.8.5.3 use_ssl

Параметр позволяет использовать HTTPS для подключения к InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.8.5.4 organization

Параметр задает рабочую область InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.8.5.5 token

Параметр задает аутентификации InfluxDB 2.x.

Формат задания настройки — `string`.

12.8.5.6 bucket

Параметр задаёт имя бакета InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_monitoring`.

12.8.5.7 host

Параметр задаёт IP-адрес InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.8.5.8 port

Параметр задает порт InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `8086`.

12.8.5.9 flushing_period

Параметр задает частоту отправки данных мониторинга в InfluxDB.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `1`.

12.8.6 Группа параметров LUNA_SENDER_HTTP_SETTINGS

В данной группе параметров содержатся настройки, отвечающие за обработку HTTP-подключений. См. подробную информацию по следующей ссылке: https://sanic.dev/en/guide/deployment/configuration.html#builtin-values

12.8.6.1 request_timeout

Параметр задает продолжительность времени между моментом, когда новое открытое TCP-соединение передается на сервер, и моментом, когда получен весь HTTP-запрос.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.8.6.2 response_timeout

Параметр задает продолжительность времени между моментом, когда сервер передает HTTP-запрос приложению, и моментом, когда HTTP-ответ отправляется клиенту.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `600`.

12.8.6.3 request_max_size

Параметр задает максимальный размер запроса.

Формат задания настройки — `integer` (байты).

Значение по умолчанию — `1073741824`.

12.8.6.4 keep_alive_timeout

Параметр задает тайм-аут поддержания активности HTTP.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `15`.

12.8.7 Прочие

12.8.7.1 luna_sender_active_plugins

Параметр задает список плагинов, которые должен использовать сервис.

Имена задаются в следующем формате:

```
[
   "plugin_1",
   "plugin_2",
   "plugin_3"
]
```

> Список должен содержать имена файлов без расширения (.py).

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.9 Настройки сервиса Licenses

Данный раздел описывает параметры сервиса Licenses.

Настройку сервиса можно выполнить с помощью сервиса Configurator.

12.9.1 Группа параметров LUNA_CONFIGURATOR

Данная группа параметров задаёт настройки подключения сервиса Licenses к сервису Configurator.

Данная группа параметров не будет видна в пользовательском интерфейсе сервиса Configurator. Параметры можно изменить только в конфигурационном файле «config.conf», расположенном в директории «/srv/luna_licenses/configs/» соответствующего контейнера.

12.9.1.1 use_configurator

Параметр позволяет включить использование сервиса Configurator.

> С помощью сервиса Configurator упрощается настройка сервисов LP. Сервис сохраняет все необходимые конфигурации для всех сервисов LP в одном месте.

При отключенном параметре будут использованы настройки из файла «config.conf», расположенного в директории «/srv/luna_licenses/configs/» соответствующего контейнера.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.9.1.2 luna_configurator_origin

Параметр задает протокол, IP адрес и порт сервиса Configurator.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5070`.

12.9.1.3 luna_configurator_api

Параметр задает версию API сервиса Configurator. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.9.2 Группа параметров LUNA_LICENSES_LOGGER

Данная группа параметров задает настройки логирования.

12.9.2.1 log_level

Параметр задает уровень отладочной печати, по приоритету: «ERROR», «WARNING», «INFO», «DEBUG».

Формат задания настройки — `string`.

Значение по умолчанию — `INFO`.

12.9.2.2 log_time

Параметр задает формат времени, используемый в записях лога. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.9.2.3 log_to_stdout

Параметр позволяет отправлять логи в стандартный вывод (stdout).

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`

12.9.2.4 log_to_file

Параметр позволяет сохранять логи в файл. Директория с файлами логов указывается в параметре «folder_with_logs».

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.9.2.5 folder_with_logs

Параметр задает директорию, в которой хранятся логи. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить параметр «log_to_file».

Формат задания настройки — `string`.

Значение по умолчанию — `./`

Пример:

```
  "folder_with_logs": "/srv/logs"
```

12.9.2.6 max_log_file_size

Параметр задает максимальный размер файла лога в МБ перед выполнением его ротации (0 — не использовать ротацию).

Для использования данного параметра требуется включить параметр «log_to_file».

> При необходимости можно настроить ротацию логов Docker. См. раздел «Настройка ротации логов Docker» в руководстве по установке LUNA PLATFORM.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024`

12.9.2.7 multiline_stack_trace

Параметр включает многострочную трассировку стека в логах. Когда параметр включен, информация о стеке вызовов записывается в логах так, что каждый фрейм стека помещается на отдельной строке, что улучшает читаемость. Если параметр выключен, информация о стеке вызовов записывается на одной строке, что может сделать логи менее удобными для анализа.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.9.2.8 format

Параметр определяет формат выводимых логов. Доступны следующие значения:

- «default» — стандартный формат вывода логов LUNA PLATFORM
- «json» — вывод логов в формате json
- «ecs» — вывод логов в формате ECS (Elastic Common Schema)

При использовании значения «ecs» будут использоваться следующие поля:

- «http.response.status_code» — содержит код состояния ответа HTTP (200, 404, 500 и т.д.);
- «http.response.execution_time» — содержит информацию о времени, затраченном на выполнение запроса и получение ответа;
- «http.request.method» — содержит метод HTTP-запроса (GET, POST, PUT и т.д.);
- «url.path» — содержит путь в URL-адресе запроса;
- «error.code» — содержит код ошибки, если запрос завершается с ошибкой.

Формат задания настройки — `string`.

Значение по умолчанию — `default`.

12.9.3 Группа параметров LUNA_MONITORING

Данная группа параметров задает настройки мониторинга.

См. подробную информацию о мониторинге в разделе «Мониторинг».

12.9.3.1 storage_type

Тип хранилища для хранения данных мониторинга.

В настоящее время доступно только БД Influx.

Формат задания настройки — `string`.

Значение по умолчанию — `influx`.

12.9.3.2 send_data_for_monitoring

Параметр позволяет включить или отключить отправку данных для мониторинга в InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.9.3.3 use_ssl

Параметр позволяет использовать HTTPS для подключения к InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.9.3.4 organization

Параметр задает рабочую область InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.9.3.5 token

Параметр задает аутентификации InfluxDB 2.x.

Формат задания настройки — `string`.

12.9.3.6 bucket

Параметр задаёт имя бакета InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_monitoring`.

12.9.3.7 host

Параметр задаёт IP-адрес InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.9.3.8 port

Параметр задает порт InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `8086`.

12.9.3.9 flushing_period

Параметр задает частоту отправки данных мониторинга в InfluxDB.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `1`.

12.9.4 Группа параметров LICENSE_VENDOR

Данная группа параметров задает настройки лицензирования.

См. подробную информацию по активации лицензии в руководстве по установке и в руководстве по активации LUNA PLATFORM.

См. подробную информацию о лицензии в разделе «Информация о лицензии».

12.9.4.1 vendor

Параметр определяет, какой вендор лицензий будет использоваться. Доступно два варианта:

- «hasp»
- «guardant»

Формат задания настройки — `string`.

Значение по умолчанию — `hasp`.

12.9.4.2 server_address

Параметр задает IP-адрес, по которому сервис Licenses будет искать сервер для управления лицензиями.

> Адрес должен указываться без протокола и порта.

Значение «127.0.0.1» означает, что сервер находится на локальном компьютере. Если вы используете внешний сервер для лицензирования, вам следует изменить это значение на адрес вашего сервера.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.9.4.3 license_id

Примечание. Параметр используется только для вендора «guardant» дополнительно с параметрами «server_address» и «vendor». Если используется вендор «hasp», то необходимо отключить данный параметр.

Параметр задает идентификатор лицензии в формате `0x<your_license_id>`. Идентификатор лицензии можно найти в пользовательском интерфейсе Guardant по адресу `http://<your_host_address>:3189/` на вкладке «Ключи».

Формат задания настройки — `string`.

Значение по умолчанию не указано.

12.9.5 Группа параметров LUNA_LICENSES_HTTP_SETTINGS

В данной группе параметров содержатся настройки, отвечающие за обработку HTTP-подключений. См. подробную информацию по следующей ссылке: https://sanic.dev/en/guide/deployment/configuration.html#builtin-values

12.9.5.1 request_timeout

Параметр задает продолжительность времени между моментом, когда новое открытое TCP-соединение передается на сервер, и моментом, когда получен весь HTTP-запрос.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.9.5.2 response_timeout

Параметр задает продолжительность времени между моментом, когда сервер передает HTTP-запрос приложению, и моментом, когда HTTP-ответ отправляется клиенту.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `600`.

12.9.5.3 request_max_size

Параметр задает максимальный размер запроса.

Формат задания настройки — `integer` (байты).

Значение по умолчанию — `1073741824`.

12.9.5.4 keep_alive_timeout

Параметр задает тайм-аут поддержания активности HTTP.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `15`.

12.9.6 Группа параметров LUNA_SERVICE_METRICS

Данная группа параметров включает и настраивает сбор метрик в формате Prometheus.

См. подробную информацию в разделе «Мониторинг».

12.9.6.1 enabled

Параметр включает сбор метрик.

Если сбор метрик отключен, то запрос к ресурсу `/metrics` вернет соответствующее сообщение.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.9.6.2 metrics_format

Параметр задает формат метрик.

На данный момент поддерживается только формат Prometheus.

См. официальную документацию Prometheus для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `prometheus`.

12.9.6.3 extra_labels

Параметр задает пользовательские типы лейблов.

Формат задания настройки — `label_name=label_value`.

Значение по умолчанию не задано.

12.9.7 Прочие

12.9.7.1 luna_licenses_active_plugins

Параметр задает список плагинов, которые должен использовать сервис.

Имена задаются в следующем формате:

```
[
   "plugin_1",
   "plugin_2",
   "plugin_3"
]
```

> Список должен содержать имена файлов без расширения (.py).

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.10 Настройки сервиса Python Matcher

Данный раздел описывает параметры сервиса Python Matcher.

Настройку сервиса можно выполнить с помощью сервиса Configurator.

12.10.1 Группа параметров LUNA_CONFIGURATOR

Данная группа параметров задаёт настройки подключения сервиса Python Matcher к сервису Configurator.

Данная группа параметров не будет видна в пользовательском интерфейсе сервиса Configurator. Параметры можно изменить только в конфигурационном файле «config.conf», расположенном в директории «/srv/luna_python_matcher/configs/» соответствующего контейнера.

12.10.1.1 use_configurator

Параметр позволяет включить использование сервиса Configurator.

> С помощью сервиса Configurator упрощается настройка сервисов LP. Сервис сохраняет все необходимые конфигурации для всех сервисов LP в одном месте.

При отключенном параметре будут использованы настройки из файла «config.conf», расположенного в директории «/srv/luna_python_matcher/configs/» соответствующего контейнера.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.10.1.2 luna_configurator_origin

Параметр задает протокол, IP адрес и порт сервиса Configurator.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5070`.

12.10.1.3 luna_configurator_api

Параметр задает версию API сервиса Configurator. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.10.2 Группа параметров LUNA_PYTHON_MATCHER_LOGGER

Данная группа параметров задает настройки логирования.

12.10.2.1 log_level

Параметр задает уровень отладочной печати, по приоритету: «ERROR», «WARNING», «INFO», «DEBUG».

Формат задания настройки — `string`.

Значение по умолчанию — `INFO`.

12.10.2.2 log_time

Параметр задает формат времени, используемый в записях лога. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.10.2.3 log_to_stdout

Параметр позволяет отправлять логи в стандартный вывод (stdout).

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`

12.10.2.4 log_to_file

Параметр позволяет сохранять логи в файл. Директория с файлами логов указывается в параметре «folder_with_logs».

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.10.2.5 folder_with_logs

Параметр задает директорию, в которой хранятся логи. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить параметр «log_to_file».

Формат задания настройки — `string`.

Значение по умолчанию — `./`

Пример:

```
  "folder_with_logs": "/srv/logs"
```

12.10.2.6 max_log_file_size

Параметр задает максимальный размер файла лога в МБ перед выполнением его ротации (0 — не использовать ротацию).

Для использования данного параметра требуется включить параметр «log_to_file».

> При необходимости можно настроить ротацию логов Docker. См. раздел «Настройка ротации логов Docker» в руководстве по установке LUNA PLATFORM.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024`

12.10.2.7 multiline_stack_trace

Параметр включает многострочную трассировку стека в логах. Когда параметр включен, информация о стеке вызовов записывается в логах так, что каждый фрейм стека помещается на отдельной строке, что улучшает читаемость. Если параметр выключен, информация о стеке вызовов записывается на одной строке, что может сделать логи менее удобными для анализа.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.10.2.8 format

Параметр определяет формат выводимых логов. Доступны следующие значения:

- «default» — стандартный формат вывода логов LUNA PLATFORM
- «json» — вывод логов в формате json
- «ecs» — вывод логов в формате ECS (Elastic Common Schema)

При использовании значения «ecs» будут использоваться следующие поля:

- «http.response.status_code» — содержит код состояния ответа HTTP (200, 404, 500 и т.д.);
- «http.response.execution_time» — содержит информацию о времени, затраченном на выполнение запроса и получение ответа;
- «http.request.method» — содержит метод HTTP-запроса (GET, POST, PUT и т.д.);
- «url.path» — содержит путь в URL-адресе запроса;
- «error.code» — содержит код ошибки, если запрос завершается с ошибкой.

Формат задания настройки — `string`.

Значение по умолчанию — `default`.

12.10.3 Группа параметров LUNA_MONITORING

Данная группа параметров задает настройки мониторинга.

См. подробную информацию о мониторинге в разделе «Мониторинг».

12.10.3.1 storage_type

Тип хранилища для хранения данных мониторинга.

В настоящее время доступно только БД Influx.

Формат задания настройки — `string`.

Значение по умолчанию — `influx`.

12.10.3.2 send_data_for_monitoring

Параметр позволяет включить или отключить отправку данных для мониторинга в InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.10.3.3 use_ssl

Параметр позволяет использовать HTTPS для подключения к InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.10.3.4 organization

Параметр задает рабочую область InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.10.3.5 token

Параметр задает аутентификации InfluxDB 2.x.

Формат задания настройки — `string`.

12.10.3.6 bucket

Параметр задаёт имя бакета InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_monitoring`.

12.10.3.7 host

Параметр задаёт IP-адрес InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.10.3.8 port

Параметр задает порт InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `8086`.

12.10.3.9 flushing_period

Параметр задает частоту отправки данных мониторинга в InfluxDB.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `1`.

12.10.4 Группа параметров LUNA_FACES_DB

В данной группе параметров задаются настройки подключения к базе данных сервиса Faces.

12.10.4.1 db_type

Параметр задает тип базы данных. Доступны следующие типы:

- «postgres» — тип СУБД PostgreSQL
- «oracle» — тип СУБД Oracle

Формат задания настройки — `string`.

Значение по умолчанию — `postgres`.

12.10.4.2 db_host

Параметр задает имя сервера (хост) базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.10.4.3 db_port

Параметр задает порт базы данных.

Порт по умолчанию для типа «postgres»- `5432`.

Порт по умолчанию для типа «oracle» — `1521`.

Формат задания настройки — `string`.

Значение по умолчанию — `5432`.

12.10.4.4 db_user

Параметр задает имя пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.10.4.5 db_password

Параметр задает пароль пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.10.4.6 db_name

Параметр задает имя базы данных для типа «postgres» или имя SID для типа «oracle».

Формат задания настройки — `string`.

Значение по умолчанию — `luna_faces`.

12.10.4.7 connection_pool_size

Параметр задает размер пула соединений к БД. Реальное количество подключений может быть больше значения данной настройки на 1.

При необходимости в настройке «max_connections» конфигурационного файла PostgreSQL можно задать максимальное количество одновременных подключений к серверу БД. См. раздел «Продвинутая настройка PostrgeSQL» для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `10`.

12.10.4.8 dsn

Параметр задает строку подключения DSN для соединения с БД.

> DSN — это строка подключения, которая идентифицирует и указывает на источник данных (базу данных), к которому нужно установить соединение.

В строке DSN могут задаваться такие настройки, как множественные хосты, аутентификационные данные, порт и другие параметры.

> Настройки зависят от типа БД. Множественные хосты поддерживаются только с PostgreSQL.

> По умолчанию параметр «dsn» не отображается во вкладке «Settings» в Configurator. Проверить список всех доступных параметров для группы настроек можно на вкладке «Limitations».

Пример:

```
{
"dsn": "luna:luna@postgres01:5432,postgres02:5432/luna_faces?some_option=
    some_value"
"db_settings": {
    "connection_pool_size": 5
    }
}
```

Здесь:

- «luna:luna» — имя пользователя и пароль для подключения к PostgreSQL;
- «@postgres01:5432,postgres02:5432» — список хостов и портов, разделенных запятыми. Это означает, что сервис будет пытаться подключиться к первому хосту («postgres01») на порту 5432. Если это не удастся, она попытается подключиться ко второму хосту («postgres02») также на порту 5432;
- «/luna_faces» — название базы данных;
- «?some_option=some_value» — опциональные параметры для подключения.

При необходимости можно комбинировать строку DSN и классические настройки, однако строка DSN является более приоритетной. Можно частично заполнить строку DSN (например, «postgres01,postgres02/luna_faces»), и тогда недостающие параметры будут заполнены из значений параметров «db_host», «db_port», «db_name», «db_user» и «db_password».

При запуске сервис создаст пул подключений к одному из доступных хостов DSN. В случае возникновения проблем с установлением соединения после нескольких неудачных попыток, сервис снова попытается настроить пул подключений к любому из доступных хостов DSN.

12.10.5 Группа параметров LUNA_ATTRIBUTES_DB

12.10.5.1 user

Параметр задает имя пользователя БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.10.5.2 password

Параметр задает пароль БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию не указывается.

12.10.5.3 host

Параметр задает IP-адрес БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.10.5.4 port

Параметр задает номер порта, на котором Redis ожидает входящие сетевые соединения и прослушивает их для выполнения команд от клиентов.

Формат задания настройки — `integer`.

Значение по умолчанию — `6379`.

12.10.5.5 sentinel > master_name

Параметр задает имя мастера базы данных Redis, который контролируется и управляется системой Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_attributes`.

12.10.5.6 sentinel > sentinels

Параметр задает список адресов и портов серверов Sentinel, которые будут использоваться клиентами для обнаружения и мониторинга БД Redis.

Формат задания настройки — `list > string`.

Значение по умолчанию — `[]`.

12.10.5.7 sentinel > user

Параметр задает имя пользователя сервера Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.10.5.8 sentinel > password

Параметр задает пароль пользователя сервера Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.10.6 Группа параметров ADDITIONAL_SERVICES_USAGE

12.10.6.1 luna_events

Параметр задает возможность использования сервиса Events.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.10.6.2 luna_faces

Параметр задает возможность использования сервиса Faces.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.10.6.3 luna_handlers

Параметр задает возможность использования сервиса Handlers.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.10.6.4 luna_sender

Параметр задает возможность использования сервиса Sender.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.10.6.5 luna_matcher_proxy

Параметр задает возможность использования сервиса Python Matcher Proxy.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.10.6.6 luna_image_store

Параметр задает возможность использования сервиса Image Store.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.10.6.7 luna_lambda

Параметр задает возможность использования сервиса Lambda.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.10.6.8 luna_video_manager

Параметр задает возможность использования сервиса Video Manager.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.10.6.9 luna_video_agent

Параметр задает возможность использования сервиса Video Agent.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.10.6.10 luna_streams_retranslator

Параметр задает возможность использования сервиса Streams Retranslator.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.10.7 Группа параметров LUNA_EVENTS_DB

В данной группе параметров задаются настройки подключения к базе данных сервиса Events.

12.10.7.1 db_type

Параметр задает тип базы данных. Доступны следующие типы:

- «postgres» — тип СУБД PostgreSQL
- «oracle» — тип СУБД Oracle

Формат задания настройки — `string`.

Значение по умолчанию — `postgres`.

12.10.7.2 db_host

Параметр задает имя сервера (хост) базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.10.7.3 db_port

Параметр задает порт базы данных.

Порт по умолчанию для типа «postgres»- `5432`.

Порт по умолчанию для типа «oracle» — `1521`.

Формат задания настройки — `string`.

Значение по умолчанию — `5432`.

12.10.7.4 db_user

Параметр задает имя пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.10.7.5 db_password

Параметр задает пароль пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.10.7.6 db_name

Параметр задает имя базы данных для типа «postgres» или имя SID для типа «oracle».

Формат задания настройки — `string`.

Значение по умолчанию — `luna_events`.

12.10.7.7 connection_pool_size

Параметр задает размер пула соединений к БД. Реальное количество подключений может быть больше значения данной настройки на 1.

При необходимости в настройке «max_connections» конфигурационного файла PostgreSQL можно задать максимальное количество одновременных подключений к серверу БД. См. раздел «Продвинутая настройка PostrgeSQL» для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `10`.

12.10.7.8 dsn

Параметр задает строку подключения DSN для соединения с БД.

> DSN — это строка подключения, которая идентифицирует и указывает на источник данных (базу данных), к которому нужно установить соединение.

В строке DSN могут задаваться такие настройки, как множественные хосты, аутентификационные данные, порт и другие параметры.

> Настройки зависят от типа БД. Множественные хосты поддерживаются только с PostgreSQL.

> По умолчанию параметр «dsn» не отображается во вкладке «Settings» в Configurator. Проверить список всех доступных параметров для группы настроек можно на вкладке «Limitations».

Пример:

```
  {
  "dsn": "luna:luna@postgres01:5432,postgres02:5432/luna_events?some_option=
     some_value"
```

```
  "db_settings": {
      "connection_pool_size": 5
      }
  }
```

Здесь:

- «luna:luna» — имя пользователя и пароль для подключения к PostgreSQL;
- «@postgres01:5432,postgres02:5432» — список хостов и портов, разделенных запятыми. Это означает, что сервис будет пытаться подключиться к первому хосту («postgres01») на порту 5432. Если это не удастся, она попытается подключиться ко второму хосту («postgres02») также на порту 5432;
- «/luna_events» — название базы данных;
- «?some_option=some_value» — опциональные параметры для подключения.

При необходимости можно комбинировать строку DSN и классические настройки, однако строка DSN является более приоритетной. Можно частично заполнить строку DSN (например, «postgres01,postgres02/luna_events»), и тогда недостающие параметры будут заполнены из значений параметров «db_host», «db_port», «db_name», «db_user» и «db_password».

При запуске сервис создаст пул подключений к одному из доступных хостов DSN. В случае возникновения проблем с установлением соединения после нескольких неудачных попыток, сервис снова попытается настроить пул подключений к любому из доступных хостов DSN.

12.10.8 Группа параметров LUNA_PYTHON_MATCHER_ADDRESS

Данная группа параметров задает настройки подключения к сервису Python Matcher.

12.10.8.1 origin

Параметр задает протокол, IP адрес и порт сервиса Python Matcher.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Python Matcher, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Python Matcher.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5100`.

12.10.8.2 api_version

Параметр задает версию API сервиса Python Matcher. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.10.9 Группа параметров LUNA_PYTHON_MATCHER_HTTP_SETTINGS

В данной группе параметров содержатся настройки, отвечающие за обработку HTTP-подключений. См. подробную информацию по следующей ссылке: https://sanic.dev/en/guide/deployment/configuration.html#builtin-values

12.10.9.1 request_timeout

Параметр задает продолжительность времени между моментом, когда новое открытое TCP-соединение передается на сервер, и моментом, когда получен весь HTTP-запрос.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.10.9.2 response_timeout

Параметр задает продолжительность времени между моментом, когда сервер передает HTTP-запрос приложению, и моментом, когда HTTP-ответ отправляется клиенту.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `600`.

12.10.9.3 request_max_size

Параметр задает максимальный размер запроса.

Формат задания настройки — `integer` (байты).

Значение по умолчанию — `1073741824`.

12.10.9.4 keep_alive_timeout

Параметр задает тайм-аут поддержания активности HTTP.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `15`.

12.10.10 Группа параметров PLATFORM_LIMITS

Данная группа параметров определяет ограничения, связанные с операциями сравнения лиц, событий или атрибутов. Эти параметры управляют количеством объектов, которые могут быть включены в операцию сравнения, а также ограничивают количество результатов, возвращаемых в ответе.

Эти параметры полезны для балансировки производительности и использования ресурсов.

Параметры из подгруппы «match» определяют ограничения, связанные с операциями сравнения.

Параметры из подгруппы «cross_match» определяют ограничения, связанные с операциями перекрестного сравнения (см. описание задачи «Cross-matching»).

12.10.10.1 match > array_filter_limit

Параметр задает максимальное количество объектов (максимальный размер массива), указанных в фильтрах «face_ids», «event_ids» и «attribute_ids».

Формат задания настройки — `integer`.

Значение по умолчанию — `1000`.

12.10.10.2 match > reference_limit

Параметр задает максимальное количество эталонов (максимальный размер массива), которых можно указать в запросе на сравнение.

> Необходимо подбирать значение данного параметра в соответствии со своими бизнес-кейсами, т.к. он значительно потребляет ресурсы системы. Так, например, не следует использовать значение по умолчанию «30», если необходимо работать только с тремя эталонами. См. дополнительные советы по оптимизации ресурсов в разделе «Оптимизация ресурсов».

Формат задания настройки — `integer`.

Значение по умолчанию — `30`.

12.10.10.3 match > candidate_limit

Параметр задает максимальное количество кандидатов (максимальный размер массива), которых можно указать в запросе на сравнение.

> Необходимо подбирать значение данного параметра в соответствии со своими бизнес-кейсами, т.к. он значительно потребляет ресурсы системы. Так, например, не следует использовать значение по умолчанию «30», если необходимо работать только с тремя кандидатами. См. дополнительные советы по оптимизации ресурсов в разделе «Оптимизация ресурсов».

Формат задания настройки — `integer`.

Значение по умолчанию — `30`.

12.10.10.4 match > result_candidate_limit

Параметр задает максимальное количество результатов сравнения (максимальный размер массива) для списка кандидатов, возвращаемых в ответе запроса на сравнение.

В теле запроса на сравнение также можно ограничить количество возвращаемых кандидатов в ответе с помощью параметра «limit», однако параметр «result_candidate_limit» имеет более высокий приоритет. Это означает, что при значении «result_candidate_limit» = 4 и «limit» = 6, будет возвращена ошибка с кодом «31007».

Формат задания настройки — `integer`.

Значение по умолчанию — `100`.

12.10.10.5 cross_match > short_array_filter_limit

Параметр задает максимальное количество объектов (максимальный размер массива), указанных во всех фильтрах, кроме фильтров «face_ids», «event_ids» и «attribute_ids» в задаче Cross-matching.

> Например, максимальный размер массива для фильтров «sources», «tags», «cities» и др. В спецификации OpenAPI такие фильтры помечены примером `[ 1 .. 1000 ] items`.

Формат задания настройки — `integer`.

Значение по умолчанию — `1000`.

12.10.10.6 cross_match > array_filter_limit

Параметр задает максимальное количество объектов (максимальный размер массива), указанных в фильтрах «face_ids», «event_ids» и «attribute_ids» в задаче Cross-matching.

> В спецификации OpenAPI такие фильтры помечены примером `[ 1 .. 20000 ] items`.

Формат задания настройки — `integer`.

Значение по умолчанию — `20000`.

12.10.10.7 cross_match > result_candidate_limit

Параметр задает максимальное количество результатов сравнения (максимальный размер массива) для списка кандидатов, возвращаемых в ответе запроса на получение результата задачи Cross-matching.

В теле запроса на сравнение также можно ограничить количество возвращаемых кандидатов в ответе с помощью параметра «limit», однако параметр «result_candidate_limit» имеет более высокий приоритет. Это означает, что при значении «result_candidate_limit» = 4 и «limit» = 6, будет возвращена ошибка с кодом «31007».

Формат задания настройки — `integer`.

Значение по умолчанию — `20000`.

