

VisionLabs LUNA PLATFORM 5

Краткое руководство пользователя

v.5.67.0

Содержание

Глоссарий	3
Введение	5
1 Структура комплекта поставки	6
2 Документация комплекта поставки	7
2.1 Руководства по эксплуатации	7
2.2 Руководства по установке	7
2.3 Спецификация OpenAPI	9
3 Начало работы	9
3.1 Создание запросов	10
3.1.1 Использование спецификации OpenAPI	11
4 Пример	14
4.1 Создание аккаунта	14
4.2 Создание обработчика	15
4.3 Генерация события	17

Глоссарий

Термин	Определение
Атрибуты	Базовые атрибуты и биометрический шаблон.
Аватар	Визуальное изображение лица, используемое в пользовательском интерфейсе.
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.
Детектор	Нейронная сеть, используемая для обнаружения одного из типов объектов (лиц, тел, лиц+тел) на исходном изображении.
Контрольные точки	Опорные точки на лице или теле, используемые алгоритмами распознавания для локализации лица или тела.
Лица	Изменяемые объекты, содержащие информацию о лице человека.
Обработчики	Изменяемые объекты, в которых хранятся правила для обработки изображений.
Ограничивающий прямоугольник	Прямоугольник, ограничивающий пространство изображения с обнаруженным лицом или телом.
События	Неизменяемые объекты, которые содержат информацию об одном лице и/или теле.
Сравнение, матчинг	Операция сопоставления биометрических шаблонов, хранящихся в базе данных.
Параметры лиц	Характеристики лица (эмоции, параметры рта, положение головы и т.д.), определяемые на исходном изображении во время детекции.
Параметры тел	Характеристики тела (наличие рюкзака, головной убор, цвет одежды и т.д.), определяемые на исходном изображении во время детекции.
Параметры изображений	Характеристики изображения (ширина и высота, соотношение сторон, размер и т.д.), определяемые на исходном изображении во время детекции.

Аббревиатура	Расшифровка
БО	Биометрический образец
БШ	Биометрический шаблон
БД, DB	База данных
LP	LUNA PLATFORM
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
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

Введение

В главе «[Структура комплекта поставки](#)» описывается содержимое комплекта поставки.

В главе «[Документация комплекта поставки](#)» перечислены все документы, входящие в комплект поставки.

Глава «[Начало работы](#)» поможет вам начать работу с LUNA PLATFORM.

В главе «[Пример](#)» приведен пример отправки запроса на распознавание лица в LUNA PLATFORM с подробным описанием тел запроса и ответа.

1 Структура комплекта поставки

Комплект поставки состоит из следующих директорий:

Имя директории	Описание
/example-docker	Директория включает в себя все файлы, необходимые для запуска Docker-контейнеров
/extras	Дополнительные зависимости и вспомогательные скрипты
/docs	Документация для LUNA PLATFORM

Директория «extras» содержит:

Имя директории	Описание
/conf	Конфигурационные файлы для сервисов LP и NGINX
/hasp	HASP утилиту и файлы, необходимые для активации лицензии
/utils	Утилиты для работы с LP
/VLMatch	Библиотеки и исходные файлы, необходимые для выполнения операций сравнения по БД с использованием сервиса Python Matcher

Директория «example-docker» содержит:

Имя директории	Описание
/luna_configurator	Конфигурации для сервиса Configurator
/postgresql	Скрипты для создания баз данных в PostgreSQL
/logging	Файлы для LUNA Dashboards, Grafana Loki и Promtail

2 Документация комплекта поставки

В данном разделе описывается пакет документации из комплекта поставки LUNA PLATFORM. Все документы находятся в папке «/docs» комплекта поставки.

2.1 Руководства по эксплуатации

Нижеперечисленные руководства описывают общие процессы LP, её архитектуру, интерфейс системные требования и примечания к выпуску.

Документы представлены в форматах PDF и HTML.

Документ	Описание
LP_Release_Notes_Rus	Примечания к выпуску.
LP_Administrator_Manual_Rus	Руководство администратора.
LP_User_Interface_Manual_Rus	Руководство по пользовательскому интерфейсу.
LP_System_Requirements_Rus	Системные требования.

