|  |  |
| --- | --- |
|  |  |

Программный комплекс для ЭВМ

ВИДЕОДЕТЕКТОР ДЛЯ ПОИСКА СИЛУЭТОВ ЛЮДЕЙ В КАДРЕ

|  |  |
| --- | --- |
| ***Подпись и дата*** |  |
| ***Инв. № дубл.*** |  |
| ***Взам. инв. №*** |  |
| ***Подпись и дата*** |  |
| ***Инв. № подл.*** |  |

РУКОВОДСТВО

**14161592-ХХХХХХ.ИЗ-002**

Листов 26

|  |  |
| --- | --- |
| Согласовано:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_)«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. | Согласовано:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_)«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |

2022

Содержание

1 Введение 3

1.1 О документе 3

1.2 Общие сведения о ПО 3

1.3 Термины, определения и сокращения 3

2 Руководство администратора 5

2.1 Подготовка к работе 5

2.1.1 Системные требования 5

2.1.2 Архитектура и взаимодействие с внешними системами 5

2.1.2.1 Архитектура ПО во взаимосвязи с компонентами Системы 5

2.1.2.2 Краткое описание компонентов 5

2.1.2.3 Особенности технической реализации ПО 6

2.1.3 Развертывание и установка ПО 6

2.1.4 Особенности работы ПО в Kubernetes 6

2.2 Настройка ПО 7

2.2.1 Настройка взаимодействия ПО с другими системами через общую шину данных 7

2.2.1.1 Общие сведения 7

2.2.1.2 Формат сообщений, отправляемых в KAFKA 7

2.3 Основные действия и приемы при эксплуатации ПО 9

2.4 Обслуживание и устранение неисправностей 9

2.4.1 Мероприятия по обеспечению защиты информации 9

2.4.2 Ограничение доступа к ПО 9

2.4.2.1 Формирование матрицы доступа и централизованное администрирование списка пользователей 9

2.4.2.2 Журналирование действий пользователей на уровне операционной системы 9

2.4.2.3 Идентификация смежных систем 10

2.4.3 Резервное копирование и восстановление баз данных ПО 10

2.4.4 Просмотр логов 10

2.4.5 Рекомендации по периодическому обслуживанию компонентов ПО 11

2.4.6 Мониторинг диагностических данных о состоянии ПО и комплекса технических средств среды эксплуатации. Получение и обработка диагностических данных программными средствами 11

2.4.6.1 Диагностика ПО 11

2.4.6.2 Получение и обработка диагностических данных программными средствами, поддерживающими стандарт OpenMetrics 15

2.4.7 Действия персонала при сбое и аварийном режиме 17

3 Руководство 18

3.1 Интерфейс прикладного программирования ПО и методы работы с ним 18

3.1.1 Основные сведения 18

3.1.1.1 Точка доступа 18

3.1.1.2 Объект типа job (задание) 18

3.1.1.3 Сообщения об ошибках 19

3.1.2 Методы для управления видеодетекцией силуэтов 19

3.1.2.1 Создание задания 19

3.1.2.2 Получение списка всех заданий 20

3.1.2.3 Получение задания по id 21

3.1.2.4 Удаление задания по id 22

3.1.2.5 Изменение задания 23

3.1.2.6 Перезапуск задания 24

# Введение

## О документе

Настоящий документ представляет собой Руководство Программного комплекса «Видеодетектор для поиска силуэтов людей в кадре» (далее ПО) производства компании ООО «НТех лаб», предназначенного для включения в состав Единой системы видеоаналитики г. Москвы (далее Система).

Руководство:

1. Руководство администратора. В главе дублируются сведения из одноименного документа.
2. Руководство. В главе приведено описание интерфейса прикладного программирования ПО и методы работы с ним.

## Общие сведения о ПО

ПО предназначено для решения следующих задач:

1. получение и анализ видеопотоков в режиме реального времени от камер городского сегмента Системы, и архивных видеоданных;
2. детектирование (определение) силуэтов людей на видеопотоках от камер и в архивных видеоданных;
3. обеспечение Системы массивом данных для последующего анализа.

ПО предоставляет услуги детектирования силуэтов людей на видеоизображениях на основе нейронных сетей. До развертывания ПО и настройки его интеграции с внешними системами рекомендуется ознакомиться со структурными схемами взаимодействия ПО с внешними системами и архитектурой внутреннего взаимодействия его компонентов.

## Термины, определения и сокращения

В настоящем документе применяют следующие термины и сокращения с соответствующими определениями (см. Таблица 1).