12.10.10.8 cross_match > general_limit

Параметр задает общее ограничение на количество перекрестных сравнений. Он применяется к произведению количества кандидатов и эталонов, которые участвуют в операции сравнения с учетом фильтров. Если произведение превышает значение «general_limit», запрос завершится ошибкой.

Например, если значение «general_limit» равно «20», количество кандидатов-лиц в массиве «face_ids» равно «7», количество эталонов-лиц в массиве «face_ids» равно «5» и все остальные фильтры не сокращают реальное количество кандидатов-лиц, то запрос на перекрестное сравнение не будет выполнен, т.к. общее количество перекрестных сравнений будет равно «35» (5 x 7). Если же какой-то фильтр сокращает общее количество кандидатов-лиц на «3», то запрос будет выполнен успешно, т.к. общее количество перекрестных сравнений будет равно «20» (5 x (7 - 3)).

Формат задания настройки — `integer`.

Значение по умолчанию — `100000000`.

12.10.11 Группа параметров DESCRIPTORS_CACHE

Данная группа параметров задает настройки кеширования списков при выполнении сравнения по спискам.

Использование кеширования увеличивает производительность сравнения.

См. подробную информацию в разделе «Кеширование списков».

12.10.11.1 cache_enabled

Параметр включает кеширование списков.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.10.11.2 updating_cache_interval

Параметр задает интервал обновления кеша. Все новые биометрические шаблоны будут добавлены за одну итерацию через указанное время.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `2`.

12.10.11.3 matching_settings > thread_count

Параметр задает количество потоков (threads), которые будут использоваться для сравнения данных по кешу.

Если установлено значение «0», то будет выбран автоматический метод выбора количества потоков.

Формат задания настройки — `integer` (неотрицательное значение).

Значение по умолчанию — `0`.

12.10.11.4 matching_settings > tasks_count

Параметр задает количество «рабочих процессов», обрабатывающих очередь на отправку запросов на сравнение в Cached Matcher.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.10.11.5 matching_settings > batch_size

Параметр задает максимальное количество запросов на сравнение в Cached Matcher в пределах одного пакета.

Формат задания настройки — `integer`.

Значение по умолчанию — `20`.

12.10.11.6 rpc_settings > timeouts > connect_timeout

Параметр задает максимальное время ожидания для установления соединения с Cached Matcher. Если соединение не устанавливается в пределах этого времени, то оно будет считаться неудачным.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.10.11.7 rpc_settings > timeouts > request_timeout

Параметр задает максимальное время, в течение которого запрос к Cached Matcher должен быть выполнен. Если запрос не завершится в пределах этого времени, он будет отменен.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.10.11.8 rpc_settings > timeouts > response_timeout

Параметр задает максимальное время ожидания ответа от Cached Matcher. Если ответ не поступит в пределах этого времени, операция будет считаться неудачной.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.10.11.9 rpc_settings > pool_size

Параметр задает количество подключений между сервисами Python Matcher и Cached Matcher.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `100`.

12.10.11.10 cached_data > faces_lists > exclude

Параметр задает списки, которые не будут кешированы.

> Для отключения настройки нужно задать значение «[]».

Формат задания настройки — `string`.

Значение по умолчанию — `[]`.

12.10.11.11 cached_data > faces_lists > include

Параметр задает списки, которые будут кешированы.

Формат задания настройки — `string`.

Значение по умолчанию не задано, что означает, что будут обработаны все существующие списки.

12.10.12 Группа параметров LUNA_SERVICE_METRICS

Данная группа параметров включает и настраивает сбор метрик в формате Prometheus.

См. подробную информацию в разделе «Мониторинг».

12.10.12.1 enabled

Параметр включает сбор метрик.

Если сбор метрик отключен, то запрос к ресурсу `/metrics` вернет соответствующее сообщение.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.10.12.2 metrics_format

Параметр задает формат метрик.

На данный момент поддерживается только формат Prometheus.

См. официальную документацию Prometheus для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `prometheus`.

12.10.12.3 extra_labels

Параметр задает пользовательские типы лейблов.

Формат задания настройки — `label_name=label_value`.

Значение по умолчанию не задано.

12.10.13 Группа параметров DESCRIPTOR_ENCRYPTION

Данная группа параметров задает настройки шифрования биометрических шаблонов для повышения безопасности и предотвращения несанкционированного использования.

См. подробную информацию в разделе «Шифрование биометрических шаблонов».

12.10.13.1 enabled

Параметр включает шифрование биометрических шаблонов.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.10.13.2 algorithm

Параметр задает название используемого алгоритма шифрования.

Формат задания настройки — `string`.

Значение по умолчанию — `aes256-gcm`.

12.10.13.3 params > source

Параметр задает название источника ключа шифрования. Поддерживаются два типа источников: `raw` и `vaultKV`.

Формат задания настройки — `string`.

Значение по умолчанию — `raw`.

12.10.13.4 params > key

Параметр задает ключ шифрования или учетные данные для его получения из указанного источника.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.10.14 Прочие

12.10.14.1 storage_time

Параметр задает формат времени, используемый для записей в базе данных. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.10.14.2 luna_python_matcher_active_plugins

Параметр задает список плагинов, которые должен использовать сервис.

Имена задаются в следующем формате:

```
[
   "plugin_1",
   "plugin_2",
   "plugin_3"
]
```

> Список должен содержать имена файлов без расширения (.py).

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.10.14.3 default_face_descriptor_version

Параметр задает используемую версию биометрического шаблона лица.

Формат задания настройки — `string`.

Значение по умолчанию — `59`.

Примечание. См. подробную информацию о версиях биометрических шаблонах в разделе «Нейронные сети».

12.10.14.4 default_human_descriptor_version

Параметр задает используемую версию биометрического шаблона тела.

См. подробную информацию о версиях биометрических шаблонах в разделе «Нейронные сети».

Формат задания настройки — `string`.

Значение по умолчанию — `116`.

Примечание. См. подробную информацию о версиях биометрических шаблонах в разделе «Нейронные сети».

12.11 Настройки сервиса Python Matcher Proxy

Данный раздел описывает параметры сервиса Python Matcher Proxy.

Настройку сервиса можно выполнить с помощью сервиса Configurator.

12.11.1 Группа параметров LUNA_CONFIGURATOR

Данная группа параметров задаёт настройки подключения сервиса Python Matcher Proxy к сервису Configurator.

Данная группа параметров не будет видна в пользовательском интерфейсе сервиса Configurator. Параметры можно изменить только в конфигурационном файле «config.conf», расположенном в директории «/srv/luna_python_matcher/configs/» соответствующего контейнера.

12.11.1.1 use_configurator

Параметр позволяет включить использование сервиса Configurator.

> С помощью сервиса Configurator упрощается настройка сервисов LP. Сервис сохраняет все необходимые конфигурации для всех сервисов LP в одном месте.

При отключенном параметре будут использованы настройки из файла «config.conf», расположенного в директории «/srv/luna_python_matcher/configs/» соответствующего контейнера.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.11.1.2 luna_configurator_origin

Параметр задает протокол, IP адрес и порт сервиса Configurator.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5070`.

12.11.1.3 luna_configurator_api

Параметр задает версию API сервиса Configurator. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.11.2 Группа параметров LUNA_PYTHON_MATCHER_PROXY_LOGGER

Данная группа параметров задает настройки логирования.

12.11.2.1 log_level

Параметр задает уровень отладочной печати, по приоритету: «ERROR», «WARNING», «INFO», «DEBUG».

Формат задания настройки — `string`.

Значение по умолчанию — `INFO`.

12.11.2.2 log_time

Параметр задает формат времени, используемый в записях лога. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.11.2.3 log_to_stdout

Параметр позволяет отправлять логи в стандартный вывод (stdout).

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`

12.11.2.4 log_to_file

Параметр позволяет сохранять логи в файл. Директория с файлами логов указывается в параметре «folder_with_logs».

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.11.2.5 folder_with_logs

Параметр задает директорию, в которой хранятся логи. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить параметр «log_to_file».

Формат задания настройки — `string`.

Значение по умолчанию — `./`

Пример:

```
  "folder_with_logs": "/srv/logs"
```

12.11.2.6 max_log_file_size

Параметр задает максимальный размер файла лога в МБ перед выполнением его ротации (0 — не использовать ротацию).

Для использования данного параметра требуется включить параметр «log_to_file».

> При необходимости можно настроить ротацию логов Docker. См. раздел «Настройка ротации логов Docker» в руководстве по установке LUNA PLATFORM.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024`

12.11.2.7 multiline_stack_trace

Параметр включает многострочную трассировку стека в логах. Когда параметр включен, информация о стеке вызовов записывается в логах так, что каждый фрейм стека помещается на отдельной строке, что улучшает читаемость. Если параметр выключен, информация о стеке вызовов записывается на одной строке, что может сделать логи менее удобными для анализа.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.11.2.8 format

Параметр определяет формат выводимых логов. Доступны следующие значения:

- «default» — стандартный формат вывода логов LUNA PLATFORM
- «json» — вывод логов в формате json
- «ecs» — вывод логов в формате ECS (Elastic Common Schema)

При использовании значения «ecs» будут использоваться следующие поля:

- «http.response.status_code» — содержит код состояния ответа HTTP (200, 404, 500 и т.д.);
- «http.response.execution_time» — содержит информацию о времени, затраченном на выполнение запроса и получение ответа;
- «http.request.method» — содержит метод HTTP-запроса (GET, POST, PUT и т.д.);
- «url.path» — содержит путь в URL-адресе запроса;
- «error.code» — содержит код ошибки, если запрос завершается с ошибкой.

Формат задания настройки — `string`.

Значение по умолчанию — `default`.

12.11.3 Группа параметров LUNA_MONITORING

Данная группа параметров задает настройки мониторинга.

См. подробную информацию о мониторинге в разделе «Мониторинг».

12.11.3.1 storage_type

Тип хранилища для хранения данных мониторинга.

В настоящее время доступно только БД Influx.

Формат задания настройки — `string`.

Значение по умолчанию — `influx`.

12.11.3.2 send_data_for_monitoring

Параметр позволяет включить или отключить отправку данных для мониторинга в InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.11.3.3 use_ssl

Параметр позволяет использовать HTTPS для подключения к InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.11.3.4 organization

Параметр задает рабочую область InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.11.3.5 token

Параметр задает аутентификации InfluxDB 2.x.

Формат задания настройки — `string`.

12.11.3.6 bucket

Параметр задаёт имя бакета InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_monitoring`.

12.11.3.7 host

Параметр задаёт IP-адрес InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.11.3.8 port

Параметр задает порт InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `8086`.

12.11.3.9 flushing_period

Параметр задает частоту отправки данных мониторинга в InfluxDB.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `1`.

12.11.4 Группа параметров PLATFORM_LIMITS

Данная группа параметров определяет ограничения, связанные с операциями сравнения лиц, событий или атрибутов. Эти параметры управляют количеством объектов, которые могут быть включены в операцию сравнения, а также ограничивают количество результатов, возвращаемых в ответе.

Эти параметры полезны для балансировки производительности и использования ресурсов.

Параметры из подгруппы «match» определяют ограничения, связанные с операциями сравнения.

Параметры из подгруппы «cross_match» определяют ограничения, связанные с операциями перекрестного сравнения (см. описание задачи «Cross-matching»).

12.11.4.1 match > array_filter_limit

Параметр задает максимальное количество объектов (максимальный размер массива), указанных в фильтрах «face_ids», «event_ids» и «attribute_ids».

Формат задания настройки — `integer`.

Значение по умолчанию — `1000`.

12.11.4.2 match > reference_limit

Параметр задает максимальное количество эталонов (максимальный размер массива), которых можно указать в запросе на сравнение.

> Необходимо подбирать значение данного параметра в соответствии со своими бизнес-кейсами, т.к. он значительно потребляет ресурсы системы. Так, например, не следует использовать значение по умолчанию «30», если необходимо работать только с тремя эталонами. См. дополнительные советы по оптимизации ресурсов в разделе «Оптимизация ресурсов».

Формат задания настройки — `integer`.

Значение по умолчанию — `30`.

12.11.4.3 match > candidate_limit

Параметр задает максимальное количество кандидатов (максимальный размер массива), которых можно указать в запросе на сравнение.

> Необходимо подбирать значение данного параметра в соответствии со своими бизнес-кейсами, т.к. он значительно потребляет ресурсы системы. Так, например, не следует использовать значение по умолчанию «30», если необходимо работать только с тремя кандидатами. См. дополнительные советы по оптимизации ресурсов в разделе «Оптимизация ресурсов».

Формат задания настройки — `integer`.

Значение по умолчанию — `30`.

12.11.4.4 match > result_candidate_limit

Параметр задает максимальное количество результатов сравнения (максимальный размер массива) для списка кандидатов, возвращаемых в ответе запроса на сравнение.

В теле запроса на сравнение также можно ограничить количество возвращаемых кандидатов в ответе с помощью параметра «limit», однако параметр «result_candidate_limit» имеет более высокий приоритет. Это означает, что при значении «result_candidate_limit» = 4 и «limit» = 6, будет возвращена ошибка с кодом «31007».

Формат задания настройки — `integer`.

Значение по умолчанию — `100`.

12.11.4.5 cross_match > short_array_filter_limit

Параметр задает максимальное количество объектов (максимальный размер массива), указанных во всех фильтрах, кроме фильтров «face_ids», «event_ids» и «attribute_ids» в задаче Cross-matching.

> Например, максимальный размер массива для фильтров «sources», «tags», «cities» и др. В спецификации OpenAPI такие фильтры помечены примером `[ 1 .. 1000 ] items`.

Формат задания настройки — `integer`.

Значение по умолчанию — `1000`.

12.11.4.6 cross_match > array_filter_limit

Параметр задает максимальное количество объектов (максимальный размер массива), указанных в фильтрах «face_ids», «event_ids» и «attribute_ids» в задаче Cross-matching.

> В спецификации OpenAPI такие фильтры помечены примером `[ 1 .. 20000 ] items`.

Формат задания настройки — `integer`.

Значение по умолчанию — `20000`.

12.11.4.7 cross_match > result_candidate_limit

Параметр задает максимальное количество результатов сравнения (максимальный размер массива) для списка кандидатов, возвращаемых в ответе запроса на получение результата задачи Cross-matching.

В теле запроса на сравнение также можно ограничить количество возвращаемых кандидатов в ответе с помощью параметра «limit», однако параметр «result_candidate_limit» имеет более высокий приоритет. Это означает, что при значении «result_candidate_limit» = 4 и «limit» = 6, будет возвращена ошибка с кодом «31007».

Формат задания настройки — `integer`.

Значение по умолчанию — `20000`.

12.11.4.8 cross_match > general_limit

Параметр задает общее ограничение на количество перекрестных сравнений. Он применяется к произведению количества кандидатов и эталонов, которые участвуют в операции сравнения с учетом фильтров. Если произведение превышает значение «general_limit», запрос завершится ошибкой.

Например, если значение «general_limit» равно «20», количество кандидатов-лиц в массиве «face_ids» равно «7», количество эталонов-лиц в массиве «face_ids» равно «5» и все остальные фильтры не сокращают реальное количество кандидатов-лиц, то запрос на перекрестное сравнение не будет выполнен, т.к. общее количество перекрестных сравнений будет равно «35» (5 x 7). Если же какой-то фильтр сокращает общее количество кандидатов-лиц на «3», то запрос будет выполнен успешно, т.к. общее количество перекрестных сравнений будет равно «20» (5 x (7 - 3)).

Формат задания настройки — `integer`.

Значение по умолчанию — `100000000`.

12.11.5 Группа параметров LUNA_PYTHON_MATCHER_DB

В данной группе параметров задаются настройки подключения к базе данных сервиса Python-Matcher.

12.11.5.1 db_type

Параметр задает тип базы данных. Доступны следующие типы:

- «postgres» — тип СУБД PostgreSQL
- «oracle» — тип СУБД Oracle

Формат задания настройки — `string`.

Значение по умолчанию — `postgres`.

12.11.5.2 db_host

Параметр задает имя сервера (хост) базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.11.5.3 db_port

Параметр задает порт базы данных.

Порт по умолчанию для типа «postgres»- `5432`.

Порт по умолчанию для типа «oracle» — `1521`.

Формат задания настройки — `string`.

Значение по умолчанию — `5432`.

12.11.5.4 db_user

Параметр задает имя пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.11.5.5 db_password

Параметр задает пароль пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.11.5.6 db_name

Параметр задает имя базы данных для типа «postgres» или имя SID для типа «oracle».

Формат задания настройки — `string`.

Значение по умолчанию — `luna_python_matcher`.

12.11.5.7 connection_pool_size

Параметр задает размер пула соединений к БД. Реальное количество подключений может быть больше значения данной настройки на 1.

При необходимости в настройке «max_connections» конфигурационного файла PostgreSQL можно задать максимальное количество одновременных подключений к серверу БД. См. раздел «Продвинутая настройка PostrgeSQL» для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `10`.

12.11.5.8 dsn

Параметр задает строку подключения DSN для соединения с БД.

> DSN — это строка подключения, которая идентифицирует и указывает на источник данных (базу данных), к которому нужно установить соединение.

В строке DSN могут задаваться такие настройки, как множественные хосты, аутентификационные данные, порт и другие параметры.

> Настройки зависят от типа БД. Множественные хосты поддерживаются только с PostgreSQL.

> По умолчанию параметр «dsn» не отображается во вкладке «Settings» в Configurator. Проверить список всех доступных параметров для группы настроек можно на вкладке «Limitations».

Пример:

```
{
"dsn": "luna:luna@postgres01:5432,postgres02:5432/luna_python_matcher?
    some_option=some_value"
```

```
  "db_settings": {
      "connection_pool_size": 5
      }
  }
```

Здесь:

- «luna:luna» — имя пользователя и пароль для подключения к PostgreSQL;
- «@postgres01:5432,postgres02:5432» — список хостов и портов, разделенных запятыми. Это означает, что сервис будет пытаться подключиться к первому хосту («postgres01») на порту 5432. Если это не удастся, она попытается подключиться ко второму хосту («postgres02») также на порту 5432;
- «/luna_python_matcher» — название базы данных;
- «?some_option=some_value» — опциональные параметры для подключения.

При необходимости можно комбинировать строку DSN и классические настройки, однако строка DSN является более приоритетной. Можно частично заполнить строку DSN (например, «postgres01,postgres02/luna_python_matcher»), и тогда недостающие параметры будут заполнены из значений параметров «db_host», «db_port», «db_name», «db_user» и «db_password».

При запуске сервис создаст пул подключений к одному из доступных хостов DSN. В случае возникновения проблем с установлением соединения после нескольких неудачных попыток, сервис снова попытается настроить пул подключений к любому из доступных хостов DSN.

12.11.6 Группа параметров LUNA_PROXY_TO_PYTHON_MATCHER_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Python Matcher Proxy.

12.11.6.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Python Matcher Proxy. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.11.6.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `300`.

12.11.6.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.11.6.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `300`.

12.11.7 Группа параметров LUNA_PYTHON_MATCHER_PROXY_HTTP_SETTINGS

В данной группе параметров содержатся настройки, отвечающие за обработку HTTP-подключений. См. подробную информацию по следующей ссылке: https://sanic.dev/en/guide/deployment/configuration.html#builtin-values

12.11.7.1 request_timeout

Параметр задает продолжительность времени между моментом, когда новое открытое TCP-соединение передается на сервер, и моментом, когда получен весь HTTP-запрос.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.11.7.2 response_timeout

Параметр задает продолжительность времени между моментом, когда сервер передает HTTP-запрос приложению, и моментом, когда HTTP-ответ отправляется клиенту.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `600`.

12.11.7.3 request_max_size

Параметр задает максимальный размер запроса.

Формат задания настройки — `integer` (байты).

Значение по умолчанию — `1073741824`.

12.11.7.4 keep_alive_timeout

Параметр задает тайм-аут поддержания активности HTTP.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `15`.

12.11.7.5 luna_python_matcher_proxy_active_plugins

Параметр задает список плагинов, которые должен использовать сервис.

Имена задаются в следующем формате:

```
[
   "plugin_1",
   "plugin_2",
   "plugin_3"
]
```

> Список должен содержать имена файлов без расширения (.py).

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.11.8 Группа параметров LUNA_FACES_DB

В данной группе параметров задаются настройки подключения к базе данных сервиса Faces.

12.11.8.1 db_type

Параметр задает тип базы данных. Доступны следующие типы:

- «postgres» — тип СУБД PostgreSQL
- «oracle» — тип СУБД Oracle

Формат задания настройки — `string`.

Значение по умолчанию — `postgres`.

12.11.8.2 db_host

Параметр задает имя сервера (хост) базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.11.8.3 db_port

Параметр задает порт базы данных.

Порт по умолчанию для типа «postgres»- `5432`.

Порт по умолчанию для типа «oracle» — `1521`.

Формат задания настройки — `string`.

Значение по умолчанию — `5432`.

12.11.8.4 db_user

Параметр задает имя пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.11.8.5 db_password

Параметр задает пароль пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.11.8.6 db_name

Параметр задает имя базы данных для типа «postgres» или имя SID для типа «oracle».

Формат задания настройки — `string`.

Значение по умолчанию — `luna_faces`.

12.11.8.7 connection_pool_size

Параметр задает размер пула соединений к БД. Реальное количество подключений может быть больше значения данной настройки на 1.

При необходимости в настройке «max_connections» конфигурационного файла PostgreSQL можно задать максимальное количество одновременных подключений к серверу БД. См. раздел «Продвинутая настройка PostrgeSQL» для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `10`.

12.11.8.8 dsn

Параметр задает строку подключения DSN для соединения с БД.

> DSN — это строка подключения, которая идентифицирует и указывает на источник данных (базу данных), к которому нужно установить соединение.

В строке DSN могут задаваться такие настройки, как множественные хосты, аутентификационные данные, порт и другие параметры.

> Настройки зависят от типа БД. Множественные хосты поддерживаются только с PostgreSQL.

> По умолчанию параметр «dsn» не отображается во вкладке «Settings» в Configurator. Проверить список всех доступных параметров для группы настроек можно на вкладке «Limitations».

Пример:

```
{
"dsn": "luna:luna@postgres01:5432,postgres02:5432/luna_faces?some_option=
    some_value"
"db_settings": {
    "connection_pool_size": 5
    }
}
```

Здесь:

- «luna:luna» — имя пользователя и пароль для подключения к PostgreSQL;
- «@postgres01:5432,postgres02:5432» — список хостов и портов, разделенных запятыми. Это означает, что сервис будет пытаться подключиться к первому хосту («postgres01») на порту 5432. Если это не удастся, она попытается подключиться ко второму хосту («postgres02») также на порту 5432;
- «/luna_faces» — название базы данных;
- «?some_option=some_value» — опциональные параметры для подключения.

При необходимости можно комбинировать строку DSN и классические настройки, однако строка DSN является более приоритетной. Можно частично заполнить строку DSN (например, «postgres01,postgres02/luna_faces»), и тогда недостающие параметры будут заполнены из значений параметров «db_host», «db_port», «db_name», «db_user» и «db_password».

При запуске сервис создаст пул подключений к одному из доступных хостов DSN. В случае возникновения проблем с установлением соединения после нескольких неудачных попыток, сервис

снова попытается настроить пул подключений к любому из доступных хостов DSN.

12.11.9 Группа параметров LUNA_EVENTS_DB

В данной группе параметров задаются настройки подключения к базе данных сервиса Events.

12.11.9.1 db_type

Параметр задает тип базы данных. Доступны следующие типы:

- «postgres» — тип СУБД PostgreSQL
- «oracle» — тип СУБД Oracle

Формат задания настройки — `string`.

Значение по умолчанию — `postgres`.

12.11.9.2 db_host

Параметр задает имя сервера (хост) базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.11.9.3 db_port

Параметр задает порт базы данных.

Порт по умолчанию для типа «postgres»- `5432`.

Порт по умолчанию для типа «oracle» — `1521`.

Формат задания настройки — `string`.

Значение по умолчанию — `5432`.

12.11.9.4 db_user

Параметр задает имя пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.11.9.5 db_password

Параметр задает пароль пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.11.9.6 db_name

Параметр задает имя базы данных для типа «postgres» или имя SID для типа «oracle».

Формат задания настройки — `string`.

Значение по умолчанию — `luna_events`.

12.11.9.7 connection_pool_size

Параметр задает размер пула соединений к БД. Реальное количество подключений может быть больше значения данной настройки на 1.

При необходимости в настройке «max_connections» конфигурационного файла PostgreSQL можно задать максимальное количество одновременных подключений к серверу БД. См. раздел «Продвинутая настройка PostrgeSQL» для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `10`.

12.11.9.8 dsn

Параметр задает строку подключения DSN для соединения с БД.

> DSN — это строка подключения, которая идентифицирует и указывает на источник данных (базу данных), к которому нужно установить соединение.

В строке DSN могут задаваться такие настройки, как множественные хосты, аутентификационные данные, порт и другие параметры.

> Настройки зависят от типа БД. Множественные хосты поддерживаются только с PostgreSQL.

> По умолчанию параметр «dsn» не отображается во вкладке «Settings» в Configurator. Проверить список всех доступных параметров для группы настроек можно на вкладке «Limitations».

Пример:

```
{
"dsn": "luna:luna@postgres01:5432,postgres02:5432/luna_events?some_option=
    some_value"
"db_settings": {
    "connection_pool_size": 5
    }
}
```

Здесь:

- «luna:luna» — имя пользователя и пароль для подключения к PostgreSQL;

- «@postgres01:5432,postgres02:5432» — список хостов и портов, разделенных запятыми. Это означает, что сервис будет пытаться подключиться к первому хосту («postgres01») на порту 5432. Если это не удастся, она попытается подключиться ко второму хосту («postgres02») также на порту 5432;
- «/luna_events» — название базы данных;
- «?some_option=some_value» — опциональные параметры для подключения.

При необходимости можно комбинировать строку DSN и классические настройки, однако строка DSN является более приоритетной. Можно частично заполнить строку DSN (например, «postgres01,postgres02/luna_events»), и тогда недостающие параметры будут заполнены из значений параметров «db_host», «db_port», «db_name», «db_user» и «db_password».

При запуске сервис создаст пул подключений к одному из доступных хостов DSN. В случае возникновения проблем с установлением соединения после нескольких неудачных попыток, сервис снова попытается настроить пул подключений к любому из доступных хостов DSN.

12.11.10 Группа параметров LUNA_ATTRIBUTES_DB

12.11.10.1 user

Параметр задает имя пользователя БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.11.10.2 password

Параметр задает пароль БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию не указывается.

12.11.10.3 host

Параметр задает IP-адрес БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.11.10.4 port

Параметр задает номер порта, на котором Redis ожидает входящие сетевые соединения и прослушивает их для выполнения команд от клиентов.

Формат задания настройки — `integer`.

Значение по умолчанию — `6379`.

12.11.10.5 sentinel > master_name

Параметр задает имя мастера базы данных Redis, который контролируется и управляется системой Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_attributes`.

12.11.10.6 sentinel > sentinels

Параметр задает список адресов и портов серверов Sentinel, которые будут использоваться клиентами для обнаружения и мониторинга БД Redis.

Формат задания настройки — `list > string`.

Значение по умолчанию — `[]`.

12.11.10.7 sentinel > user

Параметр задает имя пользователя сервера Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.11.10.8 sentinel > password

Параметр задает пароль пользователя сервера Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.11.11 Группа параметров ADDITIONAL_SERVICES_USAGE

12.11.11.1 luna_events

Параметр задает возможность использования сервиса Events.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.11.11.2 luna_faces

Параметр задает возможность использования сервиса Faces.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.11.11.3 luna_handlers

Параметр задает возможность использования сервиса Handlers.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.11.11.4 luna_sender

Параметр задает возможность использования сервиса Sender.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.11.11.5 luna_matcher_proxy

Параметр задает возможность использования сервиса Python Matcher Proxy.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.11.11.6 luna_image_store

Параметр задает возможность использования сервиса Image Store.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.11.11.7 luna_lambda

Параметр задает возможность использования сервиса Lambda.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.11.11.8 luna_video_manager

Параметр задает возможность использования сервиса Video Manager.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.11.11.9 luna_video_agent

Параметр задает возможность использования сервиса Video Agent.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.11.11.10 luna_streams_retranslator

Параметр задает возможность использования сервиса Streams Retranslator.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.11.12 Группа параметров LUNA_SERVICE_METRICS

Данная группа параметров включает и настраивает сбор метрик в формате Prometheus.