2.2 Руководства по установке

Нижеперечисленные руководства описывают развертывание и обновление разными способами, а также активацию лицензии и использование утилиты Storages.

Руководства можно найти в каталоге «./docs/InstallationManuals».

Документы представлены в форматах PDF и HTML.

Документ	Описание
LP_Docker_Compose_Deployment_Example_Rus	Пример развертывания с помощью Docker Compose. Описывает установку на пустой сервер с нуля с помощью примера скрипта Docker Compose из комплекта поставки.
LP_Kubernetes_Helm_Deployment_Example_Rus	Пример развертывания в кластере Kubernetes. Описывает установку в кластере Kubernetes с использованием примеров Helm чартов из комплекта поставки.

Документ	Описание
LP_Installation_Manual_Using_Storages_Rus	Ручная установка с помощью утилиты Storages. Описывает подготовку окружения с помощью утилиты Storages и запуск Docker-контейнеров вручную на отдельном сервере.
LP_Upgrade_Manual_Using_Storages_Rus	Ручное обновление с помощью утилиты Storages. Описывает обновление окружения с помощью утилиты Storages и запуск новых Docker-контейнеров вручную на отдельном сервере.
LP_Installation_Manual_Rus	Ручная установка. Описывает подготовку окружения и запуск всех Docker-контейнеров вручную на отдельном сервере.
LP_Upgrade_Manual_Rus	Ручное обновление. Описывает обновление окружения и запуск новых Docker-контейнеров вручную на отдельном сервере.
LP_Migration_from_LP3_Rus	Ручная миграция с LP 3 до LP 5. Описывает миграцию окружения LP 3 и запуск Docker-контейнеров LP 5 вручную на отдельном сервере.
LP_Migration_from_LP4_Rus	Ручная миграция с LP 4 до LP 5. Описывает миграцию окружения LP 4 и запуск Docker-контейнеров LP 5 вручную на отдельном сервере.
LP_License_Activation_Manual_Rus	Руководство по активации лицензии.
LP_Storages_Utility_Manual_Rus	Руководство по утилите Storages.

Утилита Storages позволяет проверить и/или подготовить окружение для сервисов LUNA PLATFORM версии 5.46.1 и выше перед их непосредственным запуском. Рекомендуется использовать руководства по установке/обновлению с помощью утилиты Storages для версий 5.46.1 и выше.

2.3 Спецификация OpenAPI

Спецификация OpenAPI — единственный документ, предоставляющий актуальную информацию о API сервисов. Спецификация может использоваться:

- Инструментами создания документации для визуализации API.
- Инструментами генерации кода.

Все документы и код, созданные с использованием этой спецификации, могут содержать неточности, и их следует тщательно проверять.

В настоящее время на русском языке доступна только спецификация OpenAPI для сервиса API. Документация расположена в директории `"/docs/ReferenceManuals/APIReferenceManual.pdf/html`. Документ в формате YML содержит запросы ко всем сервисам LUNA PLATFORM. Их можно использовать для автоматической генерации запросов в инструментах для тестирования API, например, Postman (не описано в документации LUNA PLATFORM). Не гарантируется, что все запросы будут импортированы правильно, может потребоваться ручное редактирование. Документы в формате HTML используются для визуализации данных спецификации и могут быть не полными. Спецификацию OpenAPI можно получить с помощью запроса «get openapi documentation» к сервисам LP. Заголовок «Ассерпт» должен принимать значение «application/x-yaml».

3 Начало работы

Существует несколько полезных руководств для начала работы с LUNA PLATFORM.

Документ [«LP_Administrator_Manual_Rus»](#) включает в себя всю общую информацию о LUNA PLATFORM, а именно:

- терминологию,
- процесс обработки изображений,
- процесс работы с полученными данными,
- создаваемые объекты и выполняемые задачи,
- архитектуру и взаимодействие сервисов,
- структуры баз данных,
- описание настроек сервисов.

Перед работой с LUNA PLATFORM рекомендуется ознакомиться с разделом «Основные положения», чтобы понять общие принципы работы с LUNA PLATFORM.

Комплект поставки не включает Docker-контейнеры. Вам необходимо будет загрузить их из Интернета. См. документ [«LP_Installation_Manual»](#) для получения дополнительной информации.