**Таблица 1. Термины и сокращения**

| Сокращение | Расшифровка |
| --- | --- |
| БД | База данных |
| Модуль хранения силуэтов | Модуль хранения и обработки данных силуэтов людей. Программное обеспечение, предназначенное для хранения данных о событиях, поступающих от Модуля детектирования силуэтов, построенного на базе описываемого ПО. |
| ПО | Программный комплекс для ЭВМ «Видеодетектор для поиска силуэтов людей в кадре» |
| API | Application programming interface – Интерфейс программирования приложений. |
| Bounding Box, BBox  | Прямоугольник, ограничивающий форму более сложной геометрической модели (в данном случае силуэта) |
| HTTP API | Программный интерфейс, основанный на стандарте HTTP |
| KAFKA | Распределённый программный брокер сообщений. Используется для приема данных о силуэтах Модулем хранения силуэтов |

# Руководство администратора

## Подготовка к работе

### Системные требования

К ПО предъявляются следующие системные требования:

* программное обеспечение функционирует на ОС версии Ubuntu 18.04 и выше;
* программа стабильно функционирует с выполнением всех характеристик на следующих вычислительных ресурсах:
* на обработку 1008 потоков видеоданных с разрешением 1280x720 пикселей, частотой 15 кадров в секунду, с интенсивностью движения типа камер подъездного видеонаблюдения требуется 1 сервер с характеристиками не ниже:
	+ Графический ускоритель: 16 x NVIDIA Tesla T4
	+ Процессор: 2 x Intel(R) Xeon(R) @ 2.20GHz
	+ Оперативная память: 24 x 8 Гб DDR4
	+ Внутренняя подсистема хранения данных: 2 x 480G SSD.

### Архитектура и взаимодействие с внешними системами

#### Архитектура ПО во взаимосвязи с компонентами Системы

#### Краткое описание компонентов

ПО включает в себя следующие компоненты:

1. **VideoDetector**. Распознает силуэты в видеопотоке и отправляет соответствующие видеокадры и метаданные через компонент FaceSender в общую шину данных для дальнейшей обработки.
2. **VideoManager**. Предназначен для управления и планирования заданий детектирования силуэтов в видеопотоке. Через этот компонент происходит сбор статистических данных по обработке заданий.
3. **FaceSender**. Принимает кадры c силуэтами от компонента VideoDetector и отправляет их в брокер сообщений KAFKA Модуля хранения силуэтов.
4. **CCTV API**. Прокси-сервис, разрешающий адреса видеокамер.
5. **ArchiveUpdater**. Обновляет статистику по обработке видеоархивов.
6. **Внутренняя база данных ПО**. Стороннее программное обеспечение ETCD, реализующее распределенное хранилище ключей компонента **VideoManager**. Используется в качестве координационной службы в распределенной системе, обеспечивая отказоустойчивость работы ПО при линейном масштабировании в зависимости от количества подключаемых видеопотоков.
7. **StatDB**. База данных, в которой хранятся статусы обработки архивов. Реализована на стороннем программном обеспечении MongoDB. Обеспечивает хранение информации о поставленных на индексирование задачах.

#### Особенности технической реализации ПО

ПО развертывается и функционирует в системе автоматизации разворачивания, масштабирования и управления контейнеризированными приложениями Kubernetes.

### Развертывание и установка ПО

Для развертывания ПО выполните следующие действия:

1. Установите стороннее ПО Kubernetes, руководствуясь официальной справочной документацией.
2. На серверы с GPU установите стороннее ПО docker-nvidia, руководствуясь официальной справочной документацией.
3. Установите на операторскую машину стороннее ПО helm, руководствуясь официальной справочной документацией.
4. Установите docker registry, руководствуясь официальной справочной документацией.
5. Загрузите комплект docker image из поставляемого архива tar в registry. Пример команды:

docker load -i video-detector.tar
docker tag video-detector registry\_ip:registry\_port/video-detector:latest
docker push registry\_ip:registry\_port/video-detector:latest

1. Отредактируйте файл с образцом конфигурации ПО values.yaml, указав адрес registry в параметре image компонентов.
2. Находясь в директории helm\_charts в поставке вместе с tar-архивом дистрибутива ПО, выполните команду для установки файла с образцом конфигурации ПО values.yaml.

helm install r1 -f values.yaml .

### Особенности работы ПО в Kubernetes

Структурные компоненты ПО запускаются в pod-ах Kubernetes.

Имя pod-а имеет следующий формат:

* для pod, которые развернуты через deploy<префикс>-<название deploy>-<хеш версии deploy>-<идентификатор pod-а внутри deployment>;
* для pod, развёрнутых внутри statefulset: <префикс>-<название statefulset>-<номер pod внутри statefulset>

Соответствие внутренних имен процессов структурным компонентам ПО представлено в Таблица 2.

**Таблица 2. Соответствие имен процессов в Kubernetes именам структурных компонентов ПО**

|  |  |
| --- | --- |
| **Внутреннее имя процесса** | **Структурный компонент ПО** |
| archive-scheduler | AcrhiveUpdater |
| cctv-redirector | CCTV API |
| etcd | Внутренняя база данных ПО на основе etcd |
| face-sender | FaceSender |
| mcrouter | Внутренний процесс ПО. |
| memcached | Внутренний процесс ПО. |
| mongo | StatDB |
| video-manager | VideoManager |
| video-worker | VideoDetector |

## Настройка ПО

### Настройка взаимодействия ПО с другими системами через общую шину данных

#### Общие сведения

Взаимодействие ПО с другими системами выполняется через компонент FaceSender.