См. подробную информацию в разделе «Мониторинг».

12.11.12.1 enabled

Параметр включает сбор метрик.

Если сбор метрик отключен, то запрос к ресурсу `/metrics` вернет соответствующее сообщение.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.11.12.2 metrics_format

Параметр задает формат метрик.

На данный момент поддерживается только формат Prometheus.

См. официальную документацию Prometheus для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `prometheus`.

12.11.12.3 extra_labels

Параметр задает пользовательские типы лейблов.

Формат задания настройки — `label_name=label_value`.

Значение по умолчанию не задано.

12.11.13 Прочие

12.11.13.1 storage_time

Параметр задает формат времени, используемый для записей в базе данных. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.11.13.2 default_face_descriptor_version

Параметр задает используемую версию биометрического шаблона лица.

Формат задания настройки — `string`.

Значение по умолчанию — `59`.

Примечание. См. подробную информацию о версиях биометрических шаблонах в разделе «Нейронные сети».

12.11.13.3 default_human_descriptor_version

Параметр задает используемую версию биометрического шаблона тела.

См. подробную информацию о версиях биометрических шаблонах в разделе «Нейронные сети».

Формат задания настройки — `string`.

Значение по умолчанию — `116`.

Примечание. См. подробную информацию о версиях биометрических шаблонах в разделе «Нейронные сети».

12.12 Настройки сервиса Handlers

Данный раздел описывает параметры сервиса Handlers.

Настройку сервиса можно выполнить с помощью сервиса Configurator.

12.12.1 Группа параметров LUNA_CONFIGURATOR

Данная группа параметров задаёт настройки подключения сервиса Handlers к сервису Configurator.

Данная группа параметров не будет видна в пользовательском интерфейсе сервиса Configurator. Параметры можно изменить только в конфигурационном файле «config.conf», расположенном в директории «/srv/luna_handlers/configs/» соответствующего контейнера.

12.12.1.1 use_configurator

Параметр позволяет включить использование сервиса Configurator.

> С помощью сервиса Configurator упрощается настройка сервисов LP. Сервис сохраняет все необходимые конфигурации для всех сервисов LP в одном месте.

При отключенном параметре будут использованы настройки из файла «config.conf», расположенного в директории «/srv/luna_handlers/configs/» соответствующего контейнера.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.12.1.2 luna_configurator_origin

Параметр задает протокол, IP адрес и порт сервиса Configurator.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5070`.

12.12.1.3 luna_configurator_api

Параметр задает версию API сервиса Configurator. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.12.2 Группа параметров LUNA_HANDLERS_DB

В данной группе параметров задаются настройки подключения к базе данных сервиса Handlers.

12.12.2.1 db_type

Параметр задает тип базы данных. Доступны следующие типы:

- «postgres» — тип СУБД PostgreSQL
- «oracle» — тип СУБД Oracle

Формат задания настройки — `string`.

Значение по умолчанию — `postgres`.

12.12.2.2 db_host

Параметр задает имя сервера (хост) базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.12.2.3 db_port

Параметр задает порт базы данных.

Порт по умолчанию для типа «postgres»- `5432`.

Порт по умолчанию для типа «oracle» — `1521`.

Формат задания настройки — `string`.

Значение по умолчанию — `5432`.

12.12.2.4 db_user

Параметр задает имя пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.12.2.5 db_password

Параметр задает пароль пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.12.2.6 db_name

Параметр задает имя базы данных для типа «postgres» или имя SID для типа «oracle».

Формат задания настройки — `string`.

Значение по умолчанию — `luna_handlers`.

12.12.2.7 connection_pool_size

Параметр задает размер пула соединений к БД. Реальное количество подключений может быть больше значения данной настройки на 1.

При необходимости в настройке «max_connections» конфигурационного файла PostgreSQL можно задать максимальное количество одновременных подключений к серверу БД. См. раздел «Продвинутая настройка PostrgeSQL» для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `10`.

12.12.2.8 dsn

Параметр задает строку подключения DSN для соединения с БД.

> DSN — это строка подключения, которая идентифицирует и указывает на источник данных (базу данных), к которому нужно установить соединение.

В строке DSN могут задаваться такие настройки, как множественные хосты, аутентификационные данные, порт и другие параметры.

> Настройки зависят от типа БД. Множественные хосты поддерживаются только с PostgreSQL.

> По умолчанию параметр «dsn» не отображается во вкладке «Settings» в Configurator. Проверить список всех доступных параметров для группы настроек можно на вкладке «Limitations».

Пример:

```
{
"dsn": "luna:luna@postgres01:5432,postgres02:5432/luna_handlers?some_option=
    some_value"
"db_settings": {
     "connection_pool_size": 5
     }
}
```

Здесь:

- «luna:luna» — имя пользователя и пароль для подключения к PostgreSQL;
- «@postgres01:5432,postgres02:5432» — список хостов и портов, разделенных запятыми. Это означает, что сервис будет пытаться подключиться к первому хосту («postgres01») на порту 5432. Если это не удастся, она попытается подключиться ко второму хосту («postgres02») также на порту 5432;
- «/luna_handlers» — название базы данных;
- «?some_option=some_value» — опциональные параметры для подключения.

При необходимости можно комбинировать строку DSN и классические настройки, однако строка DSN является более приоритетной. Можно частично заполнить строку DSN (например, «postgres01,postgres02/luna_handlers»), и тогда недостающие параметры будут заполнены из значений параметров «db_host», «db_port», «db_name», «db_user» и «db_password».

При запуске сервис создаст пул подключений к одному из доступных хостов DSN. В случае возникновения проблем с установлением соединения после нескольких неудачных попыток, сервис снова попытается настроить пул подключений к любому из доступных хостов DSN.

12.12.3 Группа параметров LUNA_MONITORING

Данная группа параметров задает настройки мониторинга.

См. подробную информацию о мониторинге в разделе «Мониторинг».

12.12.3.1 storage_type

Тип хранилища для хранения данных мониторинга.

В настоящее время доступно только БД Influx.

Формат задания настройки — `string`.

Значение по умолчанию — `influx`.

12.12.3.2 send_data_for_monitoring

Параметр позволяет включить или отключить отправку данных для мониторинга в InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.12.3.3 use_ssl

Параметр позволяет использовать HTTPS для подключения к InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.12.3.4 organization

Параметр задает рабочую область InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.12.3.5 token

Параметр задает аутентификации InfluxDB 2.x.

Формат задания настройки — `string`.

12.12.3.6 bucket

Параметр задаёт имя бакета InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_monitoring`.

12.12.3.7 host

Параметр задаёт IP-адрес InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.12.3.8 port

Параметр задает порт InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `8086`.

12.12.3.9 flushing_period

Параметр задает частоту отправки данных мониторинга в InfluxDB.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `1`.

12.12.4 Группа параметров LUNA_HANDLERS_LOGGER

Данная группа параметров задает настройки логирования.

12.12.4.1 log_level

Параметр задает уровень отладочной печати, по приоритету: «ERROR», «WARNING», «INFO», «DEBUG».

Формат задания настройки — `string`.

Значение по умолчанию — `INFO`.

12.12.4.2 log_time

Параметр задает формат времени, используемый в записях лога. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.12.4.3 log_to_stdout

Параметр позволяет отправлять логи в стандартный вывод (stdout).

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`

12.12.4.4 log_to_file

Параметр позволяет сохранять логи в файл. Директория с файлами логов указывается в параметре «folder_with_logs».

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.12.4.5 folder_with_logs

Параметр задает директорию, в которой хранятся логи. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить параметр «log_to_file».

Формат задания настройки — `string`.

Значение по умолчанию — `./`

Пример:

```
  "folder_with_logs": "/srv/logs"
```

12.12.4.6 max_log_file_size

Параметр задает максимальный размер файла лога в МБ перед выполнением его ротации (0 — не использовать ротацию).

Для использования данного параметра требуется включить параметр «log_to_file».

> При необходимости можно настроить ротацию логов Docker. См. раздел «Настройка ротации логов Docker» в руководстве по установке LUNA PLATFORM.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024`

12.12.4.7 multiline_stack_trace

Параметр включает многострочную трассировку стека в логах. Когда параметр включен, информация о стеке вызовов записывается в логах так, что каждый фрейм стека помещается на отдельной строке, что улучшает читаемость. Если параметр выключен, информация о стеке вызовов записывается на одной строке, что может сделать логи менее удобными для анализа.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.12.4.8 format

Параметр определяет формат выводимых логов. Доступны следующие значения:

- «default» — стандартный формат вывода логов LUNA PLATFORM
- «json» — вывод логов в формате json
- «ecs» — вывод логов в формате ECS (Elastic Common Schema)

При использовании значения «ecs» будут использоваться следующие поля:

- «http.response.status_code» — содержит код состояния ответа HTTP (200, 404, 500 и т.д.);
- «http.response.execution_time» — содержит информацию о времени, затраченном на выполнение запроса и получение ответа;
- «http.request.method» — содержит метод HTTP-запроса (GET, POST, PUT и т.д.);
- «url.path» — содержит путь в URL-адресе запроса;
- «error.code» — содержит код ошибки, если запрос завершается с ошибкой.

Формат задания настройки — `string`.

Значение по умолчанию — `default`.

12.12.5 Группа параметров LUNA_REMOTE_SDK_ADDRESS

Данная группа параметров задает настройки подключения к сервису Remote SDK.

12.12.5.1 origin

Параметр задает протокол, IP адрес и порт сервиса Remote SDK.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Remote SDK, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Remote SDK.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5220`.

12.12.5.2 api_version

Параметр задает версию API сервиса Remote SDK. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.12.6 Группа параметров LUNA_REMOTE_SDK_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Remote SDK.

12.12.6.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Remote SDK. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.12.6.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.12.6.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.12.6.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.12.7 Группа параметров LUNA_FACES_ADDRESS

Данная группа параметров задает настройки подключения к сервису Faces.

12.12.7.1 origin

Параметр задает протокол, IP адрес и порт сервиса Faces.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Faces, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Faces.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5030`.

12.12.7.2 api_version

Параметр задает версию API сервиса Faces. Доступная версия API — «3».

Формат задания настройки — `integer`.

Значение по умолчанию — `3`.

12.12.8 Группа параметров LUNA_FACES_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Faces.

12.12.8.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Faces. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.12.8.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.12.8.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.12.8.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.12.9 Группа параметров LUNA_LAMBDA_UNIT_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Lambda Unit.

12.12.9.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Lambda Unit. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.12.9.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.12.9.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.12.9.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.12.10 Группа параметров LUNA_IMAGE_STORE_FACES_SAMPLES_ADDRESS

В данной группе параметров задаются настройки бакета для хранения БО лиц.

12.12.10.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.12.10.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.12.10.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-samples`.

12.12.11 Группа параметров LUNA_IMAGE_STORE_FACES_SAMPLES_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к бакету, хранящему БО лиц.

12.12.11.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к бакету, хранящему БО лиц. Это время ожидания, в течение которого клиент пытается установить соединение с бакетом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `30`.

12.12.11.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.12.12 Группа параметров LUNA_IMAGE_STORE_BODIES_SAMPLES_ADDRESS

В данной группе параметров задаются настройки бакета для хранения БО тел.

12.12.12.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.12.12.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.12.12.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-bodies-samples`.

12.12.13 Группа параметров LUNA_IMAGE_STORE_BODIES_SAMPLES_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к бакету, хранящему БО тел.

12.12.13.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к бакету, хранящему БО тел. Это время ожидания, в течение которого клиент пытается установить соединение с бакетом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `30`.

12.12.13.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.12.14 Группа параметров LUNA_IMAGE_STORE_IMAGES_ADDRESS

В данной группе параметров задаются настройки бакета для хранения исходных изображений.

12.12.14.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.12.14.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.12.14.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-image-origin`.

12.12.15 Группа параметров LUNA_IMAGE_STORE_IMAGES_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к бакету, хранящему исходных изображений.

12.12.15.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к бакету, хранящему исходных изображений. Это время ожидания, в течение которого клиент пытается установить соединение с бакетом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `30`.

12.12.15.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.12.16 Группа параметров ADDITIONAL_SERVICES_USAGE

12.12.16.1 luna_events

Параметр задает возможность использования сервиса Events.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.12.16.2 luna_handlers

Параметр задает возможность использования сервиса Handlers.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.12.16.3 luna_faces

Параметр задает возможность использования сервиса Faces.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.12.16.4 luna_sender

Параметр задает возможность использования сервиса Sender.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.12.16.5 luna_matcher_proxy

Параметр задает возможность использования сервиса Python Matcher Proxy.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.12.16.6 luna_image_store

Параметр задает возможность использования сервиса Image Store.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.12.16.7 luna_lambda

Параметр задает возможность использования сервиса Lambda.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.12.16.8 luna_video_manager

Параметр задает возможность использования сервиса Video Manager.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.12.16.9 luna_video_agent

Параметр задает возможность использования сервиса Video Agent.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.12.16.10 luna_streams_retranslator

Параметр задает возможность использования сервиса Streams Retranslator.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.12.17 Группа параметров LUNA_EVENTS_ADDRESS

Данная группа параметров задает настройки подключения к сервису Events.

12.12.17.1 origin

Параметр задает протокол, IP адрес и порт сервиса Events.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Events, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Events.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5040`.

12.12.17.2 api_version

Параметр задает версию API сервиса Events. Доступная версия API — «2».

Формат задания настройки — `integer`.

Значение по умолчанию — `2`.

12.12.18 Группа параметров LUNA_EVENTS_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Events.

12.12.18.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Events. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.12.18.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.12.18.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.12.18.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.12.19 Группа параметров LUNA_PYTHON_MATCHER_ADDRESS

Данная группа параметров задает настройки подключения к сервису Python Matcher.

12.12.19.1 origin

Параметр задает протокол, IP адрес и порт сервиса Python Matcher.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Python Matcher, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Python Matcher.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5100`.

12.12.19.2 api_version

Параметр задает версию API сервиса Python Matcher. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.12.20 Группа параметров LUNA_PYTHON_MATCHER_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Python Matcher.

12.12.20.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Python Matcher. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.12.20.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.12.20.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.12.20.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.12.21 Группа параметров LUNA_MATCHER_PROXY_ADDRESS

Данная группа параметров задает настройки подключения к сервису Python Matcher Proxy.

12.12.21.1 origin

Параметр задает протокол, IP адрес и порт сервиса Python Matcher Proxy.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Python Matcher Proxy, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Python Matcher Proxy.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5110`.

12.12.21.2 api_version

Параметр задает версию API сервиса Python Matcher Proxy. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.12.22 Группа параметров LUNA_PYTHON_MATCHER_PROXY_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Python Matcher Proxy.

12.12.22.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Python Matcher Proxy. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.12.22.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.12.22.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.12.22.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.12.23 Группа параметров REDIS_DB_ADDRESS

12.12.23.1 user

Параметр задает имя пользователя БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.12.23.2 password

Параметр задает пароль БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию не указывается.

12.12.23.3 host

Параметр задает IP-адрес БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.12.23.4 port

Параметр задает номер порта, на котором Redis ожидает входящие сетевые соединения и прослушивает их для выполнения команд от клиентов.

Формат задания настройки — `integer`.

Значение по умолчанию — `6379`.

12.12.23.5 channel

Параметр задает канал Redis-канал, на который сервис подписывается сервис.

Формат задания настройки — `integer`.

Значение по умолчанию — `luna-sender`.

12.12.23.6 sentinel > master_name

Параметр задает имя мастера базы данных Redis, который контролируется и управляется системой Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_sender_master`.

12.12.23.7 sentinel > sentinels

Параметр задает список адресов и портов серверов Sentinel, которые будут использоваться клиентами для обнаружения и мониторинга БД Redis.

Формат задания настройки — `list > string`.

Значение по умолчанию — `[]`.

12.12.23.8 sentinel > user

Параметр задает имя пользователя сервера Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.12.23.9 sentinel > password

Параметр задает пароль пользователя сервера Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.12.24 Группа параметров FETCH_EXTERNAL_IMAGE_TIMEOUTS

Данная группа параметров представляет собой набор параметров для установки таймаутов в процессе выполнения HTTP-запросов к внешним ресурсам для загрузки изображений. Каждый из параметров в этой настройке определяет максимальное время ожидания для определенной операции в процессе выполнения запроса.

12.12.24.1 connect

Параметр задает таймаут для установки соединения с внешним ресурсом. Если соединение не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.12.24.2 request

Параметр задает таймаут для ожидания ответа на HTTP-запрос, который отправляется для получения изображения с внешнего ресурса.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.12.24.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.12.24.4 sock_request

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.12.25 Группа параметров ATTRIBUTES_STORAGE_POLICY

В данной группе параметров задаются настройки для сервиса Handlers, связанные с хранением временных атрибутов.

12.12.25.1 default_ttl

Параметр задает время существования временных атрибутов по умолчанию.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `300`.

12.12.25.2 max_ttl

Параметр задает максимальное время существования временных атрибутов.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `86400`.

12.12.26 Группа параметров LUNA_HANDLERS_LIMITS

Данная группа параметров определяет ограничения для некоторых операций, выполняемых с помощью сервиса Handlers.

Увеличение данных ограничений может привести к уменьшению производительности.

12.12.26.1 received_images_limit

Параметр задает максимальное количество изображений, которые можно указать в запросах «generate events», «detect faces» и «perform verification» с помощью схемы «multipart/form-data».

> Для пакетной обработки изображений необходимо использовать задачу Estimator.

Формат задания настройки — `integer`.

Значение по умолчанию — `8`.

12.12.26.2 raw_event_detections_limit

Параметр задает максимальное количество детекций, которые можно указать в запросе «save event».

Формат задания настройки — `integer`.

Значение по умолчанию — `100`.

12.12.26.3 raw_event_arrays_limit

Параметр задает максимальную длину массивов в теле запроса «save event».

Формат задания настройки — `integer`.

Значение по умолчанию — `30`.

12.12.26.4 result_candidate_limit

Параметр задает максимальное количество результатов сравнения биометрических шаблонов в запросе «generate events».

Формат задания настройки — `integer`.

Значение по умолчанию — `100`.

12.12.27 Группа параметров EXTERNAL_LUNA_API_ADDRESS

Данная группа параметров предназначена для корректной обработки ссылок на объекты, созданные с помощью ресурсов «/images» и «/objects» в сервисе API. В данной секции задается адрес и версия API сервиса API.

Если в качестве входных данных для ресурсов «/detector», «handlers/{handler_id}/events» и «verifiers/{verifier_id}/verification» указывается URL-адрес и версия сервиса API объекта типа «images», совпадающие с адресом и версией API из секции «EXTERNAL_LUNA_API_ADDRESS» сервиса Handlers, то данные объекты будут загружаться с помощью сервиса Image Store напрямую, а не отправлять запрос в сервис API с последующим перенаправлением в сервис Image Store.

Пример формата: `http://10.15.3.144:5000/6/images/141d2706-8baf-433b-82eb-8c7fada847da`, где значение `http://10.15.3.144:5000` должно совпадать со значением из настройки «origin», а значение `6` должно совпадать со значением настройки `api_version` секции «EXTERNAL_LUNA_API_ADDRESS».

Для избежания ошибок необходимо настроить данную секцию в настройках Handlers перед использованием URL-адресов до объектов типа «objects» или «images» в качестве источника входных данных.

12.12.27.1 origin

Параметр задает протокол, IP адрес и порт сервиса API.

См. описание логики работы в разделе «Группа параметров EXTERNAL_LUNA_API_ADDRESS».

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5000`.

12.12.27.2 api_version

Параметр задает версию API сервиса API.

См. описание логики работы в разделе «Группа параметров EXTERNAL_LUNA_API_ADDRESS».

Формат задания настройки — `integer`.

Значение по умолчанию — `6`.

12.12.28 Группа параметров LUNA_HANDLERS_HTTP_SETTINGS

В данной группе параметров содержатся настройки, отвечающие за обработку HTTP-подключений. См. подробную информацию по следующей ссылке: https://sanic.dev/en/guide/deployment/configuration.html#builtin-values

12.12.28.1 request_timeout

Параметр задает продолжительность времени между моментом, когда новое открытое TCP-соединение передается на сервер, и моментом, когда получен весь HTTP-запрос.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.12.28.2 response_timeout

Параметр задает продолжительность времени между моментом, когда сервер передает HTTP-запрос приложению, и моментом, когда HTTP-ответ отправляется клиенту.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `600`.

12.12.28.3 request_max_size

Параметр задает максимальный размер запроса.

Формат задания настройки — `integer` (байты).

Значение по умолчанию — `1073741824`.

12.12.28.4 keep_alive_timeout

Параметр задает тайм-аут поддержания активности HTTP.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `15`.

12.12.29 Группа параметров LUNA_SERVICE_METRICS

Данная группа параметров включает и настраивает сбор метрик в формате Prometheus.

См. подробную информацию в разделе «Мониторинг».

12.12.29.1 enabled

Параметр включает сбор метрик.

Если сбор метрик отключен, то запрос к ресурсу `/metrics` вернет соответствующее сообщение.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.12.29.2 metrics_format

Параметр задает формат метрик.

На данный момент поддерживается только формат Prometheus.

См. официальную документацию Prometheus для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `prometheus`.

12.12.29.3 extra_labels

Параметр задает пользовательские типы лейблов.

Формат задания настройки — `label_name=label_value`.

Значение по умолчанию не задано.

12.12.30 Прочие

12.12.30.1 luna_handlers_active_plugins

Параметр задает список плагинов, которые должен использовать сервис.

Имена задаются в следующем формате:

```
[
   "plugin_1",
   "plugin_2",
   "plugin_3"
]
```

> Список должен содержать имена файлов без расширения (.py).

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.12.30.2 storage_time

Параметр задает формат времени, используемый для записей в базе данных. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.12.30.3 default_face_descriptor_version

Параметр задает используемую версию биометрического шаблона лица.

Формат задания настройки — `string`.

Значение по умолчанию — `59`.

Примечание. См. подробную информацию о версиях биометрических шаблонах в разделе «Нейронные сети».

12.12.30.4 default_human_descriptor_version

Параметр задает используемую версию биометрического шаблона тела.

См. подробную информацию о версиях биометрических шаблонах в разделе «Нейронные сети».

Формат задания настройки — `string`.

Значение по умолчанию — `116`.

Примечание. См. подробную информацию о версиях биометрических шаблонах в разделе «Нейронные сети».

12.13 Настройки сервиса Backport 3

Данный раздел описывает параметры сервиса Backport 3.

Настройку сервиса можно выполнить с помощью сервиса Configurator.

12.13.1 Группа параметров LUNA_CONFIGURATOR

Данная группа параметров задаёт настройки подключения сервиса Backport 3 к сервису Configurator.

Данная группа параметров не будет видна в пользовательском интерфейсе сервиса Configurator. Параметры можно изменить только в конфигурационном файле «config.conf», расположенном в директории «/srv/luna_backport3/configs/» соответствующего контейнера.

12.13.1.1 use_configurator

Параметр позволяет включить использование сервиса Configurator.

> С помощью сервиса Configurator упрощается настройка сервисов LP. Сервис сохраняет все необходимые конфигурации для всех сервисов LP в одном месте.

При отключенном параметре будут использованы настройки из файла «config.conf», расположенного в директории «/srv/luna_backport3/configs/» соответствующего контейнера.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.13.1.2 luna_configurator_origin

Параметр задает протокол, IP адрес и порт сервиса Configurator.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5070`.

12.13.1.3 luna_configurator_api

Параметр задает версию API сервиса Configurator. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.13.2 Группа параметров LUNA_BACKPORT3_DB

В данной группе параметров задаются настройки подключения к базе данных сервиса Backport3.

12.13.2.1 db_type

Параметр задает тип базы данных. Доступны следующие типы:

- «postgres» — тип СУБД PostgreSQL
- «oracle» — тип СУБД Oracle

Формат задания настройки — `string`.

Значение по умолчанию — `postgres`.

12.13.2.2 db_host

Параметр задает имя сервера (хост) базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.13.2.3 db_port

Параметр задает порт базы данных.

Порт по умолчанию для типа «postgres»- `5432`.

Порт по умолчанию для типа «oracle» — `1521`.

Формат задания настройки — `string`.

Значение по умолчанию — `5432`.

12.13.2.4 db_user

Параметр задает имя пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.13.2.5 db_password

Параметр задает пароль пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.13.2.6 db_name

Параметр задает имя базы данных для типа «postgres» или имя SID для типа «oracle».

Формат задания настройки — `string`.

Значение по умолчанию — `luna_backport3`.

12.13.2.7 connection_pool_size

Параметр задает размер пула соединений к БД. Реальное количество подключений может быть больше значения данной настройки на 1.

При необходимости в настройке «max_connections» конфигурационного файла PostgreSQL можно задать максимальное количество одновременных подключений к серверу БД. См. раздел «Продвинутая настройка PostrgeSQL» для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `10`.

12.13.2.8 dsn

Параметр задает строку подключения DSN для соединения с БД.

> DSN — это строка подключения, которая идентифицирует и указывает на источник данных (базу данных), к которому нужно установить соединение.

В строке DSN могут задаваться такие настройки, как множественные хосты, аутентификационные данные, порт и другие параметры.

> Настройки зависят от типа БД. Множественные хосты поддерживаются только с PostgreSQL.

> По умолчанию параметр «dsn» не отображается во вкладке «Settings» в Configurator. Проверить список всех доступных параметров для группы настроек можно на вкладке «Limitations».

Пример:

```
{
"dsn": "luna:luna@postgres01:5432,postgres02:5432/luna_backport3?some_option
   =some_value"
"db_settings": {
    "connection_pool_size": 5
    }
}
```

Здесь:

- «luna:luna» — имя пользователя и пароль для подключения к PostgreSQL;
- «@postgres01:5432,postgres02:5432» — список хостов и портов, разделенных запятыми. Это означает, что сервис будет пытаться подключиться к первому хосту («postgres01») на порту 5432. Если это не удастся, она попытается подключиться ко второму хосту («postgres02») также на порту 5432;
- «/luna_backport3» — название базы данных;
- «?some_option=some_value» — опциональные параметры для подключения.

При необходимости можно комбинировать строку DSN и классические настройки, однако строка DSN является более приоритетной. Можно частично заполнить строку DSN (например, «postgres01,postgres02/luna_backport3»), и тогда недостающие параметры будут заполнены из значений параметров «db_host», «db_port», «db_name», «db_user» и «db_password».

При запуске сервис создаст пул подключений к одному из доступных хостов DSN. В случае возникновения проблем с установлением соединения после нескольких неудачных попыток, сервис снова попытается настроить пул подключений к любому из доступных хостов DSN.

12.13.3 Группа параметров LUNA_MONITORING

Данная группа параметров задает настройки мониторинга.

См. подробную информацию о мониторинге в разделе «Мониторинг».

12.13.3.1 storage_type

Тип хранилища для хранения данных мониторинга.

В настоящее время доступно только БД Influx.

Формат задания настройки — `string`.

Значение по умолчанию — `influx`.

12.13.3.2 send_data_for_monitoring

Параметр позволяет включить или отключить отправку данных для мониторинга в InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.13.3.3 use_ssl

Параметр позволяет использовать HTTPS для подключения к InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.13.3.4 organization

Параметр задает рабочую область InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.13.3.5 token

Параметр задает аутентификации InfluxDB 2.x.

Формат задания настройки — `string`.

12.13.3.6 bucket

Параметр задаёт имя бакета InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_monitoring`.

12.13.3.7 host

Параметр задаёт IP-адрес InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.13.3.8 port

Параметр задает порт InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `8086`.