После того, как LUNA PLATFORM будет запущена, откройте документ [«APIReferenceManual.html»](#), содержащий описание запросов к LUNA PLATFORM.

3.1 Создание запросов

LUNA PLATFORM не имеет пользовательского интерфейса по умолчанию. Для работы с системой необходимо отправлять запросы через API.

При необходимости можно использовать пользовательский интерфейс LUNA CLEMENTINE 2.0 (не входит в комплект поставки).

LUNA PLATFORM состоит из нескольких сервисов, взаимодействующих между собой. Основным интерфейсом для работы с LUNA PLATFORM является сервис API. Сервис предназначен для получения пользовательских запросов и их перенаправления в другие сервисы LP. Так, например, для обнаружения лица на изображении необходимо отправить запрос в сервис API, который перенаправит запрос в сервис Remote SDK, где будет выполнено обнаружение лица, а затем ответ от сервиса Remote SDK перенаправится в сервис API, где пользователь получит результат обнаружения.

При необходимости можно отправлять запросы напрямую в другие сервисы, однако данный способ не является рекомендуемым и предназначен только для определенных целей и опытных пользователей.

Запросы отправляются в сервис API с помощью его URL:

```
http://<API server IP-address>:<API port>/<API Version>/
```

Здесь:

- <API server IP-address> — IP-адрес, где развернут сервис API
- <API port> — порт, на котором развернут сервис API. Порт задается во время запуска контейнера (по умолчанию 5000)
- <API Version> — версия API (всегда 6)

Пример:

```
http://10.16.8.152:5000/6/
```

Запросы можно отправлять через CURL или с помощью инструментов для работы с API (например, Postman).

Почти все запросы, отправляемые в LP 5, требуют авторизации. В LUNA PLATFORM доступно три типа авторизации:

- **BasicAuth**. Авторизация по логину и паролю (задаются во время создания аккаунта);
- **BearerAuth**. Авторизация по JWT токenu (выдается после создания токена);
- **LunaAccountIdAuth**. Авторизация по заголовку «Luna-Account-Id», в котором указывается сгенерированный после создания аккаунта «account_id».

Авторизация **LunaAccountIdAuth** имеет наименьший приоритет по сравнению с другими способами и может быть отключена с помощью настройки «ALLOW_LUNA_ACCOUNT_AUTH_HEADER» в секции «OTHER» настроек сервиса API в Configurator (по умолчанию включена). В [спецификации OpenAPI](#) заголовок «Luna-Account-Id» помечен словом **Deprecated**.

Сервис Configurator содержит настройки всех сервисов.

Для того чтобы воспользоваться одним из видов авторизации, необходимо наличия аккаунта. Самым простым способом создания аккаунта является отправка POST запроса «create account» к сервису API. При создании аккаунта необходимо указать следующие данные: login (email), password и account type (тип аккаунта). Аккаунт создается на этапе установки после запуска сервиса API.

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

3.1.1 Использование спецификации OpenAPI

Спецификация включает в себя:

- Требуемые ресурсы и методы отправки запросов.
- Описание параметров запросов.
- Описание ответов.
- Примеры запросов и ответов.