Компонент FaceSender выполняет следующие задачи:

* принимает силуэты от компонента VideoDetector,
* перенаправляет полученные силуэты в Kafka Модуля ОБХД,
* обновляет статистические данные в StatDB по индексированию архивных видео, необходимые для диагностики ПО.

#### Формат сообщений, отправляемых в KAFKA

Компонент FaceSender отправляет в KAFKA сериализованное в msgpack сообщение.

Вывод `msgpack2json`:

```javascript

{

 "request\_id": "VW:c09be3df",

 "labels": {}, // метки из файла конфигурации video-worker, map[string:string]

 "meta": {

 "cam\_id": "abc", // id job-задания в video-worker

 "end\_of\_track": true, // завершение ли это трека(true), либо отправка в процессе трекинга(false) при включенном realtime в video-worker

 "track\_duration\_seconds": 6.790000000000077,

 "track": {

 "id": "162f7743a6cf-97", // уникальный идентификатор трека

 "first\_timestamp": "2020-12-16T09:57:58.764Z",

 "last\_timestamp": "2020-12-16T09:57:59.307Z",

 "body": { // силуэт

 "history": { // история bbox, время от first\_timestamp до last\_timestamp. Может быть обрезан до N последних значений настройкой video-worker!

 "first\_timestamp": "2020-12-16T09:57:58.764Z", // время первого элемента истории

 "last\_timestamp": "2020-12-16T09:57:59.307Z", // время последнего элемента истории

 "bbox": [

 [556, 396, 746, 715], [509, 397, 668, 711], [491, 397, 605, 719], [428, 400, 559, 719], [379, 384, 487, 719], [320, 392, 454, 712],

 [298, 378, 409, 718], [275, 380, 386, 706], [248, 383, 363, 705], [224, 385, 305, 668], [194, 388, 296, 667], [175, 380, 265, 640],

 [169, 378, 246, 628], [151, 385, 234, 619], [133, 371, 205, 601], [120, 380, 190, 597], [110, 371, 176, 586], [99, 368, 167, 573],

 [89, 375, 152, 567], [76, 364, 135, 562], [66, 373, 122, 549], [61, 367, 117, 541], [58, 372, 111, 535], [51, 374, 106, 533],

 [44, 367, 97, 528], [34, 371, 87, 522], [29, 361, 83, 515], [26, 360, 80, 512], [21, 361, 64, 508], [12, 361, 56, 505],

 [8, 358, 53, 500], [3, 366, 42, 489], [3, 358, 43, 493], [0, 364, 31, 485], [0, 361, 23, 486]

 ]

 },

 "first": { // первый кадр трека

 "timestamp": "2020-12-16T09:57:58.764Z",

 "bbox": [556, 396, 746, 715], // [left, top, right, bottom]

 "quality": 0.9988629817962646,

 "full\_frame": "JPEG-binary-data",

 },

 "last": { // последний кадр трека

 "timestamp": "2020-12-16T09:57:59.307Z",

 "bbox": [0, 361, 23, 486],

 "quality": 0.9535224437713623,

 "full\_frame": "JPEG-binary-data",

 },

 "best": { // лучший(по quality) кадр трека

 "full\_frame": "JPEG-binary-data", // jpeg полный кадр

 "normalized": "binary-data",

 "timestamp": "2020-12-16T09:57:58.833Z",

 "bbox": [298, 378, 409, 718],

 "quality": 0.9997832179069519

 }

 }

 }

 }

}

```

## Основные действия и приемы при эксплуатации ПО

Эксплуатация ПО заключается в следующих действиях:

1. управлениe заданиями на детектирование силуэтов в видеопотоке;
2. считывание статистики по обработке заданий.

Оба действия выполняются посредством отправки HTTP API запросов в компонент VideoManager. Подробное описание методов HTTP API приведено в документе Руководство.

## Обслуживание и устранение неисправностей

### Мероприятия по обеспечению защиты информации

### Ограничение доступа к ПО

Ограничение доступа к изменению параметров ПО выполняется средствами стороннего ПО Kubernetes (обеспечивает развертывание и функционирование ПО), etcd (внутренняя база данных ПО), MongoDB (StatDB). Подробная информация приведена в официальной справочной документации на данные программы:

1. Kubernetes: <https://kubernetes.io/docs/reference/access-authn-authz/rbac/>.
2. ETCD: <https://github.com/etcd-io/etcd/blob/master/Documentation/op-guide/security.md>
3. MongoDB: <https://docs.mongodb.com/manual/core/authentication/>.

#### Формирование матрицы доступа и централизованное администрирование списка пользователей

Формирование матрицы доступа ПО, а также обеспечение централизованного администрирования списка пользователей выполняются средствами ПО Kubernetes.