12.13.3.9 flushing_period

Параметр задает частоту отправки данных мониторинга в InfluxDB.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `1`.

12.13.4 Группа параметров LUNA_BACKPORT3_LOGGER

Данная группа параметров задает настройки логирования.

12.13.4.1 log_level

Параметр задает уровень отладочной печати, по приоритету: «ERROR», «WARNING», «INFO», «DEBUG».

Формат задания настройки — `string`.

Значение по умолчанию — `INFO`.

12.13.4.2 log_time

Параметр задает формат времени, используемый в записях лога. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.13.4.3 log_to_stdout

Параметр позволяет отправлять логи в стандартный вывод (stdout).

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`

12.13.4.4 log_to_file

Параметр позволяет сохранять логи в файл. Директория с файлами логов указывается в параметре «folder_with_logs».

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.13.4.5 folder_with_logs

Параметр задает директорию, в которой хранятся логи. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить параметр «log_to_file».

Формат задания настройки — `string`.

Значение по умолчанию — `./`

Пример:

```
  "folder_with_logs": "/srv/logs"
```

12.13.4.6 max_log_file_size

Параметр задает максимальный размер файла лога в МБ перед выполнением его ротации (0 — не использовать ротацию).

Для использования данного параметра требуется включить параметр «log_to_file».

> При необходимости можно настроить ротацию логов Docker. См. раздел «Настройка ротации логов Docker» в руководстве по установке LUNA PLATFORM.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024`

12.13.4.7 multiline_stack_trace

Параметр включает многострочную трассировку стека в логах. Когда параметр включен, информация о стеке вызовов записывается в логах так, что каждый фрейм стека помещается на отдельной строке, что улучшает читаемость. Если параметр выключен, информация о стеке вызовов записывается на одной строке, что может сделать логи менее удобными для анализа.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.13.4.8 format

Параметр определяет формат выводимых логов. Доступны следующие значения:

- «default» — стандартный формат вывода логов LUNA PLATFORM
- «json» — вывод логов в формате json
- «ecs» — вывод логов в формате ECS (Elastic Common Schema)

При использовании значения «ecs» будут использоваться следующие поля:

- «http.response.status_code» — содержит код состояния ответа HTTP (200, 404, 500 и т.д.);
- «http.response.execution_time» — содержит информацию о времени, затраченном на выполнение запроса и получение ответа;
- «http.request.method» — содержит метод HTTP-запроса (GET, POST, PUT и т.д.);
- «url.path» — содержит путь в URL-адресе запроса;
- «error.code» — содержит код ошибки, если запрос завершается с ошибкой.

Формат задания настройки — `string`.

Значение по умолчанию — `default`.

12.13.5 Группа параметров LUNA_API_ADDRESS

Данная группа параметров задает настройки подключения к сервису API.

12.13.5.1 origin

Параметр задает протокол, IP адрес и порт сервиса API.

IP адрес «127.0.0.1» означает, что будет использоваться сервис API, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом API.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5000`.

12.13.5.2 api_version

Параметр задает версию API сервиса API. Доступная версия API — «6».

Формат задания настройки — `integer`.

Значение по умолчанию — `6`.

12.13.6 Группа параметров LUNA_API_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису API.

12.13.6.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису API. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.13.6.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.13.6.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.13.6.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.13.7 Группа параметров LUNA_IMAGE_STORE_PORTRAITS_ADDRESS

В данной группе параметров задаются настройки бакета для хранения портретов.

12.13.7.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.13.7.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.13.7.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `portraits`.

12.13.8 Группа параметров LUNA_IMAGE_STORE_PORTRAITS_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к бакету, хранящему портретов.

12.13.8.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к бакету, хранящему портретов. Это время ожидания, в течение которого клиент пытается установить соединение с бакетом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `30`.

12.13.8.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.13.9 Группа параметров BACKPORT3_EVENTS_DB_ADDRESS

12.13.9.1 user

Параметр задает имя пользователя БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.13.9.2 password

Параметр задает пароль БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию не указывается.

12.13.9.3 host

Параметр задает IP-адрес БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.13.9.4 port

Параметр задает номер порта, на котором Redis ожидает входящие сетевые соединения и прослушивает их для выполнения команд от клиентов.

Формат задания настройки — `integer`.

Значение по умолчанию — `6379`.

12.13.9.5 channel

Параметр задает канал Redis-канал, на который сервис подписывается сервис.

Формат задания настройки — `integer`.

Значение по умолчанию — `luna-backport3`.

12.13.9.6 sentinel > master_name

Параметр задает имя мастера базы данных Redis, который контролируется и управляется системой Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_backport3_master`.

12.13.9.7 sentinel > sentinels

Параметр задает список адресов и портов серверов Sentinel, которые будут использоваться клиентами для обнаружения и мониторинга БД Redis.

Формат задания настройки — `list > string`.

Значение по умолчанию — `[]`.

12.13.9.8 sentinel > user

Параметр задает имя пользователя сервера Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.13.9.9 sentinel > password

Параметр задает пароль пользователя сервера Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.13.10 Группа параметров LUNA_BACKPORT3_HTTP_SETTINGS

В данной группе параметров содержатся настройки, отвечающие за обработку HTTP-подключений. См. подробную информацию по следующей ссылке: https://sanic.dev/en/guide/deployment/configuration.html#builtin-values

12.13.10.1 request_timeout

Параметр задает продолжительность времени между моментом, когда новое открытое TCP-соединение передается на сервер, и моментом, когда получен весь HTTP-запрос.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.13.10.2 response_timeout

Параметр задает продолжительность времени между моментом, когда сервер передает HTTP-запрос приложению, и моментом, когда HTTP-ответ отправляется клиенту.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `600`.

12.13.10.3 request_max_size

Параметр задает максимальный размер запроса.

Формат задания настройки — `integer` (байты).

Значение по умолчанию — `1073741824`.

12.13.10.4 keep_alive_timeout

Параметр задает тайм-аут поддержания активности HTTP.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `15`.

12.13.11 Группа параметров LUNA_SERVICE_METRICS

Данная группа параметров включает и настраивает сбор метрик в формате Prometheus.

См. подробную информацию в разделе «Мониторинг».

12.13.11.1 enabled

Параметр включает сбор метрик.

Если сбор метрик отключен, то запрос к ресурсу `/metrics` вернет соответствующее сообщение.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.13.11.2 metrics_format

Параметр задает формат метрик.

На данный момент поддерживается только формат Prometheus.

См. официальную документацию Prometheus для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `prometheus`.

12.13.11.3 extra_labels

Параметр задает пользовательские типы лейблов.

Формат задания настройки — `label_name=label_value`.

Значение по умолчанию не задано.

12.13.12 Прочие

12.13.12.1 storage_time

Параметр задает формат времени, используемый для записей в базе данных. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.13.12.2 luna_backport3_active_plugins

Параметр задает список плагинов, которые должен использовать сервис.

Имена задаются в следующем формате:

```
[
   "plugin_1",
   "plugin_2",
   "plugin_3"
]
```

> Список должен содержать имена файлов без расширения (.py).

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.13.12.3 use_samples_as_portraits

Параметр позволяет использовать биометрические образцы вместо портретов для того, чтобы не хранить оба вида сущностей.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.13.12.4 backport3_enable_portraits

Параметр позволяет отключить возможность использования портретов, но оставить возможность использования остального функционала сервиса Image Store. Если использование сервиса Image Store отключено в настройке «ADDITIONAL_SERVICES_USAGE», то данная настройка также должна быть отключена.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.13.12.5 backport3_enable_ws_events

Параметр включает поддержку веб-сокетов для Backport3.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.13.12.6 max_candidate_in_response

Параметр задает максимальное количество кандидатов в ответах на запросы сравнения.

Формат задания настройки — `integer`.

Значение по умолчанию — `5`.

12.14 Настройки сервиса Backport 4

Данный раздел описывает параметры сервиса Backport 4.

Настройку сервиса можно выполнить с помощью сервиса Configurator.

12.14.1 Группа параметров LUNA_CONFIGURATOR

Данная группа параметров задаёт настройки подключения сервиса Backport 4 к сервису Configurator.

Данная группа параметров не будет видна в пользовательском интерфейсе сервиса Configurator. Параметры можно изменить только в конфигурационном файле «config.conf», расположенном в директории «/srv/luna_backport4/configs/» соответствующего контейнера.

12.14.1.1 use_configurator

Параметр позволяет включить использование сервиса Configurator.

> С помощью сервиса Configurator упрощается настройка сервисов LP. Сервис сохраняет все необходимые конфигурации для всех сервисов LP в одном месте.

При отключенном параметре будут использованы настройки из файла «config.conf», расположенного в директории «/srv/luna_backport4/configs/» соответствующего контейнера.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.14.1.2 luna_configurator_origin

Параметр задает протокол, IP адрес и порт сервиса Configurator.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5070`.

12.14.1.3 luna_configurator_api

Параметр задает версию API сервиса Configurator. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.14.2 Группа параметров LUNA_BACKPORT4_DB

В данной группе параметров задаются настройки подключения к базе данных сервиса Backport4.

12.14.2.1 db_type

Параметр задает тип базы данных. Доступны следующие типы:

- «postgres» — тип СУБД PostgreSQL
- «oracle» — тип СУБД Oracle

Формат задания настройки — `string`.

Значение по умолчанию — `postgres`.

12.14.2.2 db_host

Параметр задает имя сервера (хост) базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.14.2.3 db_port

Параметр задает порт базы данных.

Порт по умолчанию для типа «postgres»- `5432`.

Порт по умолчанию для типа «oracle» — `1521`.

Формат задания настройки — `string`.

Значение по умолчанию — `5432`.

12.14.2.4 db_user

Параметр задает имя пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.14.2.5 db_password

Параметр задает пароль пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.14.2.6 db_name

Параметр задает имя базы данных для типа «postgres» или имя SID для типа «oracle».

Формат задания настройки — `string`.

Значение по умолчанию — `luna_backport3`.

12.14.2.7 connection_pool_size

Параметр задает размер пула соединений к БД. Реальное количество подключений может быть больше значения данной настройки на 1.

При необходимости в настройке «max_connections» конфигурационного файла PostgreSQL можно задать максимальное количество одновременных подключений к серверу БД. См. раздел «Продвинутая настройка PostrgeSQL» для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `10`.

12.14.2.8 dsn

Параметр задает строку подключения DSN для соединения с БД.

> DSN — это строка подключения, которая идентифицирует и указывает на источник данных (базу данных), к которому нужно установить соединение.

В строке DSN могут задаваться такие настройки, как множественные хосты, аутентификационные данные, порт и другие параметры.

> Настройки зависят от типа БД. Множественные хосты поддерживаются только с PostgreSQL.

> По умолчанию параметр «dsn» не отображается во вкладке «Settings» в Configurator. Проверить список всех доступных параметров для группы настроек можно на вкладке «Limitations».

Пример:

```
{
"dsn": "luna:luna@postgres01:5432,postgres02:5432/luna_backport3?some_option
   =some_value"
"db_settings": {
    "connection_pool_size": 5
    }
}
```

Здесь:

- «luna:luna» — имя пользователя и пароль для подключения к PostgreSQL;
- «@postgres01:5432,postgres02:5432» — список хостов и портов, разделенных запятыми. Это означает, что сервис будет пытаться подключиться к первому хосту («postgres01») на порту 5432. Если это не удастся, она попытается подключиться ко второму хосту («postgres02») также на порту 5432;
- «/luna_backport3» — название базы данных;
- «?some_option=some_value» — опциональные параметры для подключения.

При необходимости можно комбинировать строку DSN и классические настройки, однако строка DSN является более приоритетной. Можно частично заполнить строку DSN (например, «postgres01,postgres02/luna_backport3»), и тогда недостающие параметры будут заполнены из значений параметров «db_host», «db_port», «db_name», «db_user» и «db_password».

При запуске сервис создаст пул подключений к одному из доступных хостов DSN. В случае возникновения проблем с установлением соединения после нескольких неудачных попыток, сервис снова попытается настроить пул подключений к любому из доступных хостов DSN.

12.14.3 Группа параметров LUNA_MONITORING

Данная группа параметров задает настройки мониторинга.

См. подробную информацию о мониторинге в разделе «Мониторинг».

12.14.3.1 storage_type

Тип хранилища для хранения данных мониторинга.

В настоящее время доступно только БД Influx.

Формат задания настройки — `string`.

Значение по умолчанию — `influx`.

12.14.3.2 send_data_for_monitoring

Параметр позволяет включить или отключить отправку данных для мониторинга в InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.14.3.3 use_ssl

Параметр позволяет использовать HTTPS для подключения к InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.14.3.4 organization

Параметр задает рабочую область InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.14.3.5 token

Параметр задает аутентификации InfluxDB 2.x.

Формат задания настройки — `string`.

12.14.3.6 bucket

Параметр задаёт имя бакета InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_monitoring`.

12.14.3.7 host

Параметр задаёт IP-адрес InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.14.3.8 port

Параметр задает порт InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `8086`.

12.14.3.9 flushing_period

Параметр задает частоту отправки данных мониторинга в InfluxDB.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `1`.

12.14.4 Группа параметров LUNA_BACKPORT4_LOGGER

Данная группа параметров задает настройки логирования.

12.14.4.1 log_level

Параметр задает уровень отладочной печати, по приоритету: «ERROR», «WARNING», «INFO», «DEBUG».

Формат задания настройки — `string`.

Значение по умолчанию — `INFO`.

12.14.4.2 log_time

Параметр задает формат времени, используемый в записях лога. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.14.4.3 log_to_stdout

Параметр позволяет отправлять логи в стандартный вывод (stdout).

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`

12.14.4.4 log_to_file

Параметр позволяет сохранять логи в файл. Директория с файлами логов указывается в параметре «folder_with_logs».

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.14.4.5 folder_with_logs

Параметр задает директорию, в которой хранятся логи. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить параметр «log_to_file».

Формат задания настройки — `string`.

Значение по умолчанию — `./`

Пример:

```
  "folder_with_logs": "/srv/logs"
```

12.14.4.6 max_log_file_size

Параметр задает максимальный размер файла лога в МБ перед выполнением его ротации (0 — не использовать ротацию).

Для использования данного параметра требуется включить параметр «log_to_file».

> При необходимости можно настроить ротацию логов Docker. См. раздел «Настройка ротации логов Docker» в руководстве по установке LUNA PLATFORM.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024`

12.14.4.7 multiline_stack_trace

Параметр включает многострочную трассировку стека в логах. Когда параметр включен, информация о стеке вызовов записывается в логах так, что каждый фрейм стека помещается на отдельной строке, что улучшает читаемость. Если параметр выключен, информация о стеке вызовов записывается на одной строке, что может сделать логи менее удобными для анализа.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.14.4.8 format

Параметр определяет формат выводимых логов. Доступны следующие значения:

- «default» — стандартный формат вывода логов LUNA PLATFORM
- «json» — вывод логов в формате json
- «ecs» — вывод логов в формате ECS (Elastic Common Schema)

При использовании значения «ecs» будут использоваться следующие поля:

- «http.response.status_code» — содержит код состояния ответа HTTP (200, 404, 500 и т.д.);
- «http.response.execution_time» — содержит информацию о времени, затраченном на выполнение запроса и получение ответа;
- «http.request.method» — содержит метод HTTP-запроса (GET, POST, PUT и т.д.);
- «url.path» — содержит путь в URL-адресе запроса;
- «error.code» — содержит код ошибки, если запрос завершается с ошибкой.

Формат задания настройки — `string`.

Значение по умолчанию — `default`.

12.14.5 Группа параметров LUNA_API_ADDRESS

Данная группа параметров задает настройки подключения к сервису API.

12.14.5.1 origin

Параметр задает протокол, IP адрес и порт сервиса API.

IP адрес «127.0.0.1» означает, что будет использоваться сервис API, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом API.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5000`.

12.14.5.2 api_version

Параметр задает версию API сервиса API. Доступная версия API — «6».

Формат задания настройки — `integer`.

Значение по умолчанию — `6`.

12.14.6 Группа параметров LUNA_API_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису API.

12.14.6.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису API. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.14.6.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.14.6.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.14.6.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.14.7 Группа параметров LUNA_FACES_ADDRESS

Данная группа параметров задает настройки подключения к сервису Faces.

12.14.7.1 origin

Параметр задает протокол, IP адрес и порт сервиса Faces.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Faces, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Faces.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5030`.

12.14.7.2 api_version

Параметр задает версию API сервиса Faces. Доступная версия API — «3».

Формат задания настройки — `integer`.

Значение по умолчанию — `3`.

12.14.8 Группа параметров LUNA_FACES_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Faces.

12.14.8.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Faces. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.14.8.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.14.8.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.14.8.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.14.9 Группа параметров ATTRIBUTES_STORAGE_POLICY

В данной группе параметров задаются настройки для сервиса Backport4, связанные с хранением временных атрибутов.

12.14.9.1 default_ttl

Параметр задает время существования временных атрибутов по умолчанию.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `300`.

12.14.9.2 max_ttl

Параметр задает максимальное время существования временных атрибутов.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `86400`.

12.14.10 Группа параметров LUNA_BACKPORT4_HTTP_SETTINGS

В данной группе параметров содержатся настройки, отвечающие за обработку HTTP-подключений. См. подробную информацию по следующей ссылке: https://sanic.dev/en/guide/deployment/configuration.html#builtin-values

12.14.10.1 request_timeout

Параметр задает продолжительность времени между моментом, когда новое открытое TCP-соединение передается на сервер, и моментом, когда получен весь HTTP-запрос.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.14.10.2 response_timeout

Параметр задает продолжительность времени между моментом, когда сервер передает HTTP-запрос приложению, и моментом, когда HTTP-ответ отправляется клиенту.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `600`.

12.14.10.3 request_max_size

Параметр задает максимальный размер запроса.

Формат задания настройки — `integer` (байты).

Значение по умолчанию — `1073741824`.

12.14.10.4 keep_alive_timeout

Параметр задает тайм-аут поддержания активности HTTP.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `15`.

12.14.11 Группа параметров LUNA_SERVICE_METRICS

Данная группа параметров включает и настраивает сбор метрик в формате Prometheus.

См. подробную информацию в разделе «Мониторинг».

12.14.11.1 enabled

Параметр включает сбор метрик.

Если сбор метрик отключен, то запрос к ресурсу `/metrics` вернет соответствующее сообщение.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.14.11.2 metrics_format

Параметр задает формат метрик.

На данный момент поддерживается только формат Prometheus.

См. официальную документацию Prometheus для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `prometheus`.

12.14.11.3 extra_labels

Параметр задает пользовательские типы лейблов.

Формат задания настройки — `label_name=label_value`.

Значение по умолчанию не задано.

12.14.12 Прочие

12.14.12.1 luna_backport4_active_plugins

Параметр задает список плагинов, которые должен использовать сервис.

Имена задаются в следующем формате:

```
[
   "plugin_1",
   "plugin_2",
   "plugin_3"
]
```

> Список должен содержать имена файлов без расширения (.py).

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.15 Настройки сервиса Accounts

Данный раздел описывает параметры сервиса Accounts.

Настройку сервиса можно выполнить с помощью сервиса Configurator.

12.15.1 Группа параметров LUNA_CONFIGURATOR

Данная группа параметров задаёт настройки подключения сервиса Accounts к сервису Configurator.

Данная группа параметров не будет видна в пользовательском интерфейсе сервиса Configurator. Параметры можно изменить только в конфигурационном файле «config.conf», расположенном в директории «/srv/luna_accounts/configs/» соответствующего контейнера.

12.15.1.1 use_configurator

Параметр позволяет включить использование сервиса Configurator.

> С помощью сервиса Configurator упрощается настройка сервисов LP. Сервис сохраняет все необходимые конфигурации для всех сервисов LP в одном месте.

При отключенном параметре будут использованы настройки из файла «config.conf», расположенного в директории «/srv/luna_accounts/configs/» соответствующего контейнера.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.15.1.2 luna_configurator_origin

Параметр задает протокол, IP адрес и порт сервиса Configurator.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5070`.

12.15.1.3 luna_configurator_api

Параметр задает версию API сервиса Configurator. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.15.2 Группа параметров LUNA_ACCOUNTS_DB

В данной группе параметров задаются настройки подключения к базе данных сервиса Accounts.

12.15.2.1 db_type

Параметр задает тип базы данных. Доступны следующие типы:

- «postgres» — тип СУБД PostgreSQL
- «oracle» — тип СУБД Oracle

Формат задания настройки — `string`.

Значение по умолчанию — `postgres`.

12.15.2.2 db_host

Параметр задает имя сервера (хост) базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.15.2.3 db_port

Параметр задает порт базы данных.

Порт по умолчанию для типа «postgres»- `5432`.

Порт по умолчанию для типа «oracle» — `1521`.

Формат задания настройки — `string`.

Значение по умолчанию — `5432`.

12.15.2.4 db_user

Параметр задает имя пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.15.2.5 db_password

Параметр задает пароль пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.15.2.6 db_name

Параметр задает имя базы данных для типа «postgres» или имя SID для типа «oracle».

Формат задания настройки — `string`.

Значение по умолчанию — `luna_accounts`.

12.15.2.7 connection_pool_size

Параметр задает размер пула соединений к БД. Реальное количество подключений может быть больше значения данной настройки на 1.

При необходимости в настройке «max_connections» конфигурационного файла PostgreSQL можно задать максимальное количество одновременных подключений к серверу БД. См. раздел «Продвинутая настройка PostrgeSQL» для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `10`.

12.15.2.8 dsn

Параметр задает строку подключения DSN для соединения с БД.

> DSN — это строка подключения, которая идентифицирует и указывает на источник данных (базу данных), к которому нужно установить соединение.

В строке DSN могут задаваться такие настройки, как множественные хосты, аутентификационные данные, порт и другие параметры.

> Настройки зависят от типа БД. Множественные хосты поддерживаются только с PostgreSQL.

> По умолчанию параметр «dsn» не отображается во вкладке «Settings» в Configurator. Проверить список всех доступных параметров для группы настроек можно на вкладке «Limitations».

Пример:

```
{
"dsn": "luna:luna@postgres01:5432,postgres02:5432/luna_accounts?some_option=
    some_value"
"db_settings": {
     "connection_pool_size": 5
     }
}
```

Здесь:

- «luna:luna» — имя пользователя и пароль для подключения к PostgreSQL;
- «@postgres01:5432,postgres02:5432» — список хостов и портов, разделенных запятыми. Это означает, что сервис будет пытаться подключиться к первому хосту («postgres01») на порту 5432. Если это не удастся, она попытается подключиться ко второму хосту («postgres02») также на порту 5432;
- «/luna_accounts» — название базы данных;
- «?some_option=some_value» — опциональные параметры для подключения.

При необходимости можно комбинировать строку DSN и классические настройки, однако строка DSN является более приоритетной. Можно частично заполнить строку DSN (например, «postgres01,postgres02/luna_accounts»), и тогда недостающие параметры будут заполнены из значений параметров «db_host», «db_port», «db_name», «db_user» и «db_password».

При запуске сервис создаст пул подключений к одному из доступных хостов DSN. В случае возникновения проблем с установлением соединения после нескольких неудачных попыток, сервис снова попытается настроить пул подключений к любому из доступных хостов DSN.

12.15.3 Группа параметров LUNA_MONITORING

Данная группа параметров задает настройки мониторинга.

См. подробную информацию о мониторинге в разделе «Мониторинг».

12.15.3.1 storage_type

Тип хранилища для хранения данных мониторинга.

В настоящее время доступно только БД Influx.

Формат задания настройки — `string`.

Значение по умолчанию — `influx`.

12.15.3.2 send_data_for_monitoring

Параметр позволяет включить или отключить отправку данных для мониторинга в InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.15.3.3 use_ssl

Параметр позволяет использовать HTTPS для подключения к InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.15.3.4 organization

Параметр задает рабочую область InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.15.3.5 token

Параметр задает аутентификации InfluxDB 2.x.

Формат задания настройки — `string`.

12.15.3.6 bucket

Параметр задаёт имя бакета InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_monitoring`.

12.15.3.7 host

Параметр задаёт IP-адрес InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.15.3.8 port

Параметр задает порт InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `8086`.

12.15.3.9 flushing_period

Параметр задает частоту отправки данных мониторинга в InfluxDB.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `1`.

12.15.4 Группа параметров LUNA_ACCOUNTS_LOGGER

Данная группа параметров задает настройки логирования.

12.15.4.1 log_level

Параметр задает уровень отладочной печати, по приоритету: «ERROR», «WARNING», «INFO», «DEBUG».

Формат задания настройки — `string`.

Значение по умолчанию — `INFO`.

12.15.4.2 log_time

Параметр задает формат времени, используемый в записях лога. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.15.4.3 log_to_stdout

Параметр позволяет отправлять логи в стандартный вывод (stdout).

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`

12.15.4.4 log_to_file

Параметр позволяет сохранять логи в файл. Директория с файлами логов указывается в параметре «folder_with_logs».

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.15.4.5 folder_with_logs

Параметр задает директорию, в которой хранятся логи. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить параметр «log_to_file».

Формат задания настройки — `string`.

Значение по умолчанию — `./`

Пример:

```
  "folder_with_logs": "/srv/logs"
```

12.15.4.6 max_log_file_size

Параметр задает максимальный размер файла лога в МБ перед выполнением его ротации (0 — не использовать ротацию).

Для использования данного параметра требуется включить параметр «log_to_file».

> При необходимости можно настроить ротацию логов Docker. См. раздел «Настройка ротации логов Docker» в руководстве по установке LUNA PLATFORM.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024`

12.15.4.7 multiline_stack_trace

Параметр включает многострочную трассировку стека в логах. Когда параметр включен, информация о стеке вызовов записывается в логах так, что каждый фрейм стека помещается на отдельной строке, что улучшает читаемость. Если параметр выключен, информация о стеке вызовов записывается на одной строке, что может сделать логи менее удобными для анализа.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.15.4.8 format

Параметр определяет формат выводимых логов. Доступны следующие значения:

- «default» — стандартный формат вывода логов LUNA PLATFORM
- «json» — вывод логов в формате json
- «ecs» — вывод логов в формате ECS (Elastic Common Schema)

При использовании значения «ecs» будут использоваться следующие поля:

- «http.response.status_code» — содержит код состояния ответа HTTP (200, 404, 500 и т.д.);
- «http.response.execution_time» — содержит информацию о времени, затраченном на выполнение запроса и получение ответа;
- «http.request.method» — содержит метод HTTP-запроса (GET, POST, PUT и т.д.);
- «url.path» — содержит путь в URL-адресе запроса;
- «error.code» — содержит код ошибки, если запрос завершается с ошибкой.

Формат задания настройки — `string`.

Значение по умолчанию — `default`.

12.15.5 Группа параметров LUNA_ACCOUNTS_HTTP_SETTINGS

В данной группе параметров содержатся настройки, отвечающие за обработку HTTP-подключений. См. подробную информацию по следующей ссылке: https://sanic.dev/en/guide/deployment/configuration.html#builtin-values

12.15.5.1 request_timeout

Параметр задает продолжительность времени между моментом, когда новое открытое TCP-соединение передается на сервер, и моментом, когда получен весь HTTP-запрос.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.15.5.2 response_timeout

Параметр задает продолжительность времени между моментом, когда сервер передает HTTP-запрос приложению, и моментом, когда HTTP-ответ отправляется клиенту.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `600`.

12.15.5.3 request_max_size

Параметр задает максимальный размер запроса.

Формат задания настройки — `integer` (байты).

Значение по умолчанию — `1073741824`.

12.15.5.4 keep_alive_timeout

Параметр задает тайм-аут поддержания активности HTTP.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `15`.

12.15.6 Группа параметров LUNA_SERVICE_METRICS

Данная группа параметров включает и настраивает сбор метрик в формате Prometheus.

См. подробную информацию в разделе «Мониторинг».

12.15.6.1 enabled

Параметр включает сбор метрик.