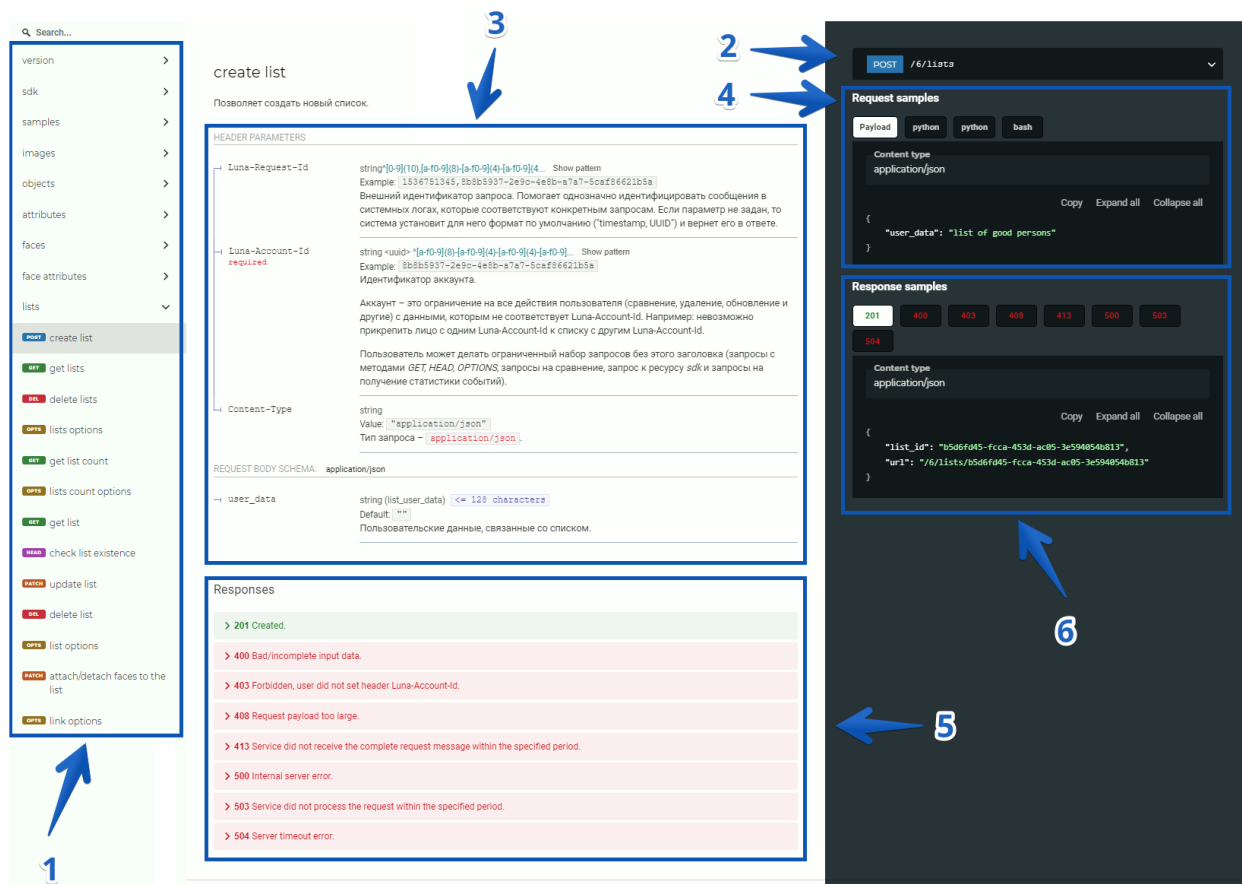
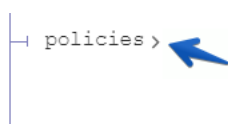


Рис. 1: Документация OpenAPI

Документ HTML включает в себя следующие элементы:

1. Запросы, поделенные на группы.
2. Метод запроса и пример URL. Для создания запроса следует использовать его с необходимым протоколом, IP-адресом и портом. Пример: POST http://<IP>:<PORT>/<Version>/matcher.
3. Описание параметров пути запроса, параметров запроса, параметров заголовка, схемы тела.
4. Пример тела запроса.
5. Описание ответов.
6. Примеры ответов.

Можно расширить описания параметров тела запроса или параметров ответа с помощью соответствующей иконки.



object (Policies)
Набор правил, определяющих обработку входных изображений. Пустые политики относятся к динамическому обработчику.

Рис. 2: Расширение описания

Можно выбрать требуемый пример тела запроса или ответа в соответствующих окнах.