#### Журналирование действий пользователей на уровне операционной системы

Для защиты логирования рекомендуется настроить логирование на уровне операционной системы с использованием функционала journald <https://docs.docker.com/config/containers/logging/journald/>.

Для этого необходимо включить настройку **Seal=True** (подробнее см. [https://lwn.net/Articles/512895/)](https://lwn.net/Articles/512895/%29), руководствуясь описанием <https://www.freedesktop.org/software/systemd/man/journald.conf.html>.

#### Идентификация смежных систем

Для аутентификации в ПО, включая программную идентификацию со стороны смежных систем, рекомендуется использовать встроенные средства ПО Kubernetes. В этом случае необходимо развернуть данный функционал Kubernetes, руководствуясь инструкцией <https://kubernetes.github.io/ingress-nginx/examples/auth/client-certs/>.

### Резервное копирование и восстановление баз данных ПО

Функционирование ПО требует использования следующих баз данных:

1. Внутренняя база данных ПО, реализованная на основе стороннего программного обеспечения ETCD.
2. **StatDB**, хранящая данные статистики, реализованная на основе стороннего программного обеспечения MongoDB.

Резервное копирование и восстановление баз данных выполняется с использованием методов и алгоритмов, предусмотренных официальной справочной документацией к стороннему программному обеспечению:

1. Описание резервного копирования и восстановления внутренней базы данных ПО (etcd): <https://github.com/etcd-io/etcd/blob/master/Documentation/op-guide/recovery.md>.
2. Описание резервного копирования и восстановления StatDB (MongoDB): <https://docs.mongodb.com/manual/tutorial/backup-and-restore-tools/>.

### Просмотр логов

При разборе нештатных ситуаций используйте логи, содержащие подробную деталировку всех событий, произошедших в системе.

Примечание. Для того чтобы настроить размер логов, внесите изменения в файл /etc/docker/daemon.json. Наиболее оптимальный размер дают следующие настройки:

{
 ...
 "log-driver": "json-file",
 "log-opts": {"max-file": "2", "max-size": "1024m"}
 ...
}

Для просмотра логов выполните следующую команду, введя вместо %pod-name% название интересующего pod kubernetes:

kubectl logs %pod-name%

Для того чтобы отобразить список активных на данный момент pod-ов, выполните команду:

kubectl get pods

Пример результата работы команды:

NAME READY STATUS RESTARTS AGE
v-archive-scheduler-588fcfd55d-cjdst 1/1 Running 0 20h
v-cctv-redirector-587d4bd5cb-mvqbs 1/1 Running 0 19h
v-etcd-0-0 1/1 Running 0 22h
v-etcd-0-1 1/1 Running 0 22h
v-etcd-0-2 1/1 Running 0 22h
v-face-sender-75f7f8c47d-9qxf8 1/1 Running 0 22h
v-mcrouter-0 1/1 Running 0 22h
v-memcached-0 1/1 Running 0 22h
v-mongo-0 1/1 Running 0 22h
v-mongo-1 1/1 Running 0 22h
v-mongo-2 1/1 Running 0 22h
v-video-manager-0-0 1/1 Running 0 22h
v-video-manager-0-1 1/1 Running 0 22h
v-video-worker-0-796674f8db-4ghqv 1/1 Running 9 22h
v-video-worker-0-796674f8db-cztt5 1/1 Running 9 22h
v-video-worker-0-796674f8db-kq8dj 1/1 Running 9 22h
v-video-worker-0-796674f8db-vhj82 1/1 Running 9 22h

Соответствие имен pod-ов структурным компонентам ПО описано в разделе Особенности работы ПО в Kubernetes.

### Рекомендации по периодическому обслуживанию компонентов ПО

Для своевременной диагностики неисправностей при функционировании ПО рекомендуется осуществлять своевременную ротацию логов (см. раздел Просмотр логов).

### Мониторинг диагностических данных о состоянии ПО и комплекса технических средств среды эксплуатации. Получение и обработка диагностических данных программными средствами

Метрики работы ПО собираются централизованно в компоненте VideoManager (соответствующий внутренний процесс в Kubernetes: **video-manager**).

Для мониторинга диагностических данных о состоянии ПО и комплекса технических средств среды эксплуатации можно использовать следующие методы:

1. мониторинг диагностических данных вручную;
2. диагностика посредством написания программного кода, автоматизирующего диагностические мероприятия, выполняемые в ручном режиме;
3. получение и обработка диагностических данных программными средствами, поддерживающими стандарт OpenMetrics.

#### Диагностика ПО

Для диагностики работы ПО необходимо выполнить следующие действия:

1. Для получения метрик работы ПО отправьте API-запрос в компонент VideoManager (подробное описание работы с данным компонентом через внутренний HTTP API приведено в документе «Руководство»).

GET /jobs

Возвращает список всех заданий на обработку видеоизображений в представлении JSON.

**Пример**

**Запрос**

curl -s 'http://localhost:11358/jobs' | jq –S .