Если сбор метрик отключен, то запрос к ресурсу `/metrics` вернет соответствующее сообщение.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.15.6.2 metrics_format

Параметр задает формат метрик.

На данный момент поддерживается только формат Prometheus.

См. официальную документацию Prometheus для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `prometheus`.

12.15.6.3 extra_labels

Параметр задает пользовательские типы лейблов.

Формат задания настройки — `label_name=label_value`.

Значение по умолчанию не задано.

12.15.7 Прочие

12.15.7.1 luna_accounts_active_plugins

Параметр задает список плагинов, которые должен использовать сервис.

Имена задаются в следующем формате:

```
[
   "plugin_1",
   "plugin_2",
   "plugin_3"
]
```

> Список должен содержать имена файлов без расширения (.py).

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.15.7.2 storage_time

Параметр задает формат времени, используемый для записей в базе данных. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.16 Настройки сервиса Remote SDK

Данный раздел описывает параметры сервиса Remote SDK.

Настройку сервиса можно выполнить с помощью сервиса Configurator.

12.16.1 Группа параметров LUNA_CONFIGURATOR

Данная группа параметров задаёт настройки подключения сервиса Remote SDK к сервису Configurator.

Данная группа параметров не будет видна в пользовательском интерфейсе сервиса Configurator. Параметры можно изменить только в конфигурационном файле «config.conf», расположенном в директории «/srv/luna_remote_sdk/configs/» соответствующего контейнера.

12.16.1.1 use_configurator

Параметр позволяет включить использование сервиса Configurator.

> С помощью сервиса Configurator упрощается настройка сервисов LP. Сервис сохраняет все необходимые конфигурации для всех сервисов LP в одном месте.

При отключенном параметре будут использованы настройки из файла «config.conf», расположенного в директории «/srv/luna_remote_sdk/configs/» соответствующего контейнера.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.16.1.2 luna_configurator_origin

Параметр задает протокол, IP адрес и порт сервиса Configurator.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5070`.

12.16.1.3 luna_configurator_api

Параметр задает версию API сервиса Configurator. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.2 Группа параметров LUNA_MONITORING

Данная группа параметров задает настройки мониторинга.

См. подробную информацию о мониторинге в разделе «Мониторинг».

12.16.2.1 storage_type

Тип хранилища для хранения данных мониторинга.

В настоящее время доступно только БД Influx.

Формат задания настройки — `string`.

Значение по умолчанию — `influx`.

12.16.2.2 send_data_for_monitoring

Параметр позволяет включить или отключить отправку данных для мониторинга в InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.2.3 use_ssl

Параметр позволяет использовать HTTPS для подключения к InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.16.2.4 organization

Параметр задает рабочую область InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.16.2.5 token

Параметр задает аутентификации InfluxDB 2.x.

Формат задания настройки — `string`.

12.16.2.6 bucket

Параметр задаёт имя бакета InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_monitoring`.

12.16.2.7 host

Параметр задаёт IP-адрес InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.16.2.8 port

Параметр задает порт InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `8086`.

12.16.2.9 flushing_period

Параметр задает частоту отправки данных мониторинга в InfluxDB.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `1`.

12.16.3 Группа параметров LUNA_REMOTE_SDK_LOGGER

Данная группа параметров задает настройки логирования.

12.16.3.1 log_level

Параметр задает уровень отладочной печати, по приоритету: «ERROR», «WARNING», «INFO», «DEBUG».

Формат задания настройки — `string`.

Значение по умолчанию — `INFO`.

12.16.3.2 log_time

Параметр задает формат времени, используемый в записях лога. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.16.3.3 log_to_stdout

Параметр позволяет отправлять логи в стандартный вывод (stdout).

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`

12.16.3.4 log_to_file

Параметр позволяет сохранять логи в файл. Директория с файлами логов указывается в параметре «folder_with_logs».

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.16.3.5 folder_with_logs

Параметр задает директорию, в которой хранятся логи. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить параметр «log_to_file».

Формат задания настройки — `string`.

Значение по умолчанию — `./`

Пример:

```
  "folder_with_logs": "/srv/logs"
```

12.16.3.6 max_log_file_size

Параметр задает максимальный размер файла лога в МБ перед выполнением его ротации (0 — не использовать ротацию).

Для использования данного параметра требуется включить параметр «log_to_file».

> При необходимости можно настроить ротацию логов Docker. См. раздел «Настройка ротации логов Docker» в руководстве по установке LUNA PLATFORM.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024`

12.16.3.7 multiline_stack_trace

Параметр включает многострочную трассировку стека в логах. Когда параметр включен, информация о стеке вызовов записывается в логах так, что каждый фрейм стека помещается на отдельной строке, что улучшает читаемость. Если параметр выключен, информация о стеке вызовов записывается на одной строке, что может сделать логи менее удобными для анализа.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.16.3.8 format

Параметр определяет формат выводимых логов. Доступны следующие значения:

- «default» — стандартный формат вывода логов LUNA PLATFORM
- «json» — вывод логов в формате json
- «ecs» — вывод логов в формате ECS (Elastic Common Schema)

При использовании значения «ecs» будут использоваться следующие поля:

- «http.response.status_code» — содержит код состояния ответа HTTP (200, 404, 500 и т.д.);
- «http.response.execution_time» — содержит информацию о времени, затраченном на выполнение запроса и получение ответа;
- «http.request.method» — содержит метод HTTP-запроса (GET, POST, PUT и т.д.);
- «url.path» — содержит путь в URL-адресе запроса;
- «error.code» — содержит код ошибки, если запрос завершается с ошибкой.

Формат задания настройки — `string`.

Значение по умолчанию — `default`.

12.16.4 Группа параметров LUNA_LICENSES_ADDRESS

Данная группа параметров задает настройки подключения к сервису Licenses.

12.16.4.1 origin

Параметр задает протокол, IP адрес и порт сервиса Licenses.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Licenses, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Licenses.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5120`.

12.16.4.2 api_version

Параметр задает версию API сервиса Licenses. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.5 Группа параметров LUNA_IMAGE_STORE_FACES_SAMPLES_ADDRESS

В данной группе параметров задаются настройки бакета для хранения БО лиц.

12.16.5.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.16.5.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.5.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-samples`.

12.16.6 Группа параметров LUNA_IMAGE_STORE_FACES_SAMPLES_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к бакету, хранящему БО лиц.

12.16.6.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к бакету, хранящему БО лиц. Это время ожидания, в течение которого клиент пытается установить соединение с бакетом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `30`.

12.16.6.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.16.7 Группа параметров LUNA_IMAGE_STORE_BODIES_SAMPLES_ADDRESS

В данной группе параметров задаются настройки бакета для хранения БО тел.

12.16.7.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.16.7.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.7.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-bodies-samples`.

12.16.8 Группа параметров LUNA_IMAGE_STORE_BODIES_SAMPLES_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к бакету, хранящему БО тел.

12.16.8.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к бакету, хранящему БО тел. Это время ожидания, в течение которого клиент пытается установить соединение с бакетом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `30`.

12.16.8.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.16.9 Группа параметров LUNA_IMAGE_STORE_IMAGES_ADDRESS

В данной группе параметров задаются настройки бакета для хранения исходных изображений.

12.16.9.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.16.9.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.9.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-image-origin`.

12.16.10 Группа параметров LUNA_IMAGE_STORE_IMAGES_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к бакету, хранящему исходных изображений.

12.16.10.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к бакету, хранящему исходных изображений. Это время ожидания, в течение которого клиент пытается установить соединение с бакетом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `30`.

12.16.10.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.16.11 Группа параметров LUNA_IMAGE_STORE_OBJECTS_ADDRESS

В данной группе параметров задаются настройки бакета для хранения объектов.

12.16.11.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.16.11.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.11.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-objects`.

12.16.12 Группа параметров ADDITIONAL_SERVICES_USAGE

12.16.12.1 luna_events

Параметр задает возможность использования сервиса Events.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.16.12.2 luna_handlers

Параметр задает возможность использования сервиса Handlers.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.16.12.3 luna_faces

Параметр задает возможность использования сервиса Faces.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.16.12.4 luna_sender

Параметр задает возможность использования сервиса Sender.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.16.12.5 luna_matcher_proxy

Параметр задает возможность использования сервиса Python Matcher Proxy.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.16.12.6 luna_image_store

Параметр задает возможность использования сервиса Image Store.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.16.12.7 luna_lambda

Параметр задает возможность использования сервиса Lambda.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.16.12.8 luna_video_manager

Параметр задает возможность использования сервиса Video Manager.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.16.12.9 luna_video_agent

Параметр задает возможность использования сервиса Video Agent.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.16.12.10 luna_streams_retranslator

Параметр задает возможность использования сервиса Streams Retranslator.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.16.13 Группа параметров LUNA_REMOTE_SDK_RUNTIME_SETTINGS

Данная группа параметров задает глобальные настройки для всех эстиматоров/детекторов.

12.16.13.1 global_device_class

Параметр задает тип устройства («cpu» или «gpu») для всех эстиматоров/детекторов, у которых значение параметра «device_class» = «global».

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.13.2 num_threads

Параметр задает количество рабочих потоков CPU-процессора, которые будут использоваться сервисом при выполнении эстимации/детекции. Желательно, чтобы значение этого параметра соответствовало числу физических ядер процессора для достижения максимальной производительности. Однако, слишком большое число потоков может привести к ухудшению производительности из-за дополнительных накладных расходов.

Формат задания настройки — `integer`.

Значение по умолчанию — `4`.

12.16.13.3 num_compute_streams

Параметр задает количество потоков вычислений на GPU-процессоре, которые будут использоваться сервисом при выполнении эстимации/детекции. Рекомендуется выбирать значение, соответствующее характеристикам конкретного GPU и задаче. Стоит учитывать, что NVIDIA может изменять поведение этого параметра в различных версиях программного обеспечения, поэтому рекомендуется проводить тестирование для определения оптимального значения.

Формат задания настройки — `integer`.

Значение по умолчанию — `6`.

12.16.14 Группа параметров LUNA_REMOTE_SDK_FACE_DETECTOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы детектора лица.

12.16.14.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.14.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.14.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.14.4 estimator_settings > min_face_size

Параметр задает минимальный размер лица в пикселях.

Максимальный размер лица равен минимальному размеру лица, умноженному на 32.

Формат задания настройки — `integer`.

Значение по умолчанию — `50`.

12.16.14.5 estimator_settings > redetect_face_target_size

Параметр задает целевой размер лица для повторной детекции.

Формат задания настройки — `integer`.

Значение по умолчанию — `80`.

12.16.14.6 estimator_settings > redetect_tensor_size

Параметр задает целевой размер лица для повторной детекции после предварительной обработки.

Детекция считается неудачной, если оценка ниже указанного значения.

Формат задания настройки — `integer`.

Значение по умолчанию — `64`.

12.16.14.7 estimator_settings > redetect_score_threshold

Параметр задает порог оценки повторной детекции.

Повторная детекция считается неудачной, если оценка ниже указанного значения.

Формат задания настройки — `number`.

Значение по умолчанию — `0.3`.

12.16.14.8 estimator_settings > score_threshold

Параметр задает порог оценки.

Детекция считается неудачной, если оценка ниже указанного значения.

Формат задания настройки — `number`.

Значение по умолчанию — `0.5`.

12.16.15 Группа параметров LUNA_REMOTE_SDK_GAZE_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора направления взгляда.

12.16.15.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.15.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.15.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.16 Группа параметров LUNA_REMOTE_SDK_QUALITY_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора качества изображения.

12.16.16.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.16.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.16.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.17 Группа параметров LUNA_REMOTE_SDK_MOUTH_ATTRIBUTES_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора атрибутов рта.

12.16.17.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.17.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.17.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.18 Группа параметров LUNA_REMOTE_SDK_EMOTIONS_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора эмоций.

12.16.18.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.18.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.18.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.19 Группа параметров LUNA_REMOTE_SDK_BASIC_ATTRIBUTES_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора базовых атрибутов.

12.16.19.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.19.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.19.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.20 Группа параметров LUNA_REMOTE_SDK_EYES_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора атрибутов глаз.

12.16.20.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.20.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.20.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.21 Группа параметров LUNA_REMOTE_SDK_HEAD_POSE_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора положения головы.

12.16.21.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.21.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.21.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.22 Группа параметров LUNA_REMOTE_SDK_FACE_DESCRIPTOR_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора извлечения биометрического шаблона лица.

12.16.22.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.22.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.22.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.23 Группа параметров LUNA_REMOTE_SDK_MASK_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора маски.

12.16.23.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.23.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.23.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.24 Группа параметров LUNA_REMOTE_SDK_LIVENESS_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора Liveness.

12.16.24.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.24.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.24.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.24.4 estimator_settings > real_threshold

Параметр задает порог «Liveness threshold», ниже которого система будет считать результат атакой на биометрическое предъявление и будет выдан результат «spoof».

Формат задания настройки — `float`.

Значение по умолчанию — `0.5`.

12.16.25 Группа параметров LUNA_REMOTE_SDK_GLASSES_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора очков.

12.16.25.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.25.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.25.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.26 Группа параметров LUNA_REMOTE_SDK_FACE_WARP_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора биометрического образца лица.

12.16.26.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.26.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.26.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.27 Группа параметров LUNA_REMOTE_SDK_FACE_LANDMARKS68_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора 68 контрольных точек лица.

12.16.27.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.27.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.27.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.28 Группа параметров LUNA_REMOTE_SDK_FACE_LANDMARKS5_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора 5 контрольных точек лица.

12.16.28.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.28.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.28.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.29 Группа параметров LUNA_REMOTE_SDK_IMAGE_COLOR_TYPE_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора типа цвета изображения на основе лица.

12.16.29.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.29.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.29.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.30 Группа параметров LUNA_REMOTE_SDK_HEADWEAR_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора головного убора.

12.16.30.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.30.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.30.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.31 Группа параметров LUNA_REMOTE_SDK_FACE_NATURAL_LIGHT_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора естественности освещения.

12.16.31.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.31.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.31.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.32 Группа параметров LUNA_REMOTE_SDK_FISHEYE_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора бочкообразной дисторсии (FishEye).

12.16.32.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.32.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.32.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.33 Группа параметров LUNA_REMOTE_SDK_EYEBROW_EXPRESSION_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора бровей.

12.16.33.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.33.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.33.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.34 Группа параметров LUNA_REMOTE_SDK_RED_EYES_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора наличия эффекта красных глаз.

12.16.34.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.34.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.34.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.35 Группа параметров LUNA_REMOTE_SDK_FACE_DETECTION_BACKGROUND_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора фона изображения.

12.16.35.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.35.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.35.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.36 Группа параметров LUNA_REMOTE_SDK_IMAGE_ORIENTATION_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора ориентации изображения.

12.16.36.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.36.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.36.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.37 Группа параметров LUNA_REMOTE_SDK_PORTRAIT_STYLE_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора положения плеч.

12.16.37.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.37.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.37.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.38 Группа параметров LUNA_REMOTE_SDK_BODY_DETECTOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы детектора тела.

12.16.38.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.38.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.38.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.38.4 estimator_settings > image_size

Параметр задает максимальный размер кадра в пикселях после масштабирования по наибольшей из сторон кадра.

Формат задания настройки — `integer`.

Значение по умолчанию — `640`.

12.16.38.5 estimator_settings > redetect_score_threshold

Параметр задает порог оценки повторной детекции.

Повторная детекция считается неудачной, если оценка ниже указанного значения.

Формат задания настройки — `number`.

Значение по умолчанию — `0.12`.

12.16.38.6 estimator_settings > score_threshold

Параметр задает порог оценки.

Детекция считается неудачной, если оценка ниже указанного значения.

Формат задания настройки — `number`.

Значение по умолчанию — `0.5`.

12.16.39 Группа параметров LUNA_REMOTE_SDK_BODY_DESCRIPTOR_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора извлечения биометрического шаблона тела.

12.16.39.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.39.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.39.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.40 Группа параметров LUNA_REMOTE_SDK_BODY_WARP_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора биометрического образца тела.

12.16.40.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.40.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.40.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.41 Группа параметров LUNA_REMOTE_SDK_BODY_LANDMARKS_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора контрольных точек тела.

12.16.41.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.41.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.41.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.42 Группа параметров LUNA_REMOTE_SDK_BODY_ATTRIBUTES_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора параметров тел.

12.16.42.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.42.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.42.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.43 Группа параметров LUNA_REMOTE_SDK_HUMAN_DETECTOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы одновременного детектора лица и тела.

12.16.43.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.43.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.43.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.44 Группа параметров LUNA_REMOTE_SDK_PEOPLE_COUNT_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора количества людей на изображении.

12.16.44.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.44.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.44.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.45 Группа параметров LUNA_REMOTE_SDK_DEEPFAKE_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора Deepfake.

12.16.45.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.16.45.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.45.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.46 LUNA_REMOTE_SDK_FACE_OCCLUSION_ESTIMATOR_SETTINGS section

Данная группа параметров задает настройки эстиматора перекрытия лица.

12.16.46.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `global`.

12.16.46.2 runtime_settings > optimal_batch_size

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.16.46.3 runtime_settings > worker_count

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.46.4 estimator_settings > occlusion_threshold

Параметр задает допустимое пороговое значение для общего перекрытия лица.

Формат задания настройки — `integer`.

Значение по умолчанию — `0.07`.

12.16.46.5 estimator_settings > hair_threshold

Параметр задает допустимое пороговое значение для перекрытия волос.

Формат задания настройки — `integer`.

Значение по умолчанию — `0.15`.

12.16.46.6 estimator_settings > forehead_threshold

Параметр задает допустимое пороговое значение для перекрытия лба.

Формат задания настройки — `integer`.

Значение по умолчанию — `0.2`.

12.16.46.7 estimator_settings > eye_threshold

Параметр задает допустимое пороговое значение для перекрытия глаз.

Формат задания настройки — `integer`.

Значение по умолчанию — `0.15`.

12.16.46.8 estimator_settings > nose_threshold

Параметр задает допустимое пороговое значение для перекрытия носа.

Формат задания настройки — `integer`.

Значение по умолчанию — `0.2`.

12.16.46.9 estimator_settings > mouth_threshold

Параметр задает допустимое пороговое значение для перекрытия рта.

Формат задания настройки — `integer`.

Значение по умолчанию — `0.15`.

12.16.46.10 estimator_settings > lower_face_threshold

Параметр задает допустимое пороговое значение для перекрытия нижней части лица.

Формат задания настройки — `integer`.

Значение по умолчанию — `0.2`.

Группа параметров FETCH_EXTERNAL_IMAGE_TIMEOUTS {#remote_sdk_fetch_external_image_-timeouts}

Данная группа параметров представляет собой набор параметров для установки таймаутов в процессе выполнения HTTP-запросов к внешним ресурсам для загрузки изображений. Каждый из параметров в этой настройке определяет максимальное время ожидания для определенной операции в процессе выполнения запроса.

12.16.46.11 connect

Параметр задает таймаут для установки соединения с внешним ресурсом. Если соединение не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.16.46.12 request

Параметр задает таймаут для ожидания ответа на HTTP-запрос, который отправляется для получения изображения с внешнего ресурса.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.16.46.13 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.16.46.14 sock_request

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.16.47 Группа параметров EXTERNAL_LUNA_API_ADDRESS

Данная группа параметров предназначена для корректной обработки ссылок на объекты, созданные с помощью ресурсов «/images» и «/objects» в сервисе API. В данной секции задается адрес и версия API сервиса API.

Если в качестве входных данных для ресурсов «/iso» и «/sdk» указывается URL-адрес и версия сервиса API объекта типа «images», совпадающие с адресом и версией API из секции «EXTERNAL_LUNA_API_ADDRESS» сервиса Remote SDK, то данные объекты будут загружаться с помощью сервиса Image Store напрямую, а не отправлять запрос в сервис API с последующим перенаправлением в сервис Image Store.

Пример формата: `http://10.15.3.144:5000/6/images/141d2706-8baf-433b-82eb-8c7fada847da`, где значение `http://10.15.3.144:5000` должно совпадать со значением из настройки «origin», а значение `6` должно совпадать со значением настройки `api_version` секции «EXTERNAL_LUNA_API_ADDRESS».

Для избежания ошибок необходимо настроить данную секцию в настройках Remote SDK перед использованием URL-адресов до объектов типа «objects» или «images» в качестве источника входных данных.

12.16.47.1 origin

Параметр задает протокол, IP адрес и порт сервиса API.

См. описание логики работы в разделе «Группа параметров EXTERNAL_LUNA_API_ADDRESS».

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5000`.

12.16.47.2 api_version

Параметр задает версию API сервиса API.

См. описание логики работы в разделе «Группа параметров EXTERNAL_LUNA_API_ADDRESS».

Формат задания настройки — `integer`.

Значение по умолчанию — `6`.

12.16.48 Группа параметров LUNA_REMOTE_SDK_LIMITS

Данная группа параметров определяет ограничения для некоторых операций, выполняемых с помощью сервиса Remote SDK.

Увеличение данных ограничений может привести к уменьшению производительности.

12.16.48.1 received_images_limit

Параметр задает максимальное количество изображений, которые можно указать в запросах «iso», «sdk» с помощью схемы «multipart/form-data».

> Для пакетной обработки изображений необходимо использовать задачу Estimator.

Формат задания настройки — `integer`.

Значение по умолчанию — `8`.

12.16.49 Группа параметров LUNA_SERVICE_METRICS

Данная группа параметров включает и настраивает сбор метрик в формате Prometheus.

См. подробную информацию в разделе «Мониторинг».

12.16.49.1 enabled

Параметр включает сбор метрик.

Если сбор метрик отключен, то запрос к ресурсу `/metrics` вернет соответствующее сообщение.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.16.49.2 metrics_format

Параметр задает формат метрик.

На данный момент поддерживается только формат Prometheus.

См. официальную документацию Prometheus для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `prometheus`.

12.16.49.3 extra_labels

Параметр задает пользовательские типы лейблов.

Формат задания настройки — `label_name=label_value`.

Значение по умолчанию не задано.

12.16.50 Группа параметров FACE_QUALITY_SETTINGS

В данной группе параметров задаются пороги, при которых LUNA PLATFORM будет считать, что проверка пройдена.

В данной группе параметров есть два типа стандартных порогов:

- пороги из поля «threshold», определяющие при каком значении LUNA PLATFORM будет считать, что проверка пройдена. Данные пороги аналогичны порогам в группе проверок «face_quality» в соответствующих запросах. Задание порогов в запросах переопределяет стандартные пороги, задаваемые в группе параметров «FACE_QUALITY_DEFAULT_THRESHOLDS».
- пороги из поля «config», определяющие как интерпретировать ответ эстиматоров Natural Light, Fish Eye, Color Type и Red Eyes. Ранее данные пороги были недоступны.

Пример настройки порогов для эстиматора Red Eyes:

```
  "red_eye": {
      "threshold": 0,
      "config": {
          "estimator": {
              "threshold": {
                  "min": 0.5
  }     }     }     }
```

Здесь:

- поле «threshold» определяет значение «0» или «1», при котором LUNA PLATFORM будет считать, что проверка пройдена. Аналогично параметру «threshold» в запросах на выполнение «face_quality»
- секция «config» > «estimator» > «threshold» определяет ответ эстиматора Red Eyes от «0» до «1».

См. подробную информацию в разделе «Проверка изображений».

Важно! Данные пороги являются стандартными. Задание порогов в группе параметров «face_quality» соответствующего запроса переопределяет значение данных порогов.

12.16.50.1 image_format

Параметр задает стандартные пороги для проверки формата изображения.

Формат задания настройки — `array`.

Значение по умолчанию:

```
  "threshold": [
      "JPEG",
      "JPEG2000",
      "PNG"
  ]
```

12.16.50.2 illumination

Параметр задает стандартные пороги для проверки степени равномерности освещения.

Формат задания настройки — `object`.

Значение по умолчанию:

```
  "threshold": {
      "min": 0.3,
      "max": 1
  }
```

12.16.50.3 specularity

Параметр задает стандартные пороги для проверки степени отсутствия бликов.

Формат задания настройки — `object`.

Значение по умолчанию:

```
  "threshold": {
      "min": 0.3,
      "max": 1
  }
```

12.16.50.4 blurriness

Параметр задает стандартные пороги для проверки степени размытости.

Формат задания настройки — `object`.

Значение по умолчанию:

```
  "threshold": {
      "min": 0.61,
      "max": 1
  }
```

12.16.50.5 dark

Параметр задает стандартные пороги для проверки степени того, что фото не затемнено.

Формат задания настройки — `object`.

Значение по умолчанию:

```
  "threshold": {
      "min": 0.5,
      "max": 1
  }
```

12.16.50.6 light

Параметр задает стандартные пороги для проверки степени того, что фото не засвечено.

Формат задания настройки — `object`.

Значение по умолчанию:

```
  "threshold": {
      "min": 0.57,
      "max": 1
  }
```

12.16.50.7 head_yaw

Параметр задает стандартные пороги для проверки угла поворота головы вправо/влево.

Формат задания настройки — `object`.

Значение по умолчанию:

```
  "threshold": {
      "min": -5,
      "max": 5
  }
```

12.16.50.8 head_pitch

Параметр задает стандартные пороги для проверки угла наклона головы вверх/вниз.

Формат задания настройки — `object`.

Значение по умолчанию:

```
  "threshold": {
      "min": -5,
      "max": 5
  }
```

12.16.50.9 head_roll

Параметр задает стандартные пороги для проверки угла отклонения головы вправо/влево.

Формат задания настройки — `object`.

Значение по умолчанию:

```
"threshold": {
    "min": -8,
    "max": 8
}
```

12.16.50.10 gaze_yaw

Параметр задает стандартные пороги для проверки угла поворота взгляда вправо/влево.

Формат задания настройки — `object`.

Значение по умолчанию:

```
"threshold": {
    "min": -5,
    "max": 5
}
```

12.16.50.11 gaze_pitch

Параметр задает стандартные пороги для проверки угла наклона взгляда вверх/вниз.

Формат задания настройки — `object`.

Значение по умолчанию:

```
"threshold": {
    "min": -5,
    "max": 5
}
```

12.16.50.12 mouth_smiling

Параметр задает стандартные пороги для проверки степени наличия улыбки.

Формат задания настройки — `object`.

Значение по умолчанию:

```
"threshold": {
    "min": 0,
    "max": 0.5
}
```

12.16.50.13 mouth_occluded

Параметр задает стандартные пороги для проверки степени наличия перекрытого рта.

Формат задания настройки — `object`.

Значение по умолчанию:

```
  "threshold": {
      "min": 0,
      "max": 0.5
  }
```

12.16.50.14 mouth_open

Параметр задает стандартные пороги для проверки степени наличия открытого рта.

Формат задания настройки — `object`.

Значение по умолчанию:

```
  "threshold": {
      "min": 0,
      "max": 0.5
  }
```

12.16.50.15 glasses

Параметр задает стандартные пороги для проверки наиболее вероятного состояния очков.

Формат задания настройки — `array`.

Значение по умолчанию:

```
  "threshold": [
      "no_glasses",
      "eyeglasses"
  ]
```

12.16.50.16 eyes

Параметр задает стандартные пороги для проверки положения каждого глаза.

Формат задания настройки — `array`.

Значение по умолчанию:

```
  "threshold": [
      "open"
  ]
```

12.16.50.17 head_horizontal_center

Параметр задает стандартные пороги для проверки положения лица по горизонтали.

Формат задания настройки — `object`.

Значение по умолчанию:

```
  "threshold": {
      "min": 0.45,
      "max": 0.55
  }
```

12.16.50.18 head_vertical_center

Параметр задает стандартные пороги для проверки положения лица по вертикали.

Формат задания настройки — `object`.

Значение по умолчанию:

```
  "threshold": {
      "min": 0.3,
      "max": 0.5
  }
```

12.16.50.19 head_width

Параметр задает стандартные пороги для проверки ширины головы.

Формат задания настройки — `object`.