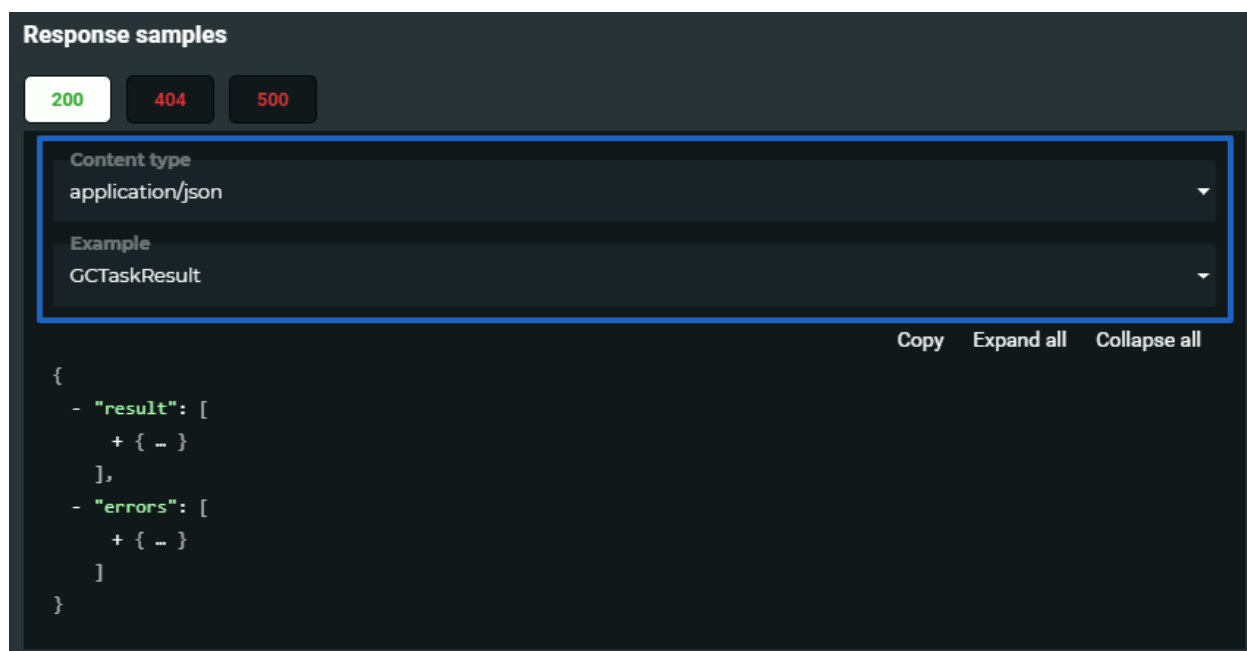


Рис. 3: Выбор примера

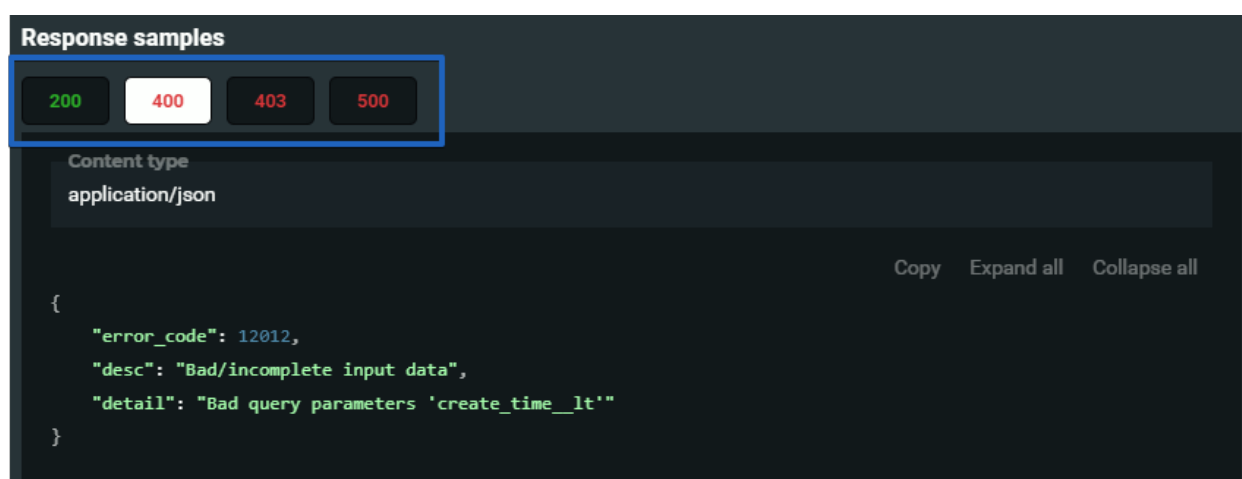


Рис. 4: Пример ответа

При задании фильтров для запросов необходимо использовать полное значение, если не указано обратное. Возможность использовать частичное значение отображается в описании.

4 Пример

В LUNA PLATFORM существует два основных подхода выполнения запросов.