**Ответ**

[

 {

 "enabled": true,

 "id": "myid-123",

 "labels": {

 "district": "SVAO"

 },

 "router\_url": "",

 "single\_pass": false,

 "statistic": {

 "faces\_failed": 1,

 "faces\_not\_posted": 4,

 "faces\_posted": 16,

 "frames\_dropped": 0,

 "frames\_processed": 0,

 "processed\_duration": 1.5078692436218262,

 "processing\_fps": 0

 },

 "status": "INPROGRESS",

 "status\_msg": "Job is being processed",

 "stream\_settings": {

 "filter\_min\_faces\_size": 1,

 "filter\_min\_quality": -2,

 "start\_stream\_timestamp": 0,

 "use\_stream\_timestamp": true

 },

 "stream\_url": "http://1.2.3.4/stream.mp4",

 "version": "2019-09-23T17:02:57.700316",

 "worker\_id": "d60301b4-69f6-4c83-97e8-642bbe4c7226"

 }

]

1. Ответ VideoManager содержит необходимую для диагностики информацию. В первую очередь требуется интерпретировать значение поля **status**, в котором передается информация о статусе задания на обработку видеоизображения (Таблица 3):

**Таблица 3. Статусы заданий**

| **Значение поля status** | **Интерпретация** |
| --- | --- |
| UNKNOWN | Статус задания неизвестен. |
| AWAITING | Задание ожидает распределения в один из экземпляров VideoDetector. При длительном нахождении задания в данном статусе необходимо проверить, есть ли свободные экземпляры VideoDetector: такая ситуация возможна в случае недостаточного количества ресурсов, выделенных на обработку видеоизображения. |
| DISABLED | Задание неактивно (рабочее состояние, не указывающее на возможный сбой). |
| STARTING | Задание распределено в один из экземпляров VideoDetector. Обработка видеоизображения в процессе, однако первая статистика еще не получена. |
| INPROGRESS | Задание в данный момент обрабатывается с регулярным обновлением статистики.  |
| COMPLETED | Обработка задания завершена. После данного статуса задание при отсутствии сбоя может перейти в AWAITING->STARTING->INPROGRESS. Если задание перешло в STARTING->COMPLETED, требуется проверить содержимое поля status\_msg, содержащего дополнительные данные, по данному заданию (GET /job/:id). |
| INTERRUPTED | От VideoManager поступил запрос на остановку выполнения задания компонентом VideoDetector, однако данный процесс завершился с ошибкой. Может быть связано с рассинхронизацией, когда на момент получения подобного запроса VideoDetector уже не обрабатывает данное задание. |
| NOT\_STARTED | Задание было распределено в один из экземпляров VideoDetector, однако статистика так и не была получена. Свидетельствует о сбое. |
| DISCONNECTED | VideoDetector, обрабатывающий задание, недоступен. Необходимо проверить, что через некоторое время статус перешел в AWAITING->STARTING->INPROGRESS. |

Для удобства интерпретации при диагностике ПО вы можете пользоваться схемой на Рис. 3. Цвета стрелок, обозначающих переходы между статусами, интерпретируются следующим образом:

* Зеленый – штатный режим работы.
* Красный – сбой.
* Фиолетовый – перезапуск.
* Серый – выключение.



Рис. 3 Возможные изменения статуса задания

1. Выполнить анализ логов CCTV API.

kubectl get pods | grep cctv # получить список работающих pod-ов
kubect logs v-cctv-redirector-587d4bd5cb-mvqbs --tail=10 -f

1. Выполните анализ логов для других компонентов ПО (см. раздел Просмотр логов).

Описанные выше диагностические мероприятия можно также автоматизировать посредством написания программного кода.

#### Получение и обработка диагностических данных программными средствами, поддерживающими стандарт OpenMetrics

Для полностью автоматизированного сбора и обработки диагностических данных рекомендуется использовать стороннее ПО Prometheus (https://prometheus.io).

Для автоматизированного сбора метрик video-manager, функционирующего в виде pod-а Kubernetes, выполните следующие действия:

1. Сформируйте job-задание для Prometheus по аналогии с приведенным в примере ниже:

scrape\_configs:
- job\_name: 'pods.em.k8s'
 scheme: https
 tls\_config:
 insecure\_skip\_verify: true
 ca\_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
 server\_name: kubernetes
 bearer\_token\_file: /var/run/secrets/kubernetes.io/serviceaccount/token
 kubernetes\_sd\_configs:
 - role: pod
 relabel\_configs:
 - source\_labels: [\_\_meta\_kubernetes\_pod\_annotation\_prometheus\_io\_scrape]
 action: keep
 regex: true
 - source\_labels: [\_\_scheme\_\_]
 action: replace
 replacement: http
 target\_label: \_\_scheme\_\_
 - source\_labels: [\_\_meta\_kubernetes\_namespace]
 action: replace
 target\_label: k8snamespace
 - source\_labels: [\_\_address\_\_, \_\_meta\_kubernetes\_pod\_annotation\_prometheus\_io\_port]
 action: replace
 regex: ([^:]+)(?::\d+)?;(\d+)
 replacement: $1:$2
 target\_label: \_\_address\_\_
 - action: labelmap
 regex: \_\_meta\_kubernetes\_pod\_label\_(.+)
 - source\_labels: [\_\_meta\_kubernetes\_pod\_name]
 action: replace
 target\_label: pod

