

**Руководство пользователя
«Системы гарантированной доставки http - сообщений»**

1. Введение

1.1. Область применения

Настоящий документ предназначен для сотрудников эксплуатирующей организации и отражает основные функциональные возможности и порядок действий при выполнении операций, связанных с администрированием программного обеспечения «Система гарантированной доставки http — сообщений» (далее - «Система»)

1.2. Перечень выполняемых функций администратора/оператора

В перечень выполняемых функций администратора Системы входят:

- Установка и настройка Системы
- Реализация планов устранения сбоев и нетиповых нештатных ситуаций
- Выполнение сбора и предоставление в вышестоящую линию технической поддержки информации для воспроизведения технических проблем и выработки решений по их разрешению
- Реализация рекомендаций по устранению нештатных ситуаций, полученных с вышестоящей линии поддержки
- Восстановление работоспособности Системы при сбоях в работе функциональных модулей
- Разработка решения по устранению технических проблем в работе функциональных модулей

2. Настройка Системы

2.1. Общие сведения

В данном документе приводятся примеры настройки Системы с использованием среды Docker Compose. Настройка операционной системы, сервера очередей Apache Kafka, а также возможная настройка использования систем оркестрации, находятся вне компетенции этого документа и не будут тут описаны.

2.2. Модуль приема сообщений

2.2.1 Конфигурируемые параметры

Для корректной работы модуля приема сообщений, необходимо настроить для него следующие переменные окружения:

- SETTINGS_FILE - путь к файлу с настройками запросов. Файл подключается к контейнеру с помощью volume. В данной переменной окружения указывается локальный путь внутри контейнера, к которому был примонтирован volume.
- METRICS_PORT - порт к подсистеме проверки работоспособности.
- HOST - адрес сервиса, который будет слушать модуль. На этот адрес следует пробросить внешний порт или настроить проксирующий сервер для поддержки протокола https.
- KAFKA_HOST — имя хоста или IP сервера Kafka

- KAFKA_TOPIC - топик на сервере Kafka
- KAFKA_PARTITION — имя разделения (partition) на сервере Kafka
- KAFKA_TIMEOUT - таймаут запросов к серверу Kafka
- LOG_LEVEL - уровень логгирования. Поддерживаемые значения:
 - error
 - warn
 - info
 - debug
 - trace

Пример настройки модуля:

```

httpin:
  build:
    context: ./http-in/
  restart: always
  ports:
    - '80:80'
  volumes:
    - ./http-in/config.json:/app/config.json:ro
  environment:
    HOST: :80
    KAFKA_HOST: host.docker.internal:9092
    KAFKA_TOPIC: test
    KAFKA_PARTITION: 0
    KAFKA_TIMEOUT: 10
    LOG_LEVEL: trace
    SETTINGS_FILE: /app/config.json
  extra_hosts:
    - "host.docker.internal:host-gateway"

```

2.2.2. Настройка маршрутизации

Для того, чтобы сервер обрабатывал конкретный URL, его следует прописать в файле, подключенном к модулю и указанном в переменной окружения SETTINGS_FILE. Этот файл содержит в текстовом формате json — объект, который в свою очередь, содержит еще два объекта:

- requests — содержит описание маршрутов и обработчики поступающих данных

Ключами объекта requests являются описатели маршрутов. Они формируются путем конкатенации HTTP-метода, разделителя и относительного пути.

Префиксом названия команды является HTTP-метод. Поддерживаемые методы: GET, POST, PUT, DELETE. После префикса должен находиться разделитель |. Затем должен быть указан маршрут. Маршрут может включать в себя как параметры, так и символ *, позволяющий обрабатывать различные маршруты одной

командой. Примеры:

- GET|/users
- GET|/users/:id
- GET|/users/files/*
- POST|/users/:id

Значениями объекта `requests` являются строки, содержащие в себе JavaScript-код.

- `responses` — содержит обработчики ответов, отправляемых модулем клиенту.