Значение по умолчанию:

```
  "threshold": {
      "min": 0.5,
      "max": 0.75
  }
```

12.16.50.20 head_height

Параметр задает стандартные пороги для проверки высоты головы.

Формат задания настройки — `object`.

Значение по умолчанию:

```
"threshold": {
    "min": 0.6,
    "max": 0.9
}
```

12.16.50.21 eye_distance

Параметр задает стандартные пороги для проверки расстояния между глазами.

Формат задания настройки — `object`.

Значение по умолчанию:

```
"threshold": {
    "min": 90
}
```

12.16.50.22 image_width

Параметр задает стандартные пороги для проверки ширины изображения.

Формат задания настройки — `object`.

Значение по умолчанию:

```
"threshold": {
    "min": 180,
    "max": 1920
}
```

12.16.50.23 image_height

Параметр задает стандартные пороги для проверки высоты изображения.

Формат задания настройки — `object`.

Значение по умолчанию:

```
  "threshold": {
      "min": 180,
      "max": 1080
  }
```

12.16.50.24 aspect_ratio

Параметр задает стандартные пороги для проверки соотношения сторон.

Формат задания настройки — `object`.

Значение по умолчанию:

```
  "threshold": {
      "min": 0.74,
      "max": 0.8
  }
```

12.16.50.25 face_width

Параметр задает стандартные пороги для проверки ширины лица.

Формат задания настройки — `object`.

Значение по умолчанию:

```
  "threshold": {
      "min": 180
  }
```

12.16.50.26 face_height

Параметр задает стандартные пороги для проверки высоты лица.

Формат задания настройки — `object`.

Значение по умолчанию:

```
  "threshold": {
      "min": 180
  }
```

12.16.50.27 indent_left

Параметр задает стандартные пороги для проверки отступа от левого края изображения.

Формат задания настройки — `object`.

Значение по умолчанию:

```
"threshold": {
    "min": 20
}
```

12.16.50.28 indent_right

Параметр задает стандартные пороги для проверки отступа от правого края изображения.

Формат задания настройки — `object`.

Значение по умолчанию:

```
"threshold": {
    "min": 20
}
```

12.16.50.29 indent_upper

Параметр задает стандартные пороги для проверки отступа от верхнего края изображения.

Формат задания настройки — `object`.

Значение по умолчанию:

```
"threshold": {
    "min": 20
}
```

12.16.50.30 indent_lower

Параметр задает стандартные пороги для проверки отступа от нижнего края изображения.

Формат задания настройки — `object`.

Значение по умолчанию:

```
  "threshold": {
      "min": 20
  }
```

12.16.50.31 image_size

Параметр задает стандартные пороги для проверки размера изображения.

Формат задания настройки — `object`.

Значение по умолчанию:

```
  "threshold": {
      "min": 5120,
      "max": 2097152
  }
```

12.16.50.32 illumination_uniformity

Параметр задает стандартные пороги для проверки равномерности освещения по стандарту ICAO.

Формат задания настройки — `object`.

Значение по умолчанию:

```
  "threshold": {
      "min": 0.5
  }
```

12.16.50.33 dynamic_range

Параметр задает стандартные пороги для проверки динамического диапазона.

Формат задания настройки — `object`.

Значение по умолчанию:

```
  "threshold": {
      "min": 0.5
  }
```

12.16.50.34 eyebrows

Параметр задает стандартные пороги для проверки состояния бровей.

Формат задания настройки — `array`.

Значение по умолчанию:

```
  "threshold": [
      "neutral"
  ]
```

12.16.50.35 shoulders

Параметр задает стандартные пороги для проверки состояния плеч.

Формат задания настройки — `array`.

Значение по умолчанию:

```
  "threshold": [
      "parallel"
  ]
```

12.16.50.36 smile

Параметр задает стандартные пороги для проверки состояния улыбки.

Формат задания настройки — `array`.

Значение по умолчанию:

```
  "threshold": [
      "none"
  ]
```

12.16.50.37 headwear

Параметр задает стандартные пороги для проверки состояния головного убора.

Формат задания настройки — `array`.

Значение по умолчанию:

```
  "threshold": [
      "none"
  ]
```

12.16.50.38 natural_light

Параметр задает стандартные пороги для проверки естественности освещения.

Значение порога по умолчанию, при котором LUNA PLATFORM будет считать, что проверка выполнена:

```
  "threshold": 1
```

Значение порога по умолчанию, интерпретирующего ответ эстиматора:

```
  "min": 0.5
```

12.16.50.39 fish_eye

Параметр задает стандартные пороги для проверки бочкообразной дисторсии.

Значение порога по умолчанию, при котором LUNA PLATFORM будет считать, что проверка выполнена:

```
  "threshold": 0
```

Значение порога по умолчанию, интерпретирующего ответ эстиматора:

```
  "min": 0.5
```

12.16.50.40 red_eye

Параметр задает стандартные пороги для проверки наличия эффекта красных глаз.

Значение порога по умолчанию, при котором LUNA PLATFORM будет считать, что проверка выполнена:

```
  "threshold": 0
```

Значение порога по умолчанию, интерпретирующего ответ эстиматора:

```
  "min": 0.5
```

12.16.50.41 color_type

Параметр задает стандартные пороги для проверки типа цвета изображения на основе лица.

Значение порога по умолчанию, при котором LUNA PLATFORM будет считать, что проверка выполнена:

```
  "threshold": [
      "color"
  ]
```

Значение порога по умолчанию, интерпретирующего ответ эстиматора:

```
  "infrared": {
      "threshold": {
          "min": 0.5
      }
  },
  "color": {
      "threshold": {
          "min": 0.5
      }
  }
```

12.16.50.42 background_lightness

Параметр задает стандартные пороги для проверки яркости фона.

Формат задания настройки — `object`.

Значение по умолчанию:

```
  "threshold": {
      "min": 0.2
  }
```

12.16.50.43 background_uniformity

Параметр задает стандартные пороги для проверки однородности фона.

Формат задания настройки — `object`.

Значение по умолчанию:

```
  "threshold": {
      "min": 0.5
  }
```

12.16.50.44 face_occlusion

Параметр задает стандартный порог для проверки общего перекрытия лица.

Формат задания настройки — `object`.

Значение по умолчанию - `0`.

12.16.50.45 lower_face_occlusion

Параметр задает стандартный порог для проверки перекрытия нижней части лица.

Формат задания настройки — `object`.

Значение по умолчанию - `0`.

12.16.50.46 forehead_occlusion

Параметр задает стандартный порог для проверки перекрытия лба.

Формат задания настройки — `object`.

Значение по умолчанию - `0`.

12.16.50.47 nose_occlusion

Параметр задает стандартный порог для проверки перекрытия носа.

Формат задания настройки — `object`.

Значение по умолчанию - `0`.

12.16.51 Группа параметров DESCRIPTOR_ENCRYPTION

Данная группа параметров задает настройки шифрования биометрических шаблонов для повышения безопасности и предотвращения несанкционированного использования.

См. подробную информацию в разделе «Шифрование биометрических шаблонов».

12.16.51.1 enabled

Параметр включает шифрование биометрических шаблонов.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.16.51.2 algorithm

Параметр задает название используемого алгоритма шифрования.

Формат задания настройки — `string`.

Значение по умолчанию — `aes256-gcm`.

12.16.51.3 params > source

Параметр задает название источника ключа шифрования. Поддерживаются два типа источников: `raw` и `vaultKV`.

Формат задания настройки — `string`.

Значение по умолчанию — `raw`.

12.16.51.4 params > key

Параметр задает ключ шифрования или учетные данные для его получения из указанного источника.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.16.52 Прочие

12.16.52.1 luna_remote_sdk_active_plugins

Параметр задает список плагинов, которые должен использовать сервис.

Имена задаются в следующем формате:

```
  [
     "plugin_1",
     "plugin_2",
     "plugin_3"
  ]
```

> Список должен содержать имена файлов без расширения (.py).

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.16.52.2 storage_time

Параметр задает формат времени, используемый для записей в базе данных. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.16.52.3 default_face_descriptor_version

Параметр задает используемую версию биометрического шаблона лица.

Формат задания настройки — `string`.

Значение по умолчанию — `59`.

Примечание. См. подробную информацию о версиях биометрических шаблонах в разделе «Нейронные сети».

12.16.52.4 default_human_descriptor_version

Параметр задает используемую версию биометрического шаблона тела.

См. подробную информацию о версиях биометрических шаблонах в разделе «Нейронные сети».

Формат задания настройки — `string`.

Значение по умолчанию — `116`.

Примечание. См. подробную информацию о версиях биометрических шаблонах в разделе «Нейронные сети».

12.16.52.5 luna_remote_sdk_detector_type

Параметр задает тип используемого детектора.

В настоящий момент доступен только один тип детектора — «FACE_DET_V3».

Формат задания настройки — `string`.

Значение по умолчанию — `FACE_DET_V3`.

12.16.52.6 luna_remote_sdk_use_auto_rotation

Параметр позволяет включить автоориентацию повернутого изображения.

См. описание работы автоориентации в разделе «Автоориентация повернутого изображения».

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.17 Настройки сервиса Video Agent

Данный раздел описывает параметры сервиса Video Agent.

Настройку сервиса можно выполнить с помощью сервиса Configurator.

12.17.1 Группа параметров LUNA_CONFIGURATOR

Данная группа параметров задаёт настройки подключения сервиса Video Agent к сервису Configurator.

Данная группа параметров не будет видна в пользовательском интерфейсе сервиса Configurator. Параметры можно изменить только в конфигурационном файле «config.conf», расположенном в директории «/srv/luna_video_agent/configs/» соответствующего контейнера.

12.17.1.1 use_configurator

Параметр позволяет включить использование сервиса Configurator.

> С помощью сервиса Configurator упрощается настройка сервисов LP. Сервис сохраняет все необходимые конфигурации для всех сервисов LP в одном месте.

При отключенном параметре будут использованы настройки из файла «config.conf», расположенного в директории «/srv/luna_video_agent/configs/» соответствующего контейнера.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.17.1.2 luna_configurator_origin

Параметр задает протокол, IP адрес и порт сервиса Configurator.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5070`.

12.17.1.3 luna_configurator_api

Параметр задает версию API сервиса Configurator. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.2 Группа параметров LUNA_VIDEO_MANAGER_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Video Manager.

12.17.2.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Video Manager. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.17.2.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.17.2.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.17.2.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.17.3 Группа параметров LUNA_VIDEO_MANAGER_ADDRESS

Данная группа параметров задает настройки подключения к сервису Video Manager.

12.17.3.1 origin

Параметр задает протокол, IP адрес и порт сервиса Video Manager.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Video Manager, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Video Manager.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5230`.

12.17.3.2 api_version

Параметр задает версию API сервиса Video Manager. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.4 Группа параметров LUNA_VIDEO_AGENT_VIDEO_SETTINGS

Данная группа параметров задает настройки обработки видео для выполнения видеоаналитики.

12.17.4.1 decoder_device_class

Параметр задает тип устройства видеодекодера. Доступны следующие варианты:

- «cpu»
- «gpu»
- «auto» (в приоритете GPU)

Формат задания настройки — `string`.

Значение по умолчанию — `auto`.

12.17.4.2 decoder_worker_count

Параметр задает количество «рабочих процессов» видеодекодера.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.17.4.3 frame_pool_size

Параметр задает количество кадров, которые надо заранее выделить для обработки потока.

Это ускоряет процесс декодирования, особенно на графическом процессоре (GPU), за счёт того, что кадры уже зарезервированы и не нужно каждый раз выделять память для нового кадра.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.17.4.4 ffmpeg_thread_count

Параметр задает количество потоков (threads), которые будут использованы FFmpeg для выполнения декодирования видео.

Значение 0 означает, что система будет использовать автоматическое количество потоков, которое выберет сам FFmpeg в зависимости от конфигурации устройства.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.17.4.5 max_size

Параметр задает максимальный размер видео для выполнения видеоаналитики.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024` (мегабайты).

12.17.5 Группа параметров LUNA_VIDEO_AGENT_RUNTIME_SETTINGS

Данная группа параметров задает глобальные настройки для всех эстиматоров/детекторов.

12.17.5.1 global_device_class

Параметр задает тип устройства («cpu» или «gpu») для всех эстиматоров/детекторов, у которых значение параметра «device_class» = «global».

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.17.5.2 num_threads

Параметр задает количество рабочих потоков CPU-процессора, которые будут использоваться сервисом при выполнении эстимации/детекции. Желательно, чтобы значение этого параметра соответствовало числу физических ядер процессора для достижения максимальной производительности. Однако, слишком большое число потоков может привести к ухудшению производительности из-за дополнительных накладных расходов.

Формат задания настройки — `integer`.

Значение по умолчанию — `4`.

12.17.5.3 num_compute_streams

Параметр задает количество потоков вычислений на GPU-процессоре, которые будут использоваться сервисом при выполнении эстимации/детекции. Рекомендуется выбирать значение, соответствующее характеристикам конкретного GPU и задаче. Стоит учитывать, что NVIDIA может изменять поведение этого параметра в различных версиях программного обеспечения, поэтому рекомендуется проводить тестирование для определения оптимального значения.

Формат задания настройки — `integer`.

Значение по умолчанию — `6`.

12.17.6 Группа параметров LUNA_VIDEO_AGENT_PEOPLE_COUNT_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора количества людей на изображении.

12.17.6.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.17.6.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.17.6.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.7 Группа параметров LUNA_VIDEO_AGENT_AGS_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора AGS.

12.17.7.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.17.7.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.17.7.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.8 Группа параметров LUNA_VIDEO_AGENT_BASIC_ATTRIBUTES_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора базовых атрибутов.

12.17.8.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.17.8.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.17.8.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.9 Группа параметров LUNA_VIDEO_AGENT_BODY_ATTRIBUTES_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора параметров тел.

12.17.9.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.17.9.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.17.9.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.10 Группа параметров LUNA_VIDEO_AGENT_BODY_DESCRIPTOR_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора извлечения биометрического шаблона тела.

12.17.10.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.17.10.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.17.10.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.11 Группа параметров LUNA_VIDEO_AGENT_BODY_WARP_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора биометрического образца тела.

12.17.11.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.17.11.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.17.11.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.12 Группа параметров LUNA_VIDEO_AGENT_DEEPFAKE_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора Deepfake.

12.17.12.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.17.12.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.17.12.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.13 Группа параметров LUNA_VIDEO_AGENT_EMOTIONS_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора эмоций.

12.17.13.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.17.13.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.17.13.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.14 Группа параметров LUNA_VIDEO_AGENT_EYES_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора атрибутов глаз.

12.17.14.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.17.14.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.17.14.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.15 Группа параметров LUNA_VIDEO_AGENT_FACE_DESCRIPTOR_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора извлечения биометрического шаблона лица.

12.17.15.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.17.15.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.17.15.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.16 Группа параметров LUNA_VIDEO_AGENT_FACE_LANDMARKS5_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора 5 контрольных точек лица.

12.17.16.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.17.16.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.17.16.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.17 Группа параметров LUNA_VIDEO_AGENT_FACE_WARP_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора биометрического образца лица.

12.17.17.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.17.17.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.17.17.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.18 Группа параметров LUNA_VIDEO_AGENT_FACE_WARP_QUALITY_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора качества биометрического образца лица.

12.17.18.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.17.18.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.17.18.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.19 Группа параметров LUNA_VIDEO_AGENT_GAZE_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора направления взгляда.

12.17.19.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.17.19.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.17.19.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.20 Группа параметров LUNA_VIDEO_AGENT_GLASSES_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора очков.

12.17.20.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.17.20.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.17.20.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.21 Группа параметров LUNA_VIDEO_AGENT_HEAD_POSE_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора положения головы.

12.17.21.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.17.21.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.17.21.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.22 Группа параметров LUNA_VIDEO_AGENT_LIVENESS_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора Liveness.

12.17.22.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.17.22.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.17.22.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.23 Группа параметров LUNA_VIDEO_AGENT_MASK_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора маски.

12.17.23.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.17.23.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.17.23.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.24 Группа параметров LUNA_VIDEO_AGENT_MOUTH_ATTRIBUTES_ESTIMATOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы эстиматора атрибутов рта.

12.17.24.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.17.24.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.17.24.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.25 Группа параметров LUNA_VIDEO_AGENT_HUMAN_TRACKER_SETTINGS

Данная группа параметров задает настройки отслеживания тел.

12.17.25.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `global`.

12.17.25.2 runtime_settings > optimal_batch_size

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.17.25.3 estimator_settings > detector_step

Задает количество кадров, на которых будет выполняться редетекция лица или тела в заданной области до выполнения детекции лица или тела.

Формат задания настройки — `string`.

Значение по умолчанию — `7`.

12.17.25.4 estimator_settings > scale_result_size

Определяет, до какого размера (в пикселях) масштабируется кадр перед тем, как он будет отправлен на детекцию. Размер масштабируется по максимальному измерению (ширина или высота).

Формат задания настройки — `integer`.

Значение по умолчанию — `640`.

12.17.25.5 estimator_settings > skip_frames

Определяет количество кадров, после которых трек объекта считается завершённым, если объект не был обновлён ни детектором, ни редетекцией. Например, значение 18 указывает, что если объект не обновляется в течение 18 кадров, его трек считается завершённым и больше не отслеживается.

Формат задания настройки — `integer`.

Значение по умолчанию — `18`.

12.17.26 Группа параметров LUNA_VIDEO_AGENT_HUMAN_DETECTOR_SETTINGS

Данная группа параметров задает индивидуальные настройки работы одновременного детектора лица и тела.

12.17.26.1 runtime_settings > device_class

Параметр задает тип устройства для выполнения оценки («cpu», «gpu» или «global»)

Формат задания настройки — `string`.

Значение по умолчанию — `cpu`.

12.17.26.2 runtime_settings > num_threads

Параметр задает размер батча для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `10`.

12.17.26.3 runtime_settings > num_compute_streams

Параметр задает количество «рабочих процессов» для выполнения оценки.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.27 Группа параметров LUNA_VIDEO_AGENT_LOGGER

Данная группа параметров задает настройки логирования.

12.17.27.1 log_level

Параметр задает уровень отладочной печати, по приоритету: «ERROR», «WARNING», «INFO», «DEBUG».

Формат задания настройки — `string`.

Значение по умолчанию — `INFO`.

12.17.27.2 log_time

Параметр задает формат времени, используемый в записях лога. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.17.27.3 log_to_stdout

Параметр позволяет отправлять логи в стандартный вывод (stdout).

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`

12.17.27.4 log_to_file

Параметр позволяет сохранять логи в файл. Директория с файлами логов указывается в параметре «folder_with_logs».

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.17.27.5 folder_with_logs

Параметр задает директорию, в которой хранятся логи. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить параметр «log_to_file».

Формат задания настройки — `string`.

Значение по умолчанию — `./`

Пример:

```
  "folder_with_logs": "/srv/logs"
```

12.17.27.6 max_log_file_size

Параметр задает максимальный размер файла лога в МБ перед выполнением его ротации (0 — не использовать ротацию).

Для использования данного параметра требуется включить параметр «log_to_file».

> При необходимости можно настроить ротацию логов Docker. См. раздел «Настройка ротации логов Docker» в руководстве по установке LUNA PLATFORM.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024`

12.17.27.7 multiline_stack_trace

Параметр включает многострочную трассировку стека в логах. Когда параметр включен, информация о стеке вызовов записывается в логах так, что каждый фрейм стека помещается на отдельной строке, что улучшает читаемость. Если параметр выключен, информация о стеке вызовов записывается на одной строке, что может сделать логи менее удобными для анализа.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.17.27.8 format

Параметр определяет формат выводимых логов. Доступны следующие значения:

- «default» — стандартный формат вывода логов LUNA PLATFORM
- «json» — вывод логов в формате json
- «ecs» — вывод логов в формате ECS (Elastic Common Schema)

При использовании значения «ecs» будут использоваться следующие поля:

- «http.response.status_code» — содержит код состояния ответа HTTP (200, 404, 500 и т.д.);
- «http.response.execution_time» — содержит информацию о времени, затраченном на выполнение запроса и получение ответа;
- «http.request.method» — содержит метод HTTP-запроса (GET, POST, PUT и т.д.);
- «url.path» — содержит путь в URL-адресе запроса;
- «error.code» — содержит код ошибки, если запрос завершается с ошибкой.

Формат задания настройки — `string`.

Значение по умолчанию — `default`.

12.17.28 Группа параметров LUNA_VIDEO_AGENT_HTTP_SETTINGS

В данной группе параметров содержатся настройки, отвечающие за обработку HTTP-подключений. См. подробную информацию по следующей ссылке: https://sanic.dev/en/guide/deployment/configuration.html#builtin-values

12.17.28.1 request_timeout

Параметр задает продолжительность времени между моментом, когда новое открытое TCP-соединение передается на сервер, и моментом, когда получен весь HTTP-запрос.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.17.28.2 response_timeout

Параметр задает продолжительность времени между моментом, когда сервер передает HTTP-запрос приложению, и моментом, когда HTTP-ответ отправляется клиенту.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `600`.

12.17.28.3 request_max_size

Параметр задает максимальный размер запроса.

Формат задания настройки — `integer` (байты).

Значение по умолчанию — `1073741824`.

12.17.28.4 keep_alive_timeout

Параметр задает тайм-аут поддержания активности HTTP.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `15`.

12.17.29 Группа параметров LUNA_SERVICE_METRICS

Данная группа параметров включает и настраивает сбор метрик в формате Prometheus.

См. подробную информацию в разделе «Мониторинг».

12.17.29.1 enabled

Параметр включает сбор метрик.

Если сбор метрик отключен, то запрос к ресурсу `/metrics` вернет соответствующее сообщение.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.17.29.2 metrics_format

Параметр задает формат метрик.

На данный момент поддерживается только формат Prometheus.

См. официальную документацию Prometheus для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `prometheus`.

12.17.29.3 extra_labels

Параметр задает пользовательские типы лейблов.

Формат задания настройки — `label_name=label_value`.

Значение по умолчанию не задано.

12.17.30 Группа параметров LUNA_LICENSES_ADDRESS

Данная группа параметров задает настройки подключения к сервису Licenses.

12.17.30.1 origin

Параметр задает протокол, IP адрес и порт сервиса Licenses.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Licenses, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Licenses.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5120`.

12.17.30.2 api_version

Параметр задает версию API сервиса Licenses. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.31 Группа параметров LUNA_SENDER_ADDRESS

Данная группа параметров задает настройки подключения к сервису Sender.

12.17.31.1 origin

Параметр задает протокол, IP адрес и порт сервиса Sender.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Sender, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Sender.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5080`.

12.17.31.2 api_version

Параметр задает версию API сервиса Sender. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.32 Группа параметров LUNA_HANDLERS_ADDRESS

Данная группа параметров задает настройки подключения к сервису Handlers.

12.17.32.1 origin

Параметр задает протокол, IP адрес и порт сервиса Handlers.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Handlers, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Handlers.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5090`.

12.17.32.2 api_version

Параметр задает версию API сервиса Handlers. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.33 Группа параметров LUNA_HANDLERS_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Handlers.

12.17.33.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Handlers. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.17.33.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.17.33.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.17.33.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.17.34 Группа параметров LUNA_IMAGE_STORE_IMAGES_ADDRESS

В данной группе параметров задаются настройки бакета для хранения исходных изображений.

12.17.34.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.17.34.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.34.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-image-origin`.

12.17.35 Группа параметров LUNA_IMAGE_STORE_IMAGES_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к бакету, хранящему исходных изображений.

12.17.35.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к бакету, хранящему исходных изображений. Это время ожидания, в течение которого клиент пытается установить соединение с бакетом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `30`.

12.17.35.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.17.36 Группа параметров LUNA_IMAGE_STORE_OBJECTS_ADDRESS

В данной группе параметров задаются настройки бакета для хранения объектов.

12.17.36.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.17.36.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.36.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-objects`.

12.17.37 Группа параметров LUNA_MONITORING

Данная группа параметров задает настройки мониторинга.

См. подробную информацию о мониторинге в разделе «Мониторинг».

12.17.37.1 storage_type

Тип хранилища для хранения данных мониторинга.

В настоящее время доступно только БД Influx.

Формат задания настройки — `string`.

Значение по умолчанию — `influx`.

12.17.37.2 send_data_for_monitoring

Параметр позволяет включить или отключить отправку данных для мониторинга в InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.17.37.3 use_ssl

Параметр позволяет использовать HTTPS для подключения к InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.17.37.4 organization

Параметр задает рабочую область InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.17.37.5 token

Параметр задает аутентификации InfluxDB 2.x.

Формат задания настройки — `string`.

12.17.37.6 bucket

Параметр задаёт имя бакета InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_monitoring`.

12.17.37.7 host

Параметр задаёт IP-адрес InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.17.37.8 port

Параметр задает порт InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `8086`.

12.17.37.9 flushing_period

Параметр задает частоту отправки данных мониторинга в InfluxDB.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `1`.

12.17.38 Группа параметров ADDITIONAL_SERVICES_USAGE

12.17.38.1 luna_events

Параметр задает возможность использования сервиса Events.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.17.38.2 luna_handlers

Параметр задает возможность использования сервиса Handlers.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.17.38.3 luna_faces

Параметр задает возможность использования сервиса Faces.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.17.38.4 luna_sender

Параметр задает возможность использования сервиса Sender.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.17.38.5 luna_matcher_proxy

Параметр задает возможность использования сервиса Python Matcher Proxy.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.17.38.6 luna_image_store

Параметр задает возможность использования сервиса Image Store.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.17.38.7 luna_lambda

Параметр задает возможность использования сервиса Lambda.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.17.38.8 luna_video_manager

Параметр задает возможность использования сервиса Video Manager.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.17.38.9 luna_video_agent

Параметр задает возможность использования сервиса Video Agent.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.17.38.10 luna_streams_retranslator

Параметр задает возможность использования сервиса Streams Retranslator.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.17.39 Группа параметров EXTERNAL_LUNA_API_ADDRESS

Данная группа параметров предназначена для корректной обработки изображений, сохраняемых сервисом Video Agent, при использовании «target» = «overview» (сохранении изображений).

12.17.39.1 origin

Параметр задает протокол, IP адрес и порт сервиса API.

См. описание логики работы в разделе «Группа параметров EXTERNAL_LUNA_API_ADDRESS».

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5000`.

12.17.39.2 api_version

Параметр задает версию API сервиса API.

См. описание логики работы в разделе «Группа параметров EXTERNAL_LUNA_API_ADDRESS».

Формат задания настройки — `integer`.

Значение по умолчанию — `6`.

12.17.40 Прочие

12.17.40.1 luna_video_agent_active_plugins

Параметр задает список плагинов, которые должен использовать сервис.

Имена задаются в следующем формате:

```
[
   "plugin_1",
   "plugin_2",
   "plugin_3"
]
```

> Список должен содержать имена файлов без расширения (.py).

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.18 Настройки сервиса Video Manager

Данный раздел описывает параметры сервиса Video Manager.

Настройку сервиса можно выполнить с помощью сервиса Configurator.

12.18.1 Группа параметров LUNA_CONFIGURATOR

Данная группа параметров задаёт настройки подключения сервиса Video Manager к сервису Configurator.

Данная группа параметров не будет видна в пользовательском интерфейсе сервиса Configurator. Параметры можно изменить только в конфигурационном файле «config.conf», расположенном в директории «/srv/luna_video_manager/configs/» соответствующего контейнера.

12.18.1.1 use_configurator

Параметр позволяет включить использование сервиса Configurator.

> С помощью сервиса Configurator упрощается настройка сервисов LP. Сервис сохраняет все необходимые конфигурации для всех сервисов LP в одном месте.

При отключенном параметре будут использованы настройки из файла «config.conf», расположенного в директории «/srv/luna_video_manager/configs/» соответствующего контейнера.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.18.1.2 luna_configurator_origin

Параметр задает протокол, IP адрес и порт сервиса Configurator.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5070`.