Первый, и основной, подход заключается в задании всех правил обработки 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.

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.

В данном разделе приводится пример использования обработчиков и событий с использованием CURL-запросов с подробным описанием тел запроса и ответов.

См. подробную информацию о подходах в разделе «Подходы при работе» в руководстве администратора.

4.1 Создание аккаунта

Без аккаунта невозможно отправлять большинство запросов в LUNA PLATFORM. Аккаунт создается на этапе установки после запуска сервиса API. Создайте аккаунт с помощью следующей команды, если он еще не был создан:

```
curl --location --request POST 'http://127.0.0.1:5000/6/accounts' \
--header 'Content-Type: application/json' \
--data '{
  "login": "user@mail.com",
  "password": "password",
  "account_type": "user",
  "description": "description"
}'
```

Для использования авторизации BasicAuth в CURL-запросе, требуется конвертировать логин и пароль в формат Base64. Для конвертации требуется указать логин и пароль в формате `login:password`. В данном случае — `user:password`. Закодированные в Base64 логин и пароль выглядят следующим образом:

dXNlcjpwYXNzd29yZA==

4.2 Создание обработчика

В данном разделе приводится пример создания простого обработчика, используя CURL-запросы. С помощью данного обработчика можно будет:

- детектировать лицо
- определять базовые атрибуты (пол, возраст)
- извлекать его биометрический шаблон
- сохранять биометрический образец на диск (в бакет)
- сохранять лицо в базу данных Faces

Описание запроса: Запрос позволяет создать обработчик.

Тип запроса: POST

Запрос: http://127.0.0.1:5000/6/handlers

Запрос на создание обработчика выполняется с параметрами, указанными ниже:

Название параметра	Описание	Значение
Заголовки запроса		
Content-Type	Тип содержимого в запросе	application/json
Authorization	Basic-авторизация. Логин и пароль в формате Base64	dXNlcjpwYXNzd29yZA==
Тело запроса		См. ниже

Название параметра	Описание	Значение
	<p>Строка для ввода данных.</p> <p>Необходимо указать:</p> <ul style="list-style-type: none"> • description — описание обработчика • policies — политики обработчика • policies > detect_policy > detect_face — параметр определения лица • policies > extract_policy > extract_basic_attributes — параметр извлечения базовых атрибутов • policies > extract_policy > extract_face_descriptor — параметр извлечения биометрического шаблона • policies > storage_policy > face_policy > store_face — параметр сохранения лица в БД Faces • policies > handler_type — тип обработчика («0» — статический, «1» — динамический, «2» — lambda) 	

Пример CURL-запроса, для выполнения запроса из командной строки:

```
curl --location --request POST 'http://127.0.0.1:5000/6/handlers' \
--header 'Authorization: Basic dXNlcjpwYXNzd29yZA==' \
--header 'Content-Type: application/json' \
--data '{
  "description": "Simple handler",
  "policies": {
    "detect_policy": {
      "detect_face": 1
    },
    "extract_policy": {
      "extract_basic_attributes": 1,
      "extract_face_descriptor": 1
    },
    "storage_policy": {
      "face_sample_policy": {
        "store_sample": 1
      }
    },
    "face_policy": {
```



```

        "store_face": 1
    }
}
},
"is_dynamic": false
}'

```

При успешном выполнении запроса система возвращает идентификатор обработчика и его адрес.

Пример ответа на запрос:

```

{"handler_id": "f2831884-65b9-4b94-9639-f10f4d5f042d", "url": "\\6\\handlers\\f2831884-65b9-4b94-9639-f10f4d5f042d", "external_url": "http://127.0.0.1:5000/6/handlers/28e6358a-3753-4442-8310-617a8bba14bf"}

```

После создания обработчика, его идентификатор (параметр `handler_id`), идентификатор аккаунта (параметр `account_id`), время создания (параметр `create_time`), последнее время обновления (параметр `last_update_time`), описание (параметр `description`) и политики (группа параметров `policies`) записываются в таблицу `handler` базы данных `Handlers`.

На данном этапе больше ничего не сохраняется в какие-либо базы данных. Обработчик — простой набор правил, не способный создавать никакие объекты кроме себя. Для того, чтобы использовать эти правила на определенном изображении требуется сгенерировать событие.