Ключами объекта `responses` являются описатели маршрутов, описанные выше.

Значениями объекта `responses` являются строки, содержащие в себе JavaScript-код.

Пример настройки маршрутизации:

```
{
  "requests": {
    "GET|/ping":""
  },
  "responses": {
    "GET|/ping":""
  }
}
```

2.2.3. Обработка запросов

Для каждого поступающего запроса может быть написан JavaScript-код, который позволит модифицировать как входящие параметры, так и ответ на запрос. В силу ограничений формата `json`, в коде не поддерживаются переносы строк и кавычки. Эти символы могут быть вставлены в код с помощью символа экранирования, и соответственно, будут выглядеть как `\n` и `\"`.

Для обработки входящего запроса, в JavaScript передаются параметры:

- `raw_data` - RAW-содержимое запроса. Представляет собой `json`-закодированную строку. Передается в JavaScript как массив байт.
- `body` - тело запроса. Передается в JavaScript как массив байт.
- `cookies` - массив `cookie`
- `headers` - массив с заголовками запроса
- `method` - HTTP-метод
- `query_params` - параметры запроса в URL
- `url` - URL запроса

JavaScript-код может проанализировать эти параметры и должен вернуть параметр `raw_data`, который будет отправлен модулю доставки сообщений.

Для обработки ответа на запрос, в JavaScript передается параметр `raw_data` - RAW-

содержимое ответа.

В JavaScript дополнительно передаются внешние функции:

- atob - преобразует строку в base64
- btoa - преобразует base64 в строку

Если в разделе responses не указать соответствующий запросу маршрут, то модуль ответит кодом 200 Ok с пустым телом ответа.

2.3. Модуль доставки сообщений

2.3.1. Конфигурируемые параметры

- SETTINGS_FILE - путь к файлу с настройками запросов. Файл подключается к контейнеру с помощью volume. В данной переменной окружения указывается локальный путь внутри контейнера, к которому был примонтирован volume.
- METRICS_PORT - порт к подсистеме проверки работоспособности.
- ALLOW_INSECURE - флаг контролировать ли сертификат системы получателя
- HTTP_HOST — имя хоста или IP адрес системы получателя
- HTTP_TIMEOUT - таймаут запроса
- KAFKA_HOST - имя хоста или IP адрес сервера Kafka
- KAFKA_TOPIC - топик на сервере Kafka
- KAFKA_TIMEOUT - таймаут запросов к серверу Kafka
- KAFKA_GROUP - группа подписчиков сервера Kafka
- LOG_LEVEL - уровень логгирования. Поддерживаемые значения:
 - error
 - warn
 - info
 - debug
 - trace

Пример настройки модуля:

```
httpout:
  build:
    context: ./http-out/
  restart: always
  volumes:
    - ./http-out/config.json:/app/config.json:ro
  environment:
    KAFKA_HOST: host.docker.internal:9092
    KAFKA_TOPIC: test
    KAFKA_GROUP: consumer-group-id-1
    KAFKA_TIMEOUT: 10
    LOG_LEVEL: trace
    SETTINGS_FILE: /app/config.json
```

```
HTTP_HOST: google.com
HTTP_TIMEOUT: 10
extra_hosts:
- "host.docker.internal:host-gateway"
```

2.3.2. Обработка поступающих запросов

Каждый запрос, полученный от модуля приема сообщений, приходит с теми же настройками маршрутизации, которые были описаны выше в разделе 3.2.2.

Принимаемые сообщения могут быть обработаны с помощью JavaScript-кода, который позволит модифицировать как входящие параметры, так и проанализировать ответ на команду.

Для того, чтобы модуль обработал конкретный поступающий запрос, его следует прописать в файле, подключенном к модулю и указанном в переменной окружения `SETTINGS_FILE`. Этот файл содержит в текстовом формате `json` — объект, который в свою очередь, содержит еще два объекта:

- `requests` — содержит описание маршрутов и обработчики поступающих данных

Ключами объекта `requests` являются описатели маршрутов. Они должны совпадать с аналогичными маршрутами, указанными у модуля приема сообщений.