1. В файле **values.yaml** в ПО helm установите параметр **global.metricsExport** в значение **true**.
2. В Prometheus настройте правила, по которым система будет отправлять оповещения при наличии тех или иных неполадок в работе ПО. Пример правил отправки оповещений:

groups:
 - name: videocontroller
 rules:
 - alert: JOBS\_not\_inprogress
 expr: sum(jobs\_count{status !~ "(INPROGRESS|DISABLED)"}) by (instance,job,k8snamespace) > 0
 for: 5m
 labels:
 severity: warning
 annotations:
 summary: "Jobs in Not-INPROGRESS status detected"
 description: "Too much jobs not in INPORGRESS status . Check video-manager at {{ $labels.instance }}" - alert: JOBS\_not\_inprogress\_HIGH
 expr: sum(jobs\_count{status !~ "(INPROGRESS|DISABLED)"}) by (instance,job,k8snamespace) / sum(jobs\_count) by (instance,job,k8snamespace)> 0.2
 for: 5m
 labels:
 severity: warning
 annotations:
 summary: "Jobs in Not-INPROGRESS status detected"
 description: "Too much jobs not in INPORGRESS status . Check video-manager at {{ $labels.instance }}" - alert: JOBS\_not\_inprogress\_HIGH
 expr: sum(jobs\_count{status !~ "(INPROGRESS|DISABLED)"}) by (instance,job,k8snamespace) / sum(jobs\_count) by (instance,job,k8snamespace)> 0.5
 for: 5m
 labels:
 severity: critical
 annotations:
 summary: "Jobs in Not-INPROGRESS status detected"
 description: "Too much jobs not in INPORGRESS status . Check video-manager at {{ $labels.instance }}" - alert: JOBS\_processed\_duration\_LOW
 expr: rate(jobs\_processed\_duration[5m]) < 0.9 and jobs\_processed\_duration > 0
 for: 10m
 labels:
 severity: warning
 annotations:
 summary: "Job processed duration LOW"
 description: "Job processed duration < 0.9 for last 10 minutes. Check video-manager and video-worker at {{ $labels.instance }}" - alert: JOB\_Not\_Started
 expr: jobs\_processed\_duration == 0
 for: 5m
 labels:
 severity: warning
 annotations:
 summary: "Job not started"
 description: "Job processed duration = 0 for last 5 minutes. Check video-manager and video-worker at {{ $labels.instance }}"

### Действия персонала при сбое и аварийном режиме

При возникновении аварийного режима или неисправностей в работе ПО необходимо выполнить следующие действия:

1. Проверить состояние MongoDB и, в частности, rs.status().

kubectl exec -it vv-mongo-0 — mongo

1. Проверить доступность процессов video-manager (соответствует структурному компоненту VideoManager).

kubectl get deploy vv-video-manager-0

1. Проверить состояние etcd.

kubectl logs vv-etcd-0-0

1. Проверить, является ли количество процессов video-worker (соответствует VideoDetector) достаточным и все ли они запущены.

kubect get deploy vv-video-worker-0

1. Проверить процессы face-sender (соответствует FaceSender).

kubectl get deploy vv-face-sender

kubectl logs vv-face-sender-f865f9d5d-7l9wl

1. При наличии ошибок и важных оповещений (“warning”) в логах внутренних процессов ПО, принять меры по устранению их причин.

# Руководство

## Интерфейс прикладного программирования ПО и методы работы с ним

Интерфейсом прикладного программирования ПО является программный интерфейс HTTP API компонента VideoManager.

Данный интерфейс обеспечивает выполнение следующих функций:

1. управлениe заданиями на детектирование силуэтов в видеопотоке;
2. считывание статистики по обработке заданий.

Подробное описание архитектуры ПО и компонентов ПО приведено в главе Руководство администратора.

### Основные сведения

#### Точка доступа

Запросы HTTP API для управления видеодетекцией силуэтов нужно отправлять на адрес http://<IP адрес VideoManager>:/11358. Запросы обрабатываются компонентом VideoManager.

#### Объект типа job (задание)

Объект job представляет собой задание на обработку видеопотока, выдаваемое VideoManager компоненту VideoDetector.