4.3 Генерация события


Описание запроса: Запрос позволяет сгенерировать события и обработать их соответствующим обработчиком.

Тип запроса: POST

Запрос: `http://127.0.0.1:5000/6/handlers/{handler_id}/events`

Запрос на генерацию события выполняется с параметрами, указанными ниже:

Название параметра	Описание	Значение
Параметр запроса		
<code>handler_id</code>	Идентификатор созданного обработчика	<code>f2831884-65b9-4b94-9639-f10f4d5f042d</code>
Заголовки запроса		
<code>Content-Type</code>	Тип содержимого в запросе	<code>image/jpeg</code>

Название параметра	Описание	Значение
Authorization	Basic-авторизация. Логин и пароль в формате Base64	dXNlcjpwYXNzd29yZA==
Тело запроса		
	Изображение в формате jpeg, перенесенное на сервер, где развернута LP:	@/root/example.jpg
		

Пример CURL-запроса, для выполнения запроса из командной строки:

```
curl --location --request POST 'http://127.0.0.1:5000/6/handlers/f2831884-65b9-4b94-9639-f10f4d5f042d/events' \
--header 'Authorization: Basic dXNlcjpwYXNzd29yZA==' \
--header 'Content-Type: image/jpeg' \
--data-binary '@/root/example.jpg'
```

При необходимости можно указать изображение в формате Base64 без переноса на сервер с LP.

При успешном выполнении запроса система создает событие.

Пример ответа на запрос, отсортированный по блокам:

1) Блок «images» определяет общие данные об изображении. Если изображение было обработано unsuccessfully, то вернется статус «0» и код ошибки с описанием и ссылкой. Параметр «exif» содержит информацию о метаданных обрабатываемого изображения.

```
"images": [
  {
    "filename": "raw image",
    "status": 1,
    "error": {
      "error_code": 0,
```

```

        "desc": "Success",
        "detail": "Success",
        "link": "https://docs.visionlabs.ai/info/luna/troubleshooting/errors-description/code-0"
    },
    "exif": {}
}
],

```

2) Блок «events», определяющий всю информацию о сгенерированном событии.

```

"events": [
    ...
]

```

2.1) Подблок «face_attributes» определяет атрибуты, извлеченные с помощью политики обработчика «extract_policy», а именно, базовые атрибуты (параметр обработчика «extract_basic_attributes») и качество биометрического шаблона (параметр обработчика «extract_face_descriptor»).

Сам биометрический шаблон в ответе на запрос не выдается. См. раздел «Форматы биометрических шаблонов» в руководстве администратора.

Обратите внимание, что параметр «attribute_id» и его «url» равны «null». Это говорит о том, что атрибуты не были сохранены в базу данных Redis, поскольку не включен параметр «store_attribute» у политики «storage_policy» > «attribute_policy». Однако, в момент создания объекта «лицо», к нему привязываются атрибуты. Поскольку мы создаем лицо и сохраняем его в базу данных Faces (параметр обработчика «store_face»), то в базе данных Faces можно будет увидеть лицо с привязанными атрибутами. То есть атрибуты выдаются в ответе и сохраняются в базу данных Faces.

Поскольку включен параметр обработчика «store_sample» политики «face_sample_policy», то будет создаваться объект «биометрический образец» и сохраняться в бакет (сервис Image Store содержит ссылку на этот бакет).

Параметр обработчика «store_sample» политики «face_sample_policy» всегда включен по умолчанию. Мы явно его указали при создании обработчика. Если вы удалите его из тела запроса, то биометрический образец все равно будет создаваться. Нужно выставить значение параметра на 0.

В поле «samples» блока «face_attributes» указывается идентификатор биометрического образца, по которому были извлечены атрибуты. Полная информация по биометрическому образцу приводится ниже в подблоке «detections».

```

{
    "face_attributes": {

```

```

    "attribute_id": null,
    "url": null,
    "samples": [
        "b92b05cd-c871-4533-89ce-0a538f8227d3"
    ],
    "basic_attributes": {
        "ethnicities": {
            "predominant_ethnicity": "african_american",
            "estimations": {
                "asian": 5.721812167271785e-15,
                "indian": 1.3687103486030773e-16,
                "caucasian": 3.082826702249797e-11,
                "african_american": 1.0
            }
        },
        "age": 25,
        "gender": 0
    },
    "score": 0.8616658210754395
},
}