12.18.1.3 luna_configurator_api

Параметр задает версию API сервиса Configurator. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.18.1.4 luna_video_manager_agent_status_obsoleting_interval

Параметр задает период выполнения периодической задачи сервиса Video Manager для проверки условий обработки потоков в случае отсутствия запроса агента к сервису Video Manager. См. раздел «Управление статусом агента в случае отсутствия запроса агента».

Рекомендуется проконсультироваться со специалистами VisionLabs перед изменением данного параметра.

Формат задания настройки — `number`.

Значение по умолчанию — `5`.

12.18.1.5 luna_video_manager_agent_status_obsoleting_period

Параметр задает период выполнения периодической задачи сервиса Video Manager для проверки условий обработки потоков в случае отсутствия запроса агента к сервису Video Manager. См. раздел «Управление статусом агента в случае отсутствия запроса агента».

Рекомендуется проконсультироваться со специалистами VisionLabs перед изменением данного параметра.

Формат задания настройки — `number`.

Значение по умолчанию — `20`.

12.18.1.6 luna_video_manager_stream_status_obsoleting_interval

Параметр задает период выполнения периодической задачи сервиса Video Manager для проверки условий обработки потоков в случае отсутствия обратной связи от агента к сервису Video Manager. См. раздел «Обработка потоков в случае не появления обратной связи».

Рекомендуется проконсультироваться со специалистами VisionLabs перед изменением данного параметра.

Формат задания настройки — `number`.

Значение по умолчанию — `5`.

12.18.1.7 luna_video_manager_stream_status_obsoleting_period

Параметр задает период выполнения периодической задачи сервиса Video Manager для проверки условий обработки потоков в случае отсутствия обратной связи от агента к сервису Video Manager. См. раздел «Обработка потоков в случае не появления обратной связи».

Рекомендуется проконсультироваться со специалистами VisionLabs перед изменением данного параметра.

Формат задания настройки — `number`.

Значение по умолчанию — `20`.

12.18.1.8 luna_video_manager_streams_agent_search_interval

Параметр задает период выполнения периодической задачи сервиса Video Manager для процесса распределения потоков. См. раздел «Распределение потоков».

Рекомендуется проконсультироваться со специалистами VisionLabs перед изменением данного параметра.

Формат задания настройки — `number`.

Значение по умолчанию — `5`.

12.18.1.9 luna_video_manager_streams_autorestarter_interval

Параметр задает период выполнения периодической задачи сервиса Video Manager для процесса автоматического перезапуска потока. См. раздел «Процесс перезапуска потоков».

Рекомендуется проконсультироваться со специалистами VisionLabs перед изменением данного параметра.

Формат задания настройки — `number`.

Значение по умолчанию — `5`.

12.18.2 Группа параметров LUNA_STREAMS_RETRANSLATOR_ADDRESS

Данная группа параметров задает настройки подключения к сервису Streams Reatranslator.

12.18.2.1 origin

Параметр задает протокол, IP адрес и порт сервиса Streams Reatranslator.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Streams Reatranslator, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Streams Reatranslator.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5250`.

12.18.2.2 api_version

Параметр задает версию API сервиса Streams Reatranslator. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.18.3 Группа параметров LUNA_VIDEO_MANAGER_DB

В данной группе параметров задаются настройки подключения к базе данных сервиса Video-Manager.

12.18.3.1 db_type

Параметр задает тип базы данных. Доступны следующие типы:

- «postgres» — тип СУБД PostgreSQL
- «oracle» — тип СУБД Oracle

Формат задания настройки — `string`.

Значение по умолчанию — `postgres`.

12.18.3.2 db_host

Параметр задает имя сервера (хост) базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.18.3.3 db_port

Параметр задает порт базы данных.

Порт по умолчанию для типа «postgres»- `5432`.

Порт по умолчанию для типа «oracle» — `1521`.

Формат задания настройки — `string`.

Значение по умолчанию — `5432`.

12.18.3.4 db_user

Параметр задает имя пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.18.3.5 db_password

Параметр задает пароль пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.18.3.6 db_name

Параметр задает имя базы данных для типа «postgres» или имя SID для типа «oracle».

Формат задания настройки — `string`.

Значение по умолчанию — `luna_video_manager`.

12.18.3.7 connection_pool_size

Параметр задает размер пула соединений к БД. Реальное количество подключений может быть больше значения данной настройки на 1.

При необходимости в настройке «max_connections» конфигурационного файла PostgreSQL можно задать максимальное количество одновременных подключений к серверу БД. См. раздел «Продвинутая настройка PostrgeSQL» для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `10`.

12.18.3.8 dsn

Параметр задает строку подключения DSN для соединения с БД.

> DSN — это строка подключения, которая идентифицирует и указывает на источник данных (базу данных), к которому нужно установить соединение.

В строке DSN могут задаваться такие настройки, как множественные хосты, аутентификационные данные, порт и другие параметры.

> Настройки зависят от типа БД. Множественные хосты поддерживаются только с PostgreSQL.

> По умолчанию параметр «dsn» не отображается во вкладке «Settings» в Configurator. Проверить список всех доступных параметров для группы настроек можно на вкладке «Limitations».

Пример:

```
{
"dsn": "luna:luna@postgres01:5432,postgres02:5432/luna_video_manager?
    some_option=some_value"
"db_settings": {
    "connection_pool_size": 5
    }
}
```

Здесь:

- «luna:luna» — имя пользователя и пароль для подключения к PostgreSQL;
- «@postgres01:5432,postgres02:5432» — список хостов и портов, разделенных запятыми. Это означает, что сервис будет пытаться подключиться к первому хосту («postgres01») на порту 5432. Если это не удастся, она попытается подключиться ко второму хосту («postgres02») также на порту 5432;
- «/luna_video_manager» — название базы данных;
- «?some_option=some_value» — опциональные параметры для подключения.

При необходимости можно комбинировать строку DSN и классические настройки, однако строка DSN является более приоритетной. Можно частично заполнить строку DSN (например, «postgres01,postgres02/luna_video_manager»), и тогда недостающие параметры будут заполнены из значений параметров «db_host», «db_port», «db_name», «db_user» и «db_password».

При запуске сервис создаст пул подключений к одному из доступных хостов DSN. В случае возникновения проблем с установлением соединения после нескольких неудачных попыток, сервис снова попытается настроить пул подключений к любому из доступных хостов DSN.

12.18.4 Группа параметров LUNA_VIDEO_MANAGER_LOGGER

Данная группа параметров задает настройки логирования.

12.18.4.1 log_level

Параметр задает уровень отладочной печати, по приоритету: «ERROR», «WARNING», «INFO», «DEBUG».

Формат задания настройки — `string`.

Значение по умолчанию — `INFO`.

12.18.4.2 log_time

Параметр задает формат времени, используемый в записях лога. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.18.4.3 log_to_stdout

Параметр позволяет отправлять логи в стандартный вывод (stdout).

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`

12.18.4.4 log_to_file

Параметр позволяет сохранять логи в файл. Директория с файлами логов указывается в параметре «folder_with_logs».

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.18.4.5 folder_with_logs

Параметр задает директорию, в которой хранятся логи. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить параметр «log_to_file».

Формат задания настройки — `string`.

Значение по умолчанию — `./`

Пример:

```
  "folder_with_logs": "/srv/logs"
```

12.18.4.6 max_log_file_size

Параметр задает максимальный размер файла лога в МБ перед выполнением его ротации (0 — не использовать ротацию).

Для использования данного параметра требуется включить параметр «log_to_file».

> При необходимости можно настроить ротацию логов Docker. См. раздел «Настройка ротации логов Docker» в руководстве по установке LUNA PLATFORM.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024`

12.18.4.7 multiline_stack_trace

Параметр включает многострочную трассировку стека в логах. Когда параметр включен, информация о стеке вызовов записывается в логах так, что каждый фрейм стека помещается на отдельной строке, что улучшает читаемость. Если параметр выключен, информация о стеке вызовов записывается на одной строке, что может сделать логи менее удобными для анализа.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.18.4.8 format

Параметр определяет формат выводимых логов. Доступны следующие значения:

- «default» — стандартный формат вывода логов LUNA PLATFORM
- «json» — вывод логов в формате json
- «ecs» — вывод логов в формате ECS (Elastic Common Schema)

При использовании значения «ecs» будут использоваться следующие поля:

- «http.response.status_code» — содержит код состояния ответа HTTP (200, 404, 500 и т.д.);
- «http.response.execution_time» — содержит информацию о времени, затраченном на выполнение запроса и получение ответа;
- «http.request.method» — содержит метод HTTP-запроса (GET, POST, PUT и т.д.);
- «url.path» — содержит путь в URL-адресе запроса;
- «error.code» — содержит код ошибки, если запрос завершается с ошибкой.

Формат задания настройки — `string`.

Значение по умолчанию — `default`.

12.18.5 Группа параметров LUNA_VIDEO_MANAGER_HTTP_SETTINGS

В данной группе параметров содержатся настройки, отвечающие за обработку HTTP-подключений. См. подробную информацию по следующей ссылке: https://sanic.dev/en/guide/deployment/configuration.html#builtin-values

12.18.5.1 request_timeout

Параметр задает продолжительность времени между моментом, когда новое открытое TCP-соединение передается на сервер, и моментом, когда получен весь HTTP-запрос.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.18.5.2 response_timeout

Параметр задает продолжительность времени между моментом, когда сервер передает HTTP-запрос приложению, и моментом, когда HTTP-ответ отправляется клиенту.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `600`.

12.18.5.3 request_max_size

Параметр задает максимальный размер запроса.

Формат задания настройки — `integer` (байты).

Значение по умолчанию — `1073741824`.

12.18.5.4 keep_alive_timeout

Параметр задает тайм-аут поддержания активности HTTP.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `15`.

12.18.6 Группа параметров LUNA_SERVICE_METRICS

Данная группа параметров включает и настраивает сбор метрик в формате Prometheus.

См. подробную информацию в разделе «Мониторинг».

12.18.6.1 enabled

Параметр включает сбор метрик.

Если сбор метрик отключен, то запрос к ресурсу `/metrics` вернет соответствующее сообщение.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.18.6.2 metrics_format

Параметр задает формат метрик.

На данный момент поддерживается только формат Prometheus.

См. официальную документацию Prometheus для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `prometheus`.

12.18.6.3 extra_labels

Параметр задает пользовательские типы лейблов.

Формат задания настройки — `label_name=label_value`.

Значение по умолчанию не задано.

12.18.7 Группа параметров LUNA_MONITORING

Данная группа параметров задает настройки мониторинга.

См. подробную информацию о мониторинге в разделе «Мониторинг».

12.18.7.1 storage_type

Тип хранилища для хранения данных мониторинга.

В настоящее время доступно только БД Influx.

Формат задания настройки — `string`.

Значение по умолчанию — `influx`.

12.18.7.2 send_data_for_monitoring

Параметр позволяет включить или отключить отправку данных для мониторинга в InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.18.7.3 use_ssl

Параметр позволяет использовать HTTPS для подключения к InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.18.7.4 organization

Параметр задает рабочую область InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.18.7.5 token

Параметр задает аутентификации InfluxDB 2.x.

Формат задания настройки — `string`.

12.18.7.6 bucket

Параметр задаёт имя бакета InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_monitoring`.

12.18.7.7 host

Параметр задаёт IP-адрес InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.18.7.8 port

Параметр задает порт InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `8086`.

12.18.7.9 flushing_period

Параметр задает частоту отправки данных мониторинга в InfluxDB.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `1`.

12.18.8 Прочие

12.18.8.1 luna_video_manager_active_plugins

Параметр задает список плагинов, которые должен использовать сервис.

Имена задаются в следующем формате:

```
[
   "plugin_1",
   "plugin_2",
   "plugin_3"
]
```

> Список должен содержать имена файлов без расширения (.py).

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.18.8.2 storage_time

Параметр задает формат времени, используемый для записей в базе данных. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.19 Настройки сервиса Streams Retranslator

Данный раздел описывает параметры сервиса Streams Retranslator.

Настройку сервиса можно выполнить с помощью сервиса Configurator.

12.19.1 Группа параметров LUNA_CONFIGURATOR

Данная группа параметров задаёт настройки подключения сервиса Streams Retranslator к сервису Configurator.

Данная группа параметров не будет видна в пользовательском интерфейсе сервиса Configurator. Параметры можно изменить только в конфигурационном файле «config.conf», расположенном в директории «/srv/luna_streams_retranslator/configs/» соответствующего контейнера.

12.19.1.1 use_configurator

Параметр позволяет включить использование сервиса Configurator.

> С помощью сервиса Configurator упрощается настройка сервисов LP. Сервис сохраняет все необходимые конфигурации для всех сервисов LP в одном месте.

При отключенном параметре будут использованы настройки из файла «config.conf», расположенного в директории «/srv/luna_streams_retranslator/configs/» соответствующего контейнера.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.19.1.2 luna_configurator_origin

Параметр задает протокол, IP адрес и порт сервиса Configurator.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5070`.

12.19.1.3 luna_configurator_api

Параметр задает версию API сервиса Configurator. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.19.2 Группа параметров LUNA_STREAMS_RETRANSLATOR_LOGGER

Данная группа параметров задает настройки логирования.

12.19.2.1 log_level

Параметр задает уровень отладочной печати, по приоритету: «ERROR», «WARNING», «INFO», «DEBUG».

Формат задания настройки — `string`.

Значение по умолчанию — `INFO`.

12.19.2.2 log_time

Параметр задает формат времени, используемый в записях лога. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.19.2.3 log_to_stdout

Параметр позволяет отправлять логи в стандартный вывод (stdout).

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`

12.19.2.4 log_to_file

Параметр позволяет сохранять логи в файл. Директория с файлами логов указывается в параметре «folder_with_logs».

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.19.2.5 folder_with_logs

Параметр задает директорию, в которой хранятся логи. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить параметр «log_to_file».

Формат задания настройки — `string`.

Значение по умолчанию — `./`

Пример:

```
  "folder_with_logs": "/srv/logs"
```

12.19.2.6 max_log_file_size

Параметр задает максимальный размер файла лога в МБ перед выполнением его ротации (0 — не использовать ротацию).

Для использования данного параметра требуется включить параметр «log_to_file».

> При необходимости можно настроить ротацию логов Docker. См. раздел «Настройка ротации логов Docker» в руководстве по установке LUNA PLATFORM.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024`

12.19.2.7 multiline_stack_trace

Параметр включает многострочную трассировку стека в логах. Когда параметр включен, информация о стеке вызовов записывается в логах так, что каждый фрейм стека помещается на отдельной строке, что улучшает читаемость. Если параметр выключен, информация о стеке вызовов записывается на одной строке, что может сделать логи менее удобными для анализа.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.19.2.8 format

Параметр определяет формат выводимых логов. Доступны следующие значения:

- «default» — стандартный формат вывода логов LUNA PLATFORM
- «json» — вывод логов в формате json
- «ecs» — вывод логов в формате ECS (Elastic Common Schema)

При использовании значения «ecs» будут использоваться следующие поля:

- «http.response.status_code» — содержит код состояния ответа HTTP (200, 404, 500 и т.д.);
- «http.response.execution_time» — содержит информацию о времени, затраченном на выполнение запроса и получение ответа;
- «http.request.method» — содержит метод HTTP-запроса (GET, POST, PUT и т.д.);
- «url.path» — содержит путь в URL-адресе запроса;
- «error.code» — содержит код ошибки, если запрос завершается с ошибкой.

Формат задания настройки — `string`.

Значение по умолчанию — `default`.

12.19.3 Группа параметров LUNA_RETRANSLATOR_DB_ADDRESS

12.19.3.1 user

Параметр задает имя пользователя БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.19.3.2 password

Параметр задает пароль БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию не указывается.

12.19.3.3 host

Параметр задает IP-адрес БД Redis.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.19.3.4 port

Параметр задает номер порта, на котором Redis ожидает входящие сетевые соединения и прослушивает их для выполнения команд от клиентов.

Формат задания настройки — `integer`.

Значение по умолчанию — `6379`.

12.19.3.5 sentinel > master_name

Параметр задает имя мастера базы данных Redis, который контролируется и управляется системой Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_streams_retranslator_master`.

12.19.3.6 sentinel > sentinels

Параметр задает список адресов и портов серверов Sentinel, которые будут использоваться клиентами для обнаружения и мониторинга БД Redis.

Формат задания настройки — `list > string`.

Значение по умолчанию — `[]`.

12.19.3.7 sentinel > user

Параметр задает имя пользователя сервера Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.19.3.8 sentinel > password

Параметр задает пароль пользователя сервера Sentinel.

Формат задания настройки — `string`.

Значение по умолчанию не задано.

12.19.4 Группа параметров LUNA_MONITORING

Данная группа параметров задает настройки мониторинга.

См. подробную информацию о мониторинге в разделе «Мониторинг».

12.19.4.1 storage_type

Тип хранилища для хранения данных мониторинга.

В настоящее время доступно только БД Influx.

Формат задания настройки — `string`.

Значение по умолчанию — `influx`.

12.19.4.2 send_data_for_monitoring

Параметр позволяет включить или отключить отправку данных для мониторинга в InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.19.4.3 use_ssl

Параметр позволяет использовать HTTPS для подключения к InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.19.4.4 organization

Параметр задает рабочую область InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.19.4.5 token

Параметр задает аутентификации InfluxDB 2.x.

Формат задания настройки — `string`.

12.19.4.6 bucket

Параметр задаёт имя бакета InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_monitoring`.

12.19.4.7 host

Параметр задаёт IP-адрес InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.19.4.8 port

Параметр задает порт InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `8086`.

12.19.4.9 flushing_period

Параметр задает частоту отправки данных мониторинга в InfluxDB.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `1`.

12.19.5 Группа параметров LUNA_STREAMS_RETRANSLATOR_HTTP_SETTINGS

В данной группе параметров содержатся настройки, отвечающие за обработку HTTP-подключений. См. подробную информацию по следующей ссылке: https://sanic.dev/en/guide/deployment/configuration.html#builtin-values

12.19.5.1 request_timeout

Параметр задает продолжительность времени между моментом, когда новое открытое TCP-соединение передается на сервер, и моментом, когда получен весь HTTP-запрос.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.19.5.2 response_timeout

Параметр задает продолжительность времени между моментом, когда сервер передает HTTP-запрос приложению, и моментом, когда HTTP-ответ отправляется клиенту.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `600`.

12.19.5.3 request_max_size

Параметр задает максимальный размер запроса.

Формат задания настройки — `integer` (байты).

Значение по умолчанию — `1073741824`.

12.19.5.4 keep_alive_timeout

Параметр задает тайм-аут поддержания активности HTTP.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `15`.

12.19.6 Прочие

12.19.6.1 luna_streams_retraslator_ignored_restream_ttl

Параметр регулирует период бездействия, после которого ретрансляция будет остановлена, если поток не востребован (например, не имеет зрителей или запросов).

Формат задания настройки — `integer`.

Значение по умолчанию — `60` (секунды).#### luna_streams_retranslator_active_plugins {#luna_-streams_retranslator_active_plugins}

Параметр задает список плагинов, которые должен использовать сервис.

Имена задаются в следующем формате:

```
[
   "plugin_1",
   "plugin_2",
   "plugin_3"
]
```

> Список должен содержать имена файлов без расширения (.py).

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.20 Настройки сервиса Lambda

Данный раздел описывает параметры сервиса Lambda.

Настройку сервиса можно выполнить с помощью сервиса Configurator.

12.20.1 Группа параметров LUNA_CONFIGURATOR

Данная группа параметров задаёт настройки подключения сервиса Lambda к сервису Configurator.

Данная группа параметров не будет видна в пользовательском интерфейсе сервиса Configurator. Параметры можно изменить только в конфигурационном файле «config.conf», расположенном в директории «/srv/luna_lambda/configs/» соответствующего контейнера.

12.20.1.1 use_configurator

Параметр позволяет включить использование сервиса Configurator.

> С помощью сервиса Configurator упрощается настройка сервисов LP. Сервис сохраняет все необходимые конфигурации для всех сервисов LP в одном месте.

При отключенном параметре будут использованы настройки из файла «config.conf», расположенного в директории «/srv/luna_lambda/configs/» соответствующего контейнера.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.20.1.2 luna_configurator_origin

Параметр задает протокол, IP адрес и порт сервиса Configurator.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5070`.

12.20.1.3 luna_configurator_api

Параметр задает версию API сервиса Configurator. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.20.2 Группа параметров LUNA_LAMBDA_DB

В данной группе параметров задаются настройки подключения к базе данных сервиса Lambda.

12.20.2.1 db_type

Параметр задает тип базы данных. Доступны следующие типы:

- «postgres» — тип СУБД PostgreSQL
- «oracle» — тип СУБД Oracle

Формат задания настройки — `string`.

Значение по умолчанию — `postgres`.

12.20.2.2 db_host

Параметр задает имя сервера (хост) базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.20.2.3 db_port

Параметр задает порт базы данных.

Порт по умолчанию для типа «postgres»- `5432`.

Порт по умолчанию для типа «oracle» — `1521`.

Формат задания настройки — `string`.

Значение по умолчанию — `5432`.

12.20.2.4 db_user

Параметр задает имя пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.20.2.5 db_password

Параметр задает пароль пользователя базы данных.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.20.2.6 db_name

Параметр задает имя базы данных для типа «postgres» или имя SID для типа «oracle».

Формат задания настройки — `string`.

Значение по умолчанию — `luna_lambda`.

12.20.2.7 connection_pool_size

Параметр задает размер пула соединений к БД. Реальное количество подключений может быть больше значения данной настройки на 1.

При необходимости в настройке «max_connections» конфигурационного файла PostgreSQL можно задать максимальное количество одновременных подключений к серверу БД. См. раздел «Продвинутая настройка PostrgeSQL» для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `10`.

12.20.2.8 dsn

Параметр задает строку подключения DSN для соединения с БД.

> DSN — это строка подключения, которая идентифицирует и указывает на источник данных (базу данных), к которому нужно установить соединение.

В строке DSN могут задаваться такие настройки, как множественные хосты, аутентификационные данные, порт и другие параметры.

> Настройки зависят от типа БД. Множественные хосты поддерживаются только с PostgreSQL.

> По умолчанию параметр «dsn» не отображается во вкладке «Settings» в Configurator. Проверить список всех доступных параметров для группы настроек можно на вкладке «Limitations».

Пример:

```
{
"dsn": "luna:luna@postgres01:5432,postgres02:5432/luna_lambda?some_option=
    some_value"
"db_settings": {
     "connection_pool_size": 5
     }
}
```

Здесь:

- «luna:luna» — имя пользователя и пароль для подключения к PostgreSQL;
- «@postgres01:5432,postgres02:5432» — список хостов и портов, разделенных запятыми. Это означает, что сервис будет пытаться подключиться к первому хосту («postgres01») на порту 5432. Если это не удастся, она попытается подключиться ко второму хосту («postgres02») также на порту 5432;
- «/luna_lambda» — название базы данных;
- «?some_option=some_value» — опциональные параметры для подключения.

При необходимости можно комбинировать строку DSN и классические настройки, однако строка DSN является более приоритетной. Можно частично заполнить строку DSN (например, «postgres01,postgres02/luna_lambda»), и тогда недостающие параметры будут заполнены из значений параметров «db_host», «db_port», «db_name», «db_user» и «db_password».

При запуске сервис создаст пул подключений к одному из доступных хостов DSN. В случае возникновения проблем с установлением соединения после нескольких неудачных попыток, сервис снова попытается настроить пул подключений к любому из доступных хостов DSN.

12.20.3 Группа параметров LUNA_MONITORING

Данная группа параметров задает настройки мониторинга.

См. подробную информацию о мониторинге в разделе «Мониторинг».

12.20.3.1 storage_type

Тип хранилища для хранения данных мониторинга.

В настоящее время доступно только БД Influx.

Формат задания настройки — `string`.

Значение по умолчанию — `influx`.

12.20.3.2 send_data_for_monitoring

Параметр позволяет включить или отключить отправку данных для мониторинга в InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.20.3.3 use_ssl

Параметр позволяет использовать HTTPS для подключения к InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.20.3.4 organization

Параметр задает рабочую область InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.20.3.5 token

Параметр задает аутентификации InfluxDB 2.x.

Формат задания настройки — `string`.

12.20.3.6 bucket

Параметр задаёт имя бакета InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_monitoring`.

12.20.3.7 host

Параметр задаёт IP-адрес InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.20.3.8 port

Параметр задает порт InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `8086`.

12.20.3.9 flushing_period

Параметр задает частоту отправки данных мониторинга в InfluxDB.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `1`.

12.20.4 Группа параметров LUNA_LAMBDA_LOGGER

Данная группа параметров задает настройки логирования.

12.20.4.1 log_level

Параметр задает уровень отладочной печати, по приоритету: «ERROR», «WARNING», «INFO», «DEBUG».

Формат задания настройки — `string`.

Значение по умолчанию — `INFO`.

12.20.4.2 log_time

Параметр задает формат времени, используемый в записях лога. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.20.4.3 log_to_stdout

Параметр позволяет отправлять логи в стандартный вывод (stdout).

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`

12.20.4.4 log_to_file

Параметр позволяет сохранять логи в файл. Директория с файлами логов указывается в параметре «folder_with_logs».

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.20.4.5 folder_with_logs

Параметр задает директорию, в которой хранятся логи. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить параметр «log_to_file».

Формат задания настройки — `string`.

Значение по умолчанию — `./`

Пример:

```
  "folder_with_logs": "/srv/logs"
```

12.20.4.6 max_log_file_size

Параметр задает максимальный размер файла лога в МБ перед выполнением его ротации (0 — не использовать ротацию).

Для использования данного параметра требуется включить параметр «log_to_file».

> При необходимости можно настроить ротацию логов Docker. См. раздел «Настройка ротации логов Docker» в руководстве по установке LUNA PLATFORM.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024`

12.20.4.7 multiline_stack_trace

Параметр включает многострочную трассировку стека в логах. Когда параметр включен, информация о стеке вызовов записывается в логах так, что каждый фрейм стека помещается на отдельной строке, что улучшает читаемость. Если параметр выключен, информация о стеке вызовов записывается на одной строке, что может сделать логи менее удобными для анализа.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.20.4.8 format

Параметр определяет формат выводимых логов. Доступны следующие значения:

- «default» — стандартный формат вывода логов LUNA PLATFORM
- «json» — вывод логов в формате json
- «ecs» — вывод логов в формате ECS (Elastic Common Schema)

При использовании значения «ecs» будут использоваться следующие поля:

- «http.response.status_code» — содержит код состояния ответа HTTP (200, 404, 500 и т.д.);
- «http.response.execution_time» — содержит информацию о времени, затраченном на выполнение запроса и получение ответа;
- «http.request.method» — содержит метод HTTP-запроса (GET, POST, PUT и т.д.);
- «url.path» — содержит путь в URL-адресе запроса;
- «error.code» — содержит код ошибки, если запрос завершается с ошибкой.

Формат задания настройки — `string`.

Значение по умолчанию — `default`.

12.20.5 Группа параметров ADDITIONAL_SERVICES_USAGE

12.20.5.1 luna_events

Параметр задает возможность использования сервиса Events.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.20.5.2 luna_handlers

Параметр задает возможность использования сервиса Handlers.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.20.5.3 luna_sender

Параметр задает возможность использования сервиса Sender.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.20.5.4 luna_faces

Параметр задает возможность использования сервиса Faces.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.20.5.5 luna_matcher_proxy

Параметр задает возможность использования сервиса Python Matcher Proxy.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.20.5.6 luna_image_store

Параметр задает возможность использования сервиса Image Store.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.20.5.7 luna_lambda

Параметр задает возможность использования сервиса Lambda.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.20.5.8 luna_video_manager

Параметр задает возможность использования сервиса Video Manager.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.20.5.9 luna_video_agent

Параметр задает возможность использования сервиса Video Agent.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.20.5.10 luna_streams_retranslator

Параметр задает возможность использования сервиса Streams Retranslator.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.20.6 Группа параметров LUNA_LAMBDA_HTTP_SETTINGS

В данной группе параметров содержатся настройки, отвечающие за обработку HTTP-подключений. См. подробную информацию по следующей ссылке: https://sanic.dev/en/guide/deployment/configuration.html#builtin-values

12.20.6.1 request_timeout

Параметр задает продолжительность времени между моментом, когда новое открытое TCP-соединение передается на сервер, и моментом, когда получен весь HTTP-запрос.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.20.6.2 response_timeout

Параметр задает продолжительность времени между моментом, когда сервер передает HTTP-запрос приложению, и моментом, когда HTTP-ответ отправляется клиенту.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `600`.