Значениями объекта `requests` являются строки, содержащие в себе JavaScript-код.

- `responses` — содержит обработчики ответов, отправляемых модулем клиенту.

Ключами объекта `responses` являются описатели маршрутов, описанные выше.

Значениями объекта `responses` являются строки, содержащие в себе JavaScript-код.

Для каждой команды может быть написан JavaScript-код, который позволит модифицировать как входящие параметры, так и позволит изменить поведение при ответе на команду.

Для обработки входящей команды, в JavaScript передаются параметры:

- `raw_data` - RAW-содержимое запроса. Представляет собой `json`-закодированную строку. Передается в JavaScript как массив байт.
- `body` - тело запроса. Передается в JavaScript как массив байт.
- `cookies` - массив `cookie`
- `headers` - массив с заголовками запроса
- `method` - HTTP-метод
- `query_params` - параметры запроса в URL
- `url` - URL запроса JavaScript-код может проанализировать и изменить эти параметры.

Для обработки ответа команды, в JavaScript передаются параметры:

- `raw_data` - RAW-содержимое ответа. Представляет собой json-закодированную строку. Передается в JavaScript как массив байт.
- `body` - тело ответа. Передается в JavaScript как массив байт.
- `status_code` - HTTP-код ответа
- `headers` - массив заголовков
- `cookies` - массив cookie

JavaScript-код может проанализировать эти параметры и должен вернуть параметр `status_code`.

Если `status_code` будет 2xx, сообщение будет помечено как обработанное. В противном случае, через некоторое время будет произведена попытка повторной доставки сообщения.

В JavaScript дополнительно передаются внешние функции:

- `atob` - преобразует строку в base64
- `btoa` - преобразует base64 в строку

Если в разделе `requests` не указать соответствующий запросу маршрут, то модуль отправит запрос в систему получатель в том виде, в котором его получил модуль приема сообщений

Если в разделе `responses` не указать соответствующий запросу маршрут, то модуль будет проверять статус ответа на запрос и если его `status_code` будет 2xx, сообщение будет помечено как обработанное. В противном случае, через некоторое время будет произведена попытка повторной доставки сообщения.

Пример файла настройки обработчика запросов:

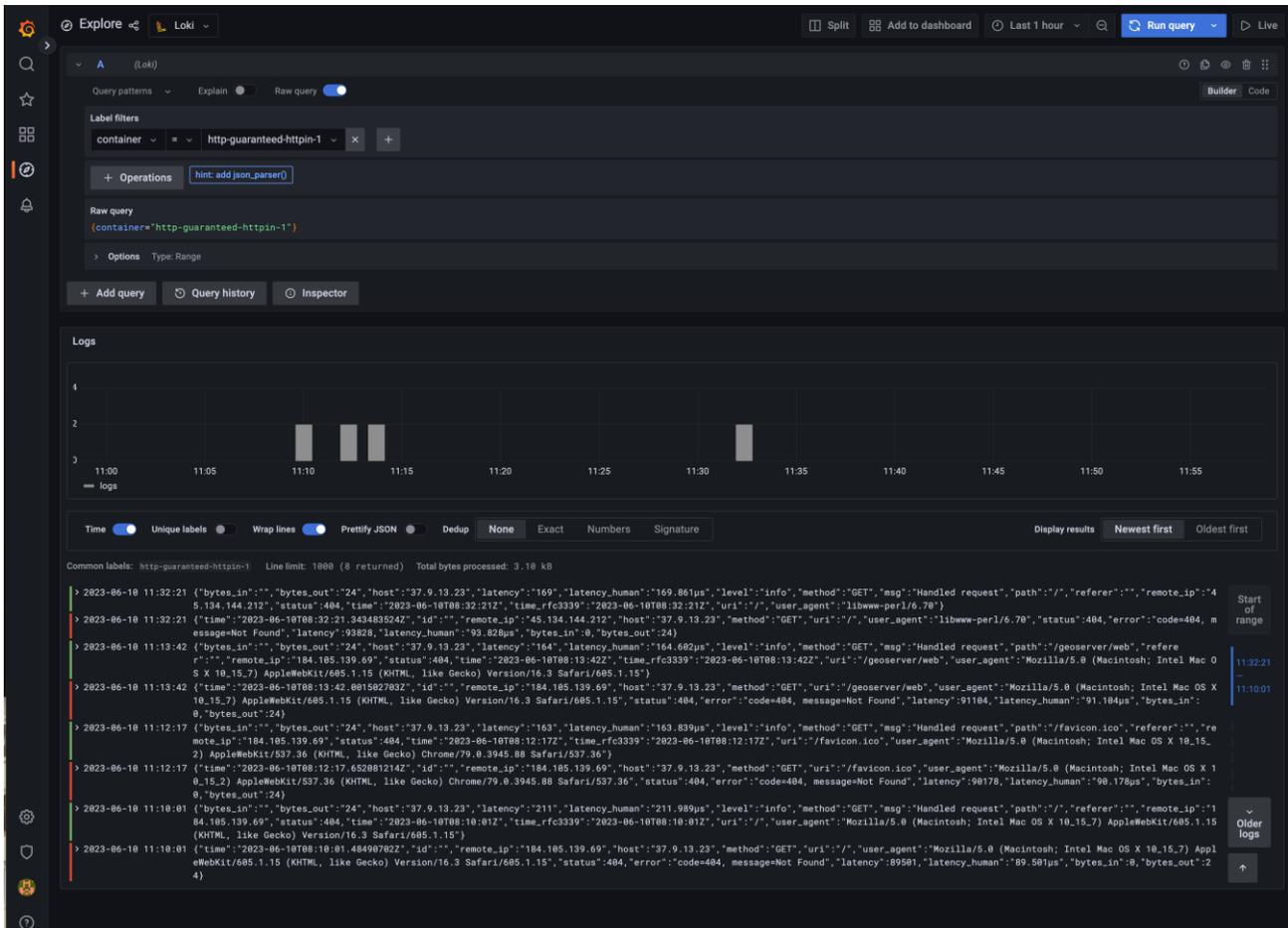
```
{
  "requests": {
    "default": " ",
    "GET|/ping": "method='GET';\nurl='https://google.com/';"
  },
  "responses": {
    "default": " ",
    "GET|/ping": " "
  }
}
```

3. Система мониторинга

В качестве системы мониторинга используется Grafana Loki — это набор компонентов для полноценной системы работы с логами. Loki-стек состоит из трёх компонентов: Promtail, Loki, Grafana. Promtail собирает логи, обрабатывает их и отправляет в Loki. Loki их хранит. A Grafana умеет запрашивать данные из Loki и показывать их. Loki можно использовать не только для хранения логов и поиска по ним. Весь стек даёт большие возможности по обработке и анализу поступающих данных

Чтобы открыть интерфейс системы мониторинга, перейдите в браузере на IP Вашего сервера и порт 3000. Если Вы входите туда в первый раз, используйте логин admin и пароль admin. После первого входа система попросит Вас изменить пароль на безопасный.

Интерфейс выглядит так:



Выберите в меню пункт «Explore» - Вы увидите страницу поиска логов.

Сам запрос состоит из двух частей: selector и filter. Selector — это поиск по индексированным метаданным (лейблам), которые присвоены логам, а filter — поисковая строка или регэксп, с помощью которого отфильтровываются записи, определённые селектором.

Выберите в разделе Label filters в ниспадающем списке Label значение container, а в ниспадающем списке value выберите нужный контейнер. Выполните запрос Run query и Вы увидите логи выбранного контейнера.