```

2.2) Подблок «detections» определяет список детекций, обнаруженных на изображении с помощью политики «detect_policy». Здесь указываются координаты ограничивающего прямоугольника лица («rect»), пять ключевых точек лица («landmarks_5»), адрес («url») к сервису Image Store (который содержит адрес до бакета «visionlabs-samples» с биометрическим образцом) и идентификатор биометрического образца («sample_id»). Поле «image_origin» имеет значение «null» потому что не была включена политика сохранения исходного изображения — «storage_policy» > «image_origin_policy». Если бы она была включена, то исходное изображение сохранилось бы в сервис Image Store как ссылка на бакет «visionlabs-image-origin». Поле «body» имеет значение «null» потому что не был включен параметр обработчика «detect_body».

```

"detections": [
    {
        "filename": "raw image",
        "samples": {
            "face": {
                "detection": {
                    "rect": {
                        "x": 103,
                        "y": 74,
                        "width": 195,
                        "height": 275
                    }
                }
            }
        }
    }
]

```

```

    },
    "landmarks5": [
      [
        29,
        129
      ],
      [
        123,
        90
      ],
      [
        91,
        148
      ],
      [
        51,
        211
      ],
      [
        160,
        171
      ]
    ]
  },
  "url": "\\6\\samples\\faces\\b92b05cd-c871-4533-89ce-0a538f8227d3",
  "sample_id": "b92b05cd-c871-4533-89ce-0a538f8227d3"
},
"body": null
},
"detect_time": "2022-11-28T17:01:09.572910+03:00",
"image_origin": null,
"detect_ts": null
}
],

```

2.3) Подблок «aggregate_estimations» отвечает за общие параметры лица или тела, полученного с разных изображений. Объединение нескольких параметров (например, возраст и пол) одного лица в один параметр называется **агрегацией**. Данный подблок пустой, поскольку в запросе на генерацию события не был указан параметр запроса «aggregate_attributes».

```

"aggregate_estimations": {
  "face": {
    "attributes": {}
  }
}

```

```

    },
    "body": {
        "attributes": {}
    }
},

```

2.4) Подблок «face» определяет объект «лицо», полученный в результате генерации события. Если бы параметр обработчика «store_face» политики «storage_policy» > «face_policy» был бы отключен, то поле «face» было бы равно значению «null», т.е. оно не сохранилось бы в базу данных Faces. Параметр «avatar» указывает адрес сохраненного биометрического образца, который будет использован в качестве аватара для лица. Обратите внимание, что за включение сохранения адреса до аватара отвечает параметр обработчика «set_sample_as_avatar» из политики «storage_policy» > «face_policy». Мы не указывали данный параметр при создании обработчика, но он включен по умолчанию. Поле «lists» пустое поскольку мы не указывали политику привязки лица к списку («link_to_lists_policy»). В поле «event_id» указан идентификатор события, который также сохраняется в базу данных Faces. Поля «external_id» и «user_data» пустые поскольку не были указаны в запросе на генерацию события.

```

"face": {
    "external_id": "",
    "face_id": "854f4b56-9a77-4be3-bc6e-797b8f3319ad",
    "user_data": "",
    "url": "\\6\\faces\\854f4b56-9a77-4be3-bc6e-797b8f3319ad",
    "lists": [],
    "avatar": "\\6\\samples\\faces\\b92b05cd-c871-4533-89ce-0a538f8227d3",
    "event_id": "5aaa902f-7085-4dc0-810a-09d170e35c0b"
}

```

2.5) Подблок «filtered_detections» будет иметь содержимое если в фильтрах политики «match_policy» будет указан какой-либо фильтр и изображение не будет проходить по указанным условиям.

```

"filtered_detections": {
    "face_detections": []
}

```

2.6) Прочие параметры. Параметры «location», «user_data», «external_id», «track_id», «tags», «source» задаются в параметрах запроса генерации события (см. раздел «Объект»Событие» руководства администратора). Поле «matches» заполняется при использовании политики «match_policy». Параметр «body_attributes» будет заполняться если будут включены параметры из группы параметров «body_attributes» политики «detect_policy».