12.20.6.3 request_max_size

Параметр задает максимальный размер запроса.

Формат задания настройки — `integer` (байты).

Значение по умолчанию — `1073741824`.

12.20.6.4 keep_alive_timeout

Параметр задает тайм-аут поддержания активности HTTP.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `15`.

12.20.7 Группа параметров LUNA_LAMBDA_BUILD_LIMITS

Данная группа параметров позволяет настроить ограничения на использование ресурсов (оперативной памяти и CPU) для задач сборки Docker-образа в сервисе Lambda.

12.20.7.1 cpu_limit

Параметр задает максимальное количество CPU, которое может быть выделено для задачи сборки. Значение указывается в милиCPU, где 1000 милиCPU эквивалентно одному CPU. Например, значение 2000 соответствует 2 CPU.

Формат задания настройки — `integer`.

Значение по умолчанию — `2000`.

12.20.7.2 ram_limit

Параметр задает максимальный объем оперативной памяти, который может быть выделен для задачи сборки. Значение указывается в гигабайтах.

Формат задания настройки — `integer`.

Значение по умолчанию — `4`.

12.20.7.3 cpu_request

Параметр задает минимальное количество CPU, которое будет зарезервировано для задачи сборки. Значение указывается в милиCPU, где 1000 милиCPU эквивалентно одному CPU. Например, значение 500 соответствует 0.5 CPU.

Формат задания настройки — `integer`.

Значение по умолчанию — `500`.

12.20.7.4 ram_request

Параметр задает минимальный объем оперативной памяти, который будет зарезервирован для задачи сборки. Значение указывается в гигабайтах.

Формат задания настройки — `integer`.

Значение по умолчанию — `0.5`.

12.20.8 Группа параметров LAMBDA_S3

Данная группа параметров содержит параметры для взаимодействия с S3-хранилищем, в котором должны храниться модифицированные архивы с lambda.

Для использования S3 нужно обязательно указать следующие параметры:

- «host»
- «aws_public_access_key»
- «aws_secret_access_key»
- «authorization_signature»
- «bucket»

12.20.8.1 host

Параметр задает URL-адрес, по которому будет установлено соединение с S3-хранилищем.

Формат задания настройки — `string`.

Значение по умолчанию — `http://localhost:7480`.

12.20.8.2 region

Параметр задает регион, который будет использоваться при взаимодействии с S3-хранилищем.

> Регион может влиять на доступность и производительность различных ресурсов в AWS S3.

Формат задания настройки — `string`.

Значение по умолчанию не указано.

12.20.8.3 aws_public_access_key

Параметр задает публичный ключ доступа, который используется для аутентификации при доступе к S3-хранилищу. Этот ключ предоставляется AWS и используется для идентификации клиента.

Формат задания настройки — `string`.

Значение по умолчанию не указано.

12.20.8.4 aws_secret_access_key

Параметр задает секретный ключа доступа, который совместно с публичным ключом обеспечивает аутентификацию при доступе к S3-хранилищу.

Формат задания настройки — `string`.

Значение по умолчанию не указано.

12.20.8.5 authorization_signature

Параметр определяет метод, используемый для создания подписи аутентификации при выполнении операций с S3.

Можно указать два значения:

- «s3v4» — использование алгоритма подписи AWS S3 Version 4
- «s3v2» — использование алгоритма подписи AWS S3 Version 2

Формат задания настройки — `string`.

Значение по умолчанию — `s3v4`.

12.20.8.6 request_timeout

Параметр задает максимальное время, в течение которого запрос к S3-хранилищу должен быть выполнен. Если запрос не завершится в пределах этого времени, он будет отменен.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.20.8.7 connect_timeout

Параметр задает максимальное время ожидания для установления соединения с S3-хранилищем. Если соединение не устанавливается в пределах этого времени, то оно будет считаться неудачным.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `30`.

12.20.8.8 verify_ssl

Параметр определяет, следует ли выполнять проверку SSL-сертификата при установлении защищенного (HTTPS) соединения с S3-хранилищем. Если значение равно `true`, то SSL-сертификат будет проверен. Если значение равно `false`, то проверка будет отключена, что может вести к проблемам с безопасностью.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.20.8.9 bucket

Параметр задает бакет для хранения модифицированных архивов с lambda.

Формат задания настройки — `string`.

Значение по умолчанию — `lambda_bucket`.

12.20.9 Группа параметров CLUSTER_CREDENTIALS

Данная группа параметров определяет настройки доступа к кластеру Kubernetes.

12.20.9.1 host

Параметр задает URL-адрес кластера Kubernetes.

Формат задания настройки — `string`.

Значение по умолчанию — `https://127.0.0.1:6443`.

12.20.9.2 token

Параметр задает токен доступа, который используется для аутентификации при подключении к кластеру Kubernetes.

Формат задания настройки — `string`.

Значение по умолчанию — `token`.

12.20.9.3 certificate_path

Параметр задает путь к файлу SSL-сертификата, который используется для защищенного (HTTPS) соединения с кластером. Значение по умолчанию `./cert.crt` указывает на относительный путь к файлу сертификата с именем «cert.crt» в текущем каталоге.

Формат задания настройки — `string`.

Значение по умолчанию — `./cert.crt`.

12.20.10 Группа параметров LUNA_LICENSES_ADDRESS

Данная группа параметров задает настройки подключения к сервису Licenses.

12.20.10.1 origin

Параметр задает протокол, IP адрес и порт сервиса Licenses.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Licenses, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Licenses.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5120`.

12.20.10.2 api_version

Параметр задает версию API сервиса Licenses. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.20.11 Группа параметров LUNA_SERVICE_METRICS

Данная группа параметров включает и настраивает сбор метрик в формате Prometheus.

См. подробную информацию в разделе «Мониторинг».

12.20.11.1 enabled

Параметр включает сбор метрик.

Если сбор метрик отключен, то запрос к ресурсу `/metrics` вернет соответствующее сообщение.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.20.11.2 metrics_format

Параметр задает формат метрик.

На данный момент поддерживается только формат Prometheus.

См. официальную документацию Prometheus для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `prometheus`.

12.20.11.3 extra_labels

Параметр задает пользовательские типы лейблов.

Формат задания настройки — `label_name=label_value`.

Значение по умолчанию не задано.

12.20.12 Прочие

12.20.12.1 luna_lambda_active_plugins

Параметр задает список плагинов, которые должен использовать сервис.

Имена задаются в следующем формате:

```
[
   "plugin_1",
   "plugin_2",
   "plugin_3"
]
```

> Список должен содержать имена файлов без расширения (.py).

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.20.12.2 storage_time

Параметр задает формат времени, используемый для записей в базе данных. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.20.12.3 cluster_location

Параметр задает местоположение кластера Kubernetes:

- «internal» — сервис Lambda работает в кластере Kubernetes и не требует других дополнительных настроек;
- «remote» — сервис Lambda работает с удаленным кластером Kubernetes и правильно определенными настройками «CLUSTER_CREDENTIALS» (хост, токен и сертификат);

- «local»- сервис Lambda работает там же, где запущен кластер Kubernetes.

В классическом варианте работы с сервисом Lambda предполагается использование параметра «internal».

Формат задания настройки — `string`.

Значение по умолчанию — `internal`.

12.20.12.4 lambda_registry

Параметр задает реестр Docker для хранения образов lambda. В реестре должны содержаться базовые образы `lpa-lambda-base` и `lpa-lambda-base-fsdk`, а также образ инструмента для сборки контейнеров `kaniko-executor`. Данные образы должны быть перенесены из реестра VisionLabs при начале работы с lambda. См. подробную информацию в руководстве по установке.

Требуется доступ на чтение и запись к этому реестру.

Если lambda запущена в кластере Kubernetes, необходимо предоставить доступ к этому реестру из кластера.

Формат задания настройки — `string`.

Значение по умолчанию не указано.

12.20.12.5 lambda_insecure_registries

Параметр задает список небезопасных реестров Docker, которые требуют дополнительных мер безопасности или внимания при взаимодействии с ними.

Формат задания настройки — `array > string`.

Значение по умолчанию не указано.

12.21 Настройки lambda

Данный раздел описывает параметры для каждой единицы lambda.

Настройку сервиса можно выполнить с помощью сервиса Configurator.

12.21.1 Группа параметров LUNA_LAMBDA_UNIT_LOGGER

Данная группа параметров задает настройки логирования.

12.21.1.1 log_level

Параметр задает уровень отладочной печати, по приоритету: «ERROR», «WARNING», «INFO», «DEBUG».

Формат задания настройки — `string`.

Значение по умолчанию — `INFO`.

12.21.1.2 log_time

Параметр задает формат времени, используемый в записях лога. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.21.1.3 log_to_stdout

Параметр позволяет отправлять логи в стандартный вывод (stdout).

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`

12.21.1.4 log_to_file

Параметр позволяет сохранять логи в файл. Директория с файлами логов указывается в параметре «folder_with_logs».

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.21.1.5 folder_with_logs

Параметр задает директорию, в которой хранятся логи. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить параметр «log_to_file».

Формат задания настройки — `string`.

Значение по умолчанию — `./`

Пример:

```
  "folder_with_logs": "/srv/logs"
```

12.21.1.6 max_log_file_size

Параметр задает максимальный размер файла лога в МБ перед выполнением его ротации (0 — не использовать ротацию).

Для использования данного параметра требуется включить параметр «log_to_file».

> При необходимости можно настроить ротацию логов Docker. См. раздел «Настройка ротации логов Docker» в руководстве по установке LUNA PLATFORM.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024`

12.21.1.7 multiline_stack_trace

Параметр включает многострочную трассировку стека в логах. Когда параметр включен, информация о стеке вызовов записывается в логах так, что каждый фрейм стека помещается на отдельной строке, что улучшает читаемость. Если параметр выключен, информация о стеке вызовов записывается на одной строке, что может сделать логи менее удобными для анализа.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.21.1.8 format

Параметр определяет формат выводимых логов. Доступны следующие значения:

- «default» — стандартный формат вывода логов LUNA PLATFORM
- «json» — вывод логов в формате json
- «ecs» — вывод логов в формате ECS (Elastic Common Schema)

При использовании значения «ecs» будут использоваться следующие поля:

- «http.response.status_code» — содержит код состояния ответа HTTP (200, 404, 500 и т.д.);
- «http.response.execution_time» — содержит информацию о времени, затраченном на выполнение запроса и получение ответа;
- «http.request.method» — содержит метод HTTP-запроса (GET, POST, PUT и т.д.);
- «url.path» — содержит путь в URL-адресе запроса;
- «error.code» — содержит код ошибки, если запрос завершается с ошибкой.

Формат задания настройки — `string`.

Значение по умолчанию — `default`.

12.21.2 Группа параметров LUNA_LAMBDA_UNIT_HTTP_SETTINGS

В данной группе параметров содержатся настройки, отвечающие за обработку HTTP-подключений. См. подробную информацию по следующей ссылке: https://sanic.dev/en/guide/deployment/configuration.html#builtin-values

12.21.2.1 request_timeout

Параметр задает продолжительность времени между моментом, когда новое открытое TCP-соединение передается на сервер, и моментом, когда получен весь HTTP-запрос.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.21.2.2 response_timeout

Параметр задает продолжительность времени между моментом, когда сервер передает HTTP-запрос приложению, и моментом, когда HTTP-ответ отправляется клиенту.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `600`.

12.21.2.3 request_max_size

Параметр задает максимальный размер запроса.

Формат задания настройки — `integer` (байты).

Значение по умолчанию — `1073741824`.

12.21.2.4 keep_alive_timeout

Параметр задает тайм-аут поддержания активности HTTP.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `15`.

12.21.3 Группа параметров LUNA_REMOTE_SDK_ADDRESS

Данная группа параметров задает настройки подключения к сервису Remote SDK.

12.21.3.1 origin

Параметр задает протокол, IP адрес и порт сервиса Remote SDK.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Remote SDK, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Remote SDK.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5220`.

12.21.3.2 api_version

Параметр задает версию API сервиса Remote SDK. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.21.4 Группа параметров LUNA_REMOTE_SDK_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Remote SDK.

12.21.4.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Remote SDK. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.21.4.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.21.4.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.21.4.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.21.5 Группа параметров LUNA_SENDER_ADDRESS

Данная группа параметров задает настройки подключения к сервису Sender.

12.21.5.1 origin

Параметр задает протокол, IP адрес и порт сервиса Sender.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Sender, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Sender.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5080`.

12.21.5.2 api_version

Параметр задает версию API сервиса Sender. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.21.6 Группа параметров LUNA_PYTHON_MATCHER_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Python Matcher.

12.21.6.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Python Matcher. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.21.6.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.21.6.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.21.6.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.21.7 Группа параметров LUNA_PYTHON_MATCHER_PROXY_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Python Matcher Proxy.

12.21.7.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Python Matcher Proxy. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.21.7.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.21.7.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.21.7.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.21.8 Группа параметров LUNA_PYTHON_MATCHER_ADDRESS

Данная группа параметров задает настройки подключения к сервису Python Matcher.

12.21.8.1 origin

Параметр задает протокол, IP адрес и порт сервиса Python Matcher.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Python Matcher, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Python Matcher.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5100`.

12.21.8.2 api_version

Параметр задает версию API сервиса Python Matcher. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.21.9 Группа параметров LUNA_MATCHER_PROXY_ADDRESS

Данная группа параметров задает настройки подключения к сервису Python Matcher Proxy.

12.21.9.1 origin

Параметр задает протокол, IP адрес и порт сервиса Python Matcher Proxy.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Python Matcher Proxy, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Python Matcher Proxy.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5110`.

12.21.9.2 api_version

Параметр задает версию API сервиса Python Matcher Proxy. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.21.10 Группа параметров LUNA_IMAGE_STORE_IMAGES_ADDRESS

В данной группе параметров задаются настройки бакета для хранения исходных изображений.

12.21.10.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.21.10.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.21.10.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-image-origin`.

12.21.11 Группа параметров LUNA_IMAGE_STORE_IMAGES_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к бакету, хранящему исходных изображений.

12.21.11.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к бакету, хранящему исходных изображений. Это время ожидания, в течение которого клиент пытается установить соединение с бакетом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `30`.

12.21.11.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.21.12 Группа параметров LUNA_IMAGE_STORE_FACES_SAMPLES_ADDRESS

В данной группе параметров задаются настройки бакета для хранения БО лиц.

12.21.12.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.21.12.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.21.12.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-samples`.

12.21.13 Группа параметров LUNA_IMAGE_STORE_FACES_SAMPLES_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к бакету, хранящему БО лиц.

12.21.13.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к бакету, хранящему БО лиц. Это время ожидания, в течение которого клиент пытается установить соединение с бакетом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `30`.

12.21.13.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.21.14 Группа параметров LUNA_IMAGE_STORE_BODIES_SAMPLES_ADDRESS

В данной группе параметров задаются настройки бакета для хранения БО тел.

12.21.14.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.21.14.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.21.14.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-bodies-samples`.

12.21.15 Группа параметров LUNA_IMAGE_STORE_BODIES_SAMPLES_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к бакету, хранящему БО тел.

12.21.15.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к бакету, хранящему БО тел. Это время ожидания, в течение которого клиент пытается установить соединение с бакетом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `30`.

12.21.15.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.21.16 Группа параметров LUNA_IMAGE_STORE_OBJECTS_ADDRESS

В данной группе параметров задаются настройки бакета для хранения объектов.

12.21.16.1 origin

Параметр задает протокол, IP адрес и порт сервиса Image Store.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Image Store, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Image Store.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5020`.

12.21.16.2 api_version

Параметр задает версию API сервиса Image Store. Доступная версия API — «1».

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.21.16.3 bucket

Параметр задает имя бакета.

См. подробное описание бакетов в разделе «Описание бакетов».

Формат задания настройки — `string`.

Значение по умолчанию — `visionlabs-objects`.

12.21.17 Группа параметров LUNA_FACES_ADDRESS

Данная группа параметров задает настройки подключения к сервису Faces.

12.21.17.1 origin

Параметр задает протокол, IP адрес и порт сервиса Faces.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Faces, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Faces.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5030`.

12.21.17.2 api_version

Параметр задает версию API сервиса Faces. Доступная версия API — «3».

Формат задания настройки — `integer`.

Значение по умолчанию — `3`.

12.21.18 Группа параметров LUNA_FACES_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Faces.

12.21.18.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Faces. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.21.18.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.21.18.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.21.18.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.21.19 Группа параметров LUNA_EVENTS_ADDRESS

Данная группа параметров задает настройки подключения к сервису Events.

12.21.19.1 origin

Параметр задает протокол, IP адрес и порт сервиса Events.

IP адрес «127.0.0.1» означает, что будет использоваться сервис Events, расположенный на сервере с Configurator. Если сервис находится на ином сервере, то в данном параметре нужно указать корректный IP адрес сервера с запущенным сервисом Events.

Формат задания настройки — `string`.

Значение по умолчанию — `http://127.0.0.1:5040`.

12.21.19.2 api_version

Параметр задает версию API сервиса Events. Доступная версия API — «2».

Формат задания настройки — `integer`.

Значение по умолчанию — `2`.

12.21.20 Группа параметров LUNA_EVENTS_TIMEOUTS

Данная группа параметров определяет временные интервалы для управления таймаутами HTTP-запросов, которые направляются к сервису Events.

12.21.20.1 connect

Параметр задает таймаут для установки соединения при отправке HTTP-запроса к сервису Events. Это время ожидания, в течение которого клиент пытается установить соединение с сервисом.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `20`.

12.21.20.2 request

Параметр задает общий таймаут для выполнения всего HTTP-запроса. Он включает в себя время установки соединения, отправки запроса, получения ответа и закрытия соединения. Если весь процесс занимает больше времени, чем указано в этом параметре, то запрос будет прерван.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.21.20.3 sock_connect

Параметр задает таймаут для установки соединения на уровне сокетов. Если соединение на уровне сокетов не устанавливается в установленное время, операция будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `10`.

12.21.20.4 sock_read

Параметр задает таймаут на чтение данных с сокета после успешного соединения. Если данные не поступают в установленное время, операция чтения будет прервана.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.21.21 Группа параметров LUNA_MONITORING

Данная группа параметров задает настройки мониторинга.

См. подробную информацию о мониторинге в разделе «Мониторинг».

12.21.21.1 storage_type

Тип хранилища для хранения данных мониторинга.

В настоящее время доступно только БД Influx.

Формат задания настройки — `string`.

Значение по умолчанию — `influx`.

12.21.21.2 send_data_for_monitoring

Параметр позволяет включить или отключить отправку данных для мониторинга в InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.21.21.3 use_ssl

Параметр позволяет использовать HTTPS для подключения к InfluxDB.

Формат задания настройки — `integer`.

Значение по умолчанию — `0`.

12.21.21.4 organization

Параметр задает рабочую область InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna`.

12.21.21.5 token

Параметр задает аутентификации InfluxDB 2.x.

Формат задания настройки — `string`.

12.21.21.6 bucket

Параметр задаёт имя бакета InfluxDB 2.x.

Формат задания настройки — `string`.

Значение по умолчанию — `luna_monitoring`.

12.21.21.7 host

Параметр задаёт IP-адрес InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `127.0.0.1`.

12.21.21.8 port

Параметр задает порт InfluxDB.

Формат задания настройки — `string`.

Значение по умолчанию — `8086`.

12.21.21.9 flushing_period

Параметр задает частоту отправки данных мониторинга в InfluxDB.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `1`.

12.21.22 Группа параметров ADDITIONAL_SERVICES_USAGE

12.21.22.1 luna_events

Параметр задает возможность использования сервиса Events.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.21.22.2 luna_handlers

Параметр задает возможность использования сервиса Handlers.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.21.22.3 luna_sender

Параметр задает возможность использования сервиса Sender.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.21.22.4 luna_faces

Параметр задает возможность использования сервиса Faces.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.21.22.5 luna_matcher_proxy

Параметр задает возможность использования сервиса Python Matcher Proxy.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.21.22.6 luna_image_store

Параметр задает возможность использования сервиса Image Store.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.21.22.7 luna_lambda

Параметр задает возможность использования сервиса Lambda.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.21.22.8 luna_video_manager

Параметр задает возможность использования сервиса Video Manager.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.21.22.9 luna_video_agent

Параметр задает возможность использования сервиса Video Agent.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.21.22.10 luna_streams_retranslator

Параметр задает возможность использования сервиса Streams Retranslator.

Включение/отключение данного сервиса может повлиять на работу других сервисов.

См. подробную информацию в разделе «Отключаемые сервисы».

В руководстве по установке приведен раздел «Использование необязательных сервисов», описывающий настройку использования необязательных сервисов перед запуском LUNA PLATFORM.

Формат задания настройки — `integer` («0» или «1»).

Значение по умолчанию — `1`.

12.21.23 Группа параметров LUNA_SERVICE_METRICS

Данная группа параметров включает и настраивает сбор метрик в формате Prometheus.

См. подробную информацию в разделе «Мониторинг».

12.21.23.1 enabled

Параметр включает сбор метрик.

Если сбор метрик отключен, то запрос к ресурсу `/metrics` вернет соответствующее сообщение.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.21.23.2 metrics_format

Параметр задает формат метрик.

На данный момент поддерживается только формат Prometheus.

См. официальную документацию Prometheus для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `prometheus`.

12.21.23.3 extra_labels

Параметр задает пользовательские типы лейблов.

Формат задания настройки — `label_name=label_value`.

Значение по умолчанию не задано.

12.21.24 Прочие

12.21.24.1 luna_lambda_unit_active_plugins

Параметр задает список плагинов, которые должен использовать сервис.

Имена задаются в следующем формате:

```
  [
     "plugin_1",
     "plugin_2",
     "plugin_3"
  ]
```

> Список должен содержать имена файлов без расширения (.py).

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.

12.22 Настройки Tasks-lambda

Данный раздел описывает параметры для каждой единицы Tasks-lambda. Большая часть настроек аналогична настройкам, описанным в разделе «Настройки сервиса Tasks». Индивидуальные настройки описаны ниже.

Настройку сервиса можно выполнить с помощью сервиса Configurator.

12.22.1 Группа параметров LUNA_LAMBDA_TASKS_UNIT_LOGGER

Данная группа параметров задает настройки логирования.

12.22.1.1 log_level

Параметр задает уровень отладочной печати, по приоритету: «ERROR», «WARNING», «INFO», «DEBUG».

Формат задания настройки — `string`.

Значение по умолчанию — `INFO`.

12.22.1.2 log_time

Параметр задает формат времени, используемый в записях лога. Доступны следующие значения:

- «LOCAL» — отображает местное время системы, на которой выполняется запись логов;
- «UTC» — отображает координированное всемирное время, которое является стандартом времени и не зависит от местной временной зоны или сезонных изменений времени.

Формат задания настройки — `string`.

Значение по умолчанию — `LOCAL`.

12.22.1.3 log_to_stdout

Параметр позволяет отправлять логи в стандартный вывод (stdout).

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`

12.22.1.4 log_to_file

Параметр позволяет сохранять логи в файл. Директория с файлами логов указывается в параметре «folder_with_logs».

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.22.1.5 folder_with_logs

Параметр задает директорию, в которой хранятся логи. Относительный путь начинается с каталога с приложением.

Для использования данного параметра требуется включить параметр «log_to_file».

Формат задания настройки — `string`.

Значение по умолчанию — `./`

Пример:

```
  "folder_with_logs": "/srv/logs"
```

12.22.1.6 max_log_file_size

Параметр задает максимальный размер файла лога в МБ перед выполнением его ротации (0 — не использовать ротацию).

Для использования данного параметра требуется включить параметр «log_to_file».

> При необходимости можно настроить ротацию логов Docker. См. раздел «Настройка ротации логов Docker» в руководстве по установке LUNA PLATFORM.

Формат задания настройки — `integer`.

Значение по умолчанию — `1024`

12.22.1.7 multiline_stack_trace

Параметр включает многострочную трассировку стека в логах. Когда параметр включен, информация о стеке вызовов записывается в логах так, что каждый фрейм стека помещается на отдельной строке, что улучшает читаемость. Если параметр выключен, информация о стеке вызовов записывается на одной строке, что может сделать логи менее удобными для анализа.

Формат задания настройки — `boolean`.

Значение по умолчанию — `true`.

12.22.1.8 format

Параметр определяет формат выводимых логов. Доступны следующие значения:

- «default» — стандартный формат вывода логов LUNA PLATFORM
- «json» — вывод логов в формате json
- «ecs» — вывод логов в формате ECS (Elastic Common Schema)

При использовании значения «ecs» будут использоваться следующие поля:

- «http.response.status_code» — содержит код состояния ответа HTTP (200, 404, 500 и т.д.);
- «http.response.execution_time» — содержит информацию о времени, затраченном на выполнение запроса и получение ответа;
- «http.request.method» — содержит метод HTTP-запроса (GET, POST, PUT и т.д.);
- «url.path» — содержит путь в URL-адресе запроса;
- «error.code» — содержит код ошибки, если запрос завершается с ошибкой.

Формат задания настройки — `string`.

Значение по умолчанию — `default`.

12.22.2 Группа параметров LUNA_LAMBDA_TASKS_UNIT_HTTP_SETTINGS

В данной группе параметров содержатся настройки, отвечающие за обработку HTTP-подключений. См. подробную информацию по следующей ссылке: https://sanic.dev/en/guide/deployment/configuration.html#builtin-values

12.22.2.1 request_timeout

Параметр задает продолжительность времени между моментом, когда новое открытое TCP-соединение передается на сервер, и моментом, когда получен весь HTTP-запрос.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `60`.

12.22.2.2 response_timeout

Параметр задает продолжительность времени между моментом, когда сервер передает HTTP-запрос приложению, и моментом, когда HTTP-ответ отправляется клиенту.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — `600`.

12.22.2.3 request_max_size

Параметр задает максимальный размер запроса.

Формат задания настройки — `integer` (байты).

Значение по умолчанию — `1073741824`.

12.22.2.4 keep_alive_timeout

Параметр задает тайм-аут поддержания активности HTTP.

Формат задания настройки — `integer` (секунды).

Значение по умолчанию — 15.

12.22.3 Группа параметров LUNA_SERVICE_METRICS

Данная группа параметров включает и настраивает сбор метрик в формате Prometheus.

См. подробную информацию в разделе «Мониторинг».

12.22.3.1 enabled

Параметр включает сбор метрик.

Если сбор метрик отключен, то запрос к ресурсу `/metrics` вернет соответствующее сообщение.

Формат задания настройки — `boolean`.

Значение по умолчанию — `false`.

12.22.3.2 metrics_format

Параметр задает формат метрик.

На данный момент поддерживается только формат Prometheus.

См. официальную документацию Prometheus для более подробной информации.

Формат задания настройки — `string`.

Значение по умолчанию — `prometheus`.

12.22.3.3 extra_labels

Параметр задает пользовательские типы лейблов.

Формат задания настройки — `label_name=label_value`.

Значение по умолчанию не задано.

12.22.4 Прочие

12.22.4.1 luna_lambda_tasks_unit_active_plugins

Параметр задает список плагинов, которые должен использовать сервис.

Имена задаются в следующем формате:

```
  [
     "plugin_1",
     "plugin_2",
     "plugin_3"
```

```
]
```

> Список должен содержать имена файлов без расширения (.py).

Формат задания настройки — `integer`.

Значение по умолчанию — `1`.