Параметры:

* id: id задания.
* stream\_url: URL видеопотока или адрес видеофайла, который нужно обработать.
* labels: метки, по которым будет осуществляться обработка обнаруженных силуэтов.
* single\_pass: если true (по умолчанию false), то не перезапускать обработку потока в случае ошибки.
* router\_url: IP-адрес и порт, на которые компонент VideoDetector отправляет обнаруженные силуэты для дальнейшей обработки.
* status: статус задания.
* status\_msg: дополнительная информация о статусе задания.
* statistic: статистика выполнения задания (продолжительность использования задания, количество отправленных силуэтов).
* worker\_id: id экземпляра компонента VideoDetector, который выполняет задание.

#### Сообщения об ошибках

Если метод выполнить не удается, Сервер возвращает ответ с кодом HTTP, отличном от 200, а также тело ответа в формате JSON, содержащее описание ошибки. Тело ответа всегда содержит минимум 2 поля — code и desc:

1. code — это код ошибки, который может быть использован для автоматического преобразования;
2. desc — это описание ошибки, предназначенное для прочтения человеком, а не для автоматического преобразования.

Распространенные коды ошибок представлены в таблице ниже:

|  |  |  |
| --- | --- | --- |
| Код ошибки | Описание | Код HTTP |
| UNKNOWN\_ERROR | Ошибка неизвестного происхождения. | 500 |
| BAD\_PARAM | Запрос можно прочитать, однако некоторые параметры метода недействительны. | 400 |
| MALFORMED\_REQUEST | Запрос неверно сформирован и не может быть прочитан. | 400 |
| CONFLICT | Конфликт. | 409 |
| NOT\_FOUND | Искомое задание не найдено. | 404 |

### Методы для управления видеодетекцией силуэтов

#### Создание задания

POST /job/:id

Создает задание на обработку видеопотока для компонента VideoDetector.

**Параметры в теле запроса**:

* stream\_url: URL видеопотока или адрес видеофайла, который нужно обработать.
* labels: метки, по которым будет осуществляться обработка обнаруженных силуэтов.
* single\_pass: если true (по умолчанию false), то не перезапускать обработку потока в случае ошибки.

**Возвращает**:

Объект job: все, что было передано в запросе, а также параметры объекта, предназначенные только для чтения:

**Пример**:

**Запрос**:

curl -s -X POST 'http://localhost:11358/job/myid-123' --data '{"stream\_url":"http://1.2.3.4/stream.mp4", "labels": {"district": "SVAO"}}' | jq -S .

**Ответ:**

{

 "enabled": true,

 "id": "myid-123",

 "labels": {

 "district": "SVAO"

 },

 "router\_url": "",

 "single\_pass": false,

 "statistic": {

 "faces\_failed": 0,

 "faces\_not\_posted": 0,

 "faces\_posted": 0,

 "frames\_dropped": 0,

 "frames\_processed": 0,

 "processed\_duration": 0,

 "processing\_fps": 0

 },

 "status": "AWAITING",

 "status\_msg": "New job is awaiting to be scheduled",

 "stream\_settings": {

 "filter\_min\_face\_size": 1,

 "filter\_min\_quality": -2,

 "start\_stream\_timestamp": 0,

 "use\_stream\_timestamp": true

 },

 "stream\_url": "http://1.2.3.4/stream.mp4",

 "version": "2019-09-23T17:02:57.700316",

 "worker\_id": null

}

#### Получение списка всех заданий

GET /jobs

Возвращает список всех заданий.

**Параметры:**

Отсутствуют.

**Возвращает:**

Список всех заданий в представлении JSON.

**Пример**

**Запрос**

curl -s 'http://localhost:11358/jobs' | jq –S .

**Ответ**

[

 {

 "enabled": true,

 "id": "myid-123",

 "labels": {

 "district": "SVAO"

 },

 "router\_url": "",

 "single\_pass": false,

 "statistic": {

 "faces\_failed": 1,

 "faces\_not\_posted": 4,

 "faces\_posted": 16,

 "frames\_dropped": 0,

 "frames\_processed": 0,

 "processed\_duration": 1.5078692436218262,

 "processing\_fps": 0

 },

 "status": "INPROGRESS",

 "status\_msg": "Job is being processed",

 "stream\_settings": {

 "filter\_min\_face\_size": 1,

 "filter\_min\_quality": -2,

 "start\_stream\_timestamp": 0,

 "use\_stream\_timestamp": true

 },

 "stream\_url": "http://1.2.3.4/stream.mp4",

 "version": "2019-09-23T17:02:57.700316",

 "worker\_id": "d60301b4-69f6-4c83-97e8-642bbe4c7226"

 }

]

#### Получение задания по id

GET /job/:id

Возвращает параметры задания по id.

**Параметры:**

id: id задания.

**Возвращает:**

Параметры задания в представлении JSON.

**Пример**

**Запрос**

curl -s 'http://localhost:11358/job/myid-123' | jq -S .

