



Программный продукт «Helmetnet»

Инструкция по эксплуатации



Оглавление

Введение.....	3
Эксплуатация программного продукта «Helmetnet»	4
Проверка состояния сервиса	4
Выполнение детектирования	4
Управление лицензией	5
Журналирование и мониторинг	5
Устранение типовых неполадок	5
Завершение работы	6



Введение

Программный продукт «Helmetnet» представляет собой серверное приложение (микросервис) для автоматического обнаружения средств индивидуальной защиты — защитных касок — на цифровых изображениях. Продукт принимает кадры от внешних информационных систем заказчика через REST API, выполняет нейросетевое детектирование людей и защитных касок и возвращает структурированный результат в формате JSON.

Программный продукт эксплуатируется в инфраструктуре заказчика и предназначен для интеграции со смежными информационными системами (платформами видеонаблюдения, системами контроля охраны труда). Настоящий документ содержит инструкцию по эксплуатации для администраторов и специалистов заказчика, осуществляющих интеграцию.

Предполагается, что программный продукт установлен и активирован в соответствии с Инструкцией по установке.

Эксплуатация программного продукта «Helmetnet»

С точки зрения пользователя программный продукт «Helmetnet» представляет собой веб-сервис, доступный по протоколу HTTP по адресу `http://<адрес_сервера>:<порт>` (порт по умолчанию — 5656). Все операции выполняются путём обращения к программным интерфейсам (REST API); все ответы сервиса возвращаются в формате JSON. Управление лицензией дополнительно доступно через встроенную веб-страницу.

Проверка состояния сервиса

Для контроля работоспособности сервиса предусмотрены следующие API-методы:

- GET /health — проверка доступности сервиса (liveness/readiness probe). При штатной работе сервис возвращает ответ `{"status": "ok"}` с кодом HTTP 200. Метод рекомендуется использовать в системах мониторинга и оркестрации заказчика.
- GET /status — получение служебной информации: версия сервиса, статус подключения к серверу инференса (Triton Inference Server), время работы (аптайм).

Пример проверки состояния из командной строки: `curl http://<адрес_сервера>:5656/health`

Выполнение детектирования

Основной рабочей операцией программного продукта является детектирование средств индивидуальной защиты на изображении. Операция выполняется методом POST /cvpredict.

1. Сформируйте HTTP-запрос POST по адресу `http://<адрес_сервера>:5656/cvpredict` с заголовком `Content-Type: application/json`.
2. В теле запроса передайте изображение в кодировке base64: `{"image": "<base64_строка>"}`.
3. Отправьте запрос и получите ответ сервиса.

При успешной обработке (HTTP 200) сервис возвращает массив детекций, где каждая детекция представлена списком [класс, вероятность, координаты]: класс объекта (например, «helmet» — каска, «person» — человек), вероятность (число от 0 до 1) и координаты ограничивающей рамки [xmin, ymin, xmax, ymax] в пикселях.

Пример ответа: `{"detections": [{"helmet", 0.95, [100, 150, 200, 250]}, {"person", 0.88, [50, 100, 300, 400]}]}`

Возможные ошибки:

- HTTP 400 — некорректный формат запроса или повреждённое изображение; проверьте корректность base64-строки и структуру JSON.
- HTTP 403 — лицензия недействительна, истёк срок её действия или исчерпан лимит запросов; выполните активацию новой лицензии (см. раздел «Управление лицензией»).

Пороговые значения детектирования (минимальная уверенность, порог IoU, размер входного изображения) задаются администратором через переменные окружения при установке.

Альтернативный способ детектирования — эндпоинт POST /: изображение передаётся файлом (multipart/form-data, поле frame) без кодирования в base64. Ответ содержит поля predicts (массив детекций), version (версия сервиса) и time (время обработки запроса).



При включённом режиме трекинга (переменная окружения IS_TRACKER) эндпоинт POST / дополнительно требует query-параметр cam_id (идентификатор камеры) и возвращает результаты трекинга объектов между кадрами с устойчивыми идентификаторами.

Управление лицензией

Доступ к операции детектирования ограничивается срочной (time-based) лицензией. Для управления лицензией предусмотрены следующие средства:

- GET /license/status — возвращает текущий статус лицензии: активна/неактивна, дата окончания действия (valid_until), количество оставшихся дней и, при наличии ограничения, остаток лимита запросов.
- POST /license/activate — активация лицензии; в теле запроса передаётся лицензионный ключ: {"license_key": "<ключ>"}. В случае успешной валидации ключа лицензия применяется без перезапуска сервиса.
- GET /license/ui — встроенная веб-страница для просмотра статуса лицензии и ввода нового ключа активации без использования API-клиентов.

Для активации лицензии через веб-страницу выполните следующие шаги:

4. Откройте в веб-браузере адрес http://<адрес_сервера>:5656/license/ui.
5. Ознакомьтесь с текущим статусом лицензии, отображаемым на странице.
6. Введите лицензионный ключ, полученный от правообладателя, в поле активации и подтвердите ввод.
7. Убедитесь, что статус лицензии изменился на «активна», а дата окончания действия соответствует приобретённой лицензии.

Журналирование и мониторинг

Сервис ведёт журнал обработки запросов. Уровень детализации журналирования задаётся переменной окружения LOGLEVEL (значения: DEBUG, INFO, WARNING, ERROR).

Просмотр журнала работы сервиса выполняется штатными средствами Docker: `docker logs <имя_контейнера>`. Для постоянного мониторинга рекомендуется настроить периодический опрос метода GET /health средствами системы мониторинга заказчика.

Устранение типовых неполадок

- Сервис не отвечает на запросы — убедитесь, что контейнеры запущены (`docker ps`), и изучите журнал (`docker logs`); при необходимости перезапустите сервис командой `docker-compose restart`.
- Метод /cvpredict возвращает HTTP 403 — проверьте статус лицензии методом GET /license/status и при необходимости активируйте новый лицензионный ключ.
- В ответе GET /status отсутствует подключение к серверу инференса — проверьте, что контейнер Triton Inference Server запущен, и что адрес в переменной TRITON_URL указан верно.
- Низкая скорость обработки — убедитесь, что вычисления выполняются на GPU (переменная DEVICE=cuda) и что GPU доступен контейнеру.



Завершение работы

Остановка программного продукта выполняется командой `docker-compose down` на сервере, где развёрнут продукт. Настройки, файлы лицензии и модели при этом сохраняются, и после повторного запуска сервис продолжает работу без дополнительной настройки.