**Ответ**

{

 "enabled": true,

 "id": "myid-123",

 "labels": {

 "district": "SVAO"

 },

 "router\_url": "",

 "single\_pass": false,

 "statistic": {

 "faces\_failed": 0,

 "faces\_not\_posted": 0,

 "faces\_posted": 0,

 "frames\_dropped": 0,

 "frames\_processed": 0,

 "processed\_duration": 0,

 "processing\_fps": 0

 },

 "status": "STARTING",

 "status\_msg": "Assigned and sent to worker",

 "stream\_settings": {

 "filter\_min\_face\_size": 1,

 "filter\_min\_quality": -2,

 "start\_stream\_timestamp": 0,

 "use\_stream\_timestamp": true

 },

 "stream\_url": "http://1.2.3.4/stream.mp4",

 "version": "2019-09-23T17:02:57.700316",

 "worker\_id": "d60301b4-69f6-4c83-97e8-642bbe4c7226"

}

#### Удаление задания по id

DELETE /job/:id

Удаляет задание по id.

**Параметры:**

id: id задания.

**Возвращает:**

Параметры удаленного задания в представлении JSON.

**Пример**

**Запрос**

curl -s 'http://localhost:11358/job/myid-123' -X DELETE | jq -S .

**Ответ**

{

 "enabled": true,

 "id": "myid-123",

 "labels": {

 "district": "SVAO"

 },

 "router\_url": "",

 "single\_pass": false,

 "statistic": {

 "faces\_failed": 0,

 "faces\_not\_posted": 0,

 "faces\_posted": 0,

 "frames\_dropped": 0,

 "frames\_processed": 0,

 "processed\_duration": 0,

 "processing\_fps": 0

 },

 "status": "INTERRUPTED",

 "status\_msg": "Job is being deleted",

 "stream\_settings": {

 "filter\_min\_face\_size": 1,

 "filter\_min\_quality": -2,

 "start\_stream\_timestamp": 0,

 "use\_stream\_timestamp": true

 },

 "stream\_url": "http://1.2.3.4/stream.mp4",

 "version": "2019-09-23T17:06:45.463192",

 "worker\_id": null

}

#### Изменение задания

PATCH /job/:id

Изменяет задание по id.

**Параметры:**

id: id задания.

**Параметры в теле запроса**:

* stream\_url: URL видеопотока или адрес видеофайла, который нужно обработать.
* labels: метки, по которым будет осуществляться обработка обнаруженных силуэтов.
* single\_pass: если true (по умолчанию false), то не перезапускать обработку потока в случае ошибки.
* router\_url: IP-адрес и порт, на которые компонент VideoDetector отправляет обнаруженные силуэты для дальнейшей обработки.
* status: статус задания.
* status\_msg: дополнительная информация о статусе задания.
* statistic: статистика выполнения задания (продолжительность использования задания, количество отправленных силуэтов).
* worker\_id: id экземпляра компонента VideoDetector, который выполняет задание.

**Возвращает:**

Параметры измененного задания в представлении JSON.

**Пример**

**Запрос**

curl -s 'http://localhost:11358/job/myid-123' -X PATCH --data '{"router\_url":"http://myrouter"}' | jq -S .

**Ответ**

{

 "enabled": true,

 "id": "myid-123",

 "labels": {

 "district": "SVAO"

 },

 "router\_url": "http://myrouter",

 "single\_pass": false,

 "statistic": {

 "faces\_failed": 0,

 " faces\_not\_posted": 0,

 " faces\_posted": 0,

 "frames\_dropped": 0,

 "frames\_processed": 0,

 "processed\_duration": 0,

 "processing\_fps": 0

 },

 "status": "AWAITING",

 "status\_msg": "Waiting to be rescheduled after an update",

 "stream\_settings": {

 "filter\_min\_face\_size": 1,

 "filter\_min\_quality": -2,

 "start\_stream\_timestamp": 0,

 "use\_stream\_timestamp": true

 },

 "stream\_url": "http://1.2.3.4/stream.mp4",

 "version": "2019-09-23T17:07:54.819971",

 "worker\_id": null

}

#### Перезапуск задания

POST /job/:id/restart

Перезапускает задание.

**Параметры:**

id: id задания.

**Возвращает:**

Параметры перезапущенного задания.

**Пример**

**Запрос**

curl -s -X POST http://localhost:11358/job/myid-123/restart| jq -S .

**Ответ**

{

 "enabled": true,

 "id": "myid-123",

 "labels": {

 "district": "SVAO"

 },

 "router\_url": "http://myrouter",

 "single\_pass": false,

 "statistic": {

 "faces\_failed": 0,

 "faces\_not\_posted": 0,

 "faces\_posted": 0,

 "frames\_dropped": 0,

 "frames\_processed": 0,

 "processed\_duration": 0,

 "processing\_fps": 0

 },

 "status": "AWAITING",

 "status\_msg": "Waiting to be scheduled after manual restart",

 "stream\_settings": {

 "filter\_min\_face\_size": 1,

 "filter\_min\_quality": -2,

 "start\_stream\_timestamp": 0,

 "use\_stream\_timestamp": true

 },

 "stream\_url": "http://1.2.3.4/stream.mp4",

 "version": "2019-09-23T17:07:54.819971",

 "worker\_id": null