Руководство администратора ПО «Система автоматизированной доставки внутрикорпоративных сообщений в персональные чаты сообщества VK»

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 автоматизированной доставки внутрикорпоративных сообщений в персональные чаты сообщества VK» (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

1.3. Уровень подготовки администратора/оператора

Администратор/оператор (далее по тексту Администратор) Системы должен уметь пользоваться и настраивать среду функционирования контейнеров или систему оркестрации, используемую на предприятии.

Рекомендуемая численность персонала для эксплуатации Системы — 1 штатная единица.

Администраторы Системы должны пройти обязательную общую и специальную подготовку для работы с Системой.

Общая подготовка должна включать в себя получение знаний и навыков работы с Системой в качестве администратора.

Специальная подготовка должна включать в себя получение знаний и навыков в объеме, необходимом для выполнения своих должностных обязанностей

1.4. Перечень документации

В состав документации, с которой необходимо ознакомиться администратору Системы входят:

- описание функциональных характеристик Системы
- описание процессов, обеспечивающих поддержание жизненного цикла программного обеспечения.

2. Установка Системы

В данном разделе будет описана установка Системы на Debian Linux. Предполагается, что были предварительно установлены также Docker, Docker Compose, PostgreSQL.

2.1. Системные требования к ПО

Минимальные аппаратные требования:

- Операционная система, способная запускать контейнеры. Предпочтительно Linux.
- Система управления контейнерной виртуализацией. Предпочтительно Docker Swarm или Kubernetes.
- Подключение к СУБД Postgresql
- Количество логических ядер процессора: 4
- Семейство процессоров: x86
- Частота процессора: 3.0. ГГц
- Объем установленной памяти: 16 Гб

2.1.2. Минимальные требования к сторонним компонентам и/или системам, необходимым для установки и работы ПО

- Debian 11 (Открытая лицензия GNU)
- Docker 24.0.2 (open-source community edition)
- Postgresql 14 (Открытая лицензия PostgreSQL license)

При необходимости внешнего логгирования и мониторинга допускается использовать дополнительное сервисное ПО:

  - Grafana Loki 2.6.1 (Открытая лицензия GNU)
  - Grafna 9.2.2 (Открытая лицензия GNU)

2.1.3. Языки программирования

При разработке Системы был использован язык программирования GoLang 1.20 (открытая лицензия BSD)

2.2. Порядок установки

1. Смонтируйте диск с дистрибутивом в папку /mnt
2. Скопируйте из дистрибутива исходники из папки /mnt в папку /root
3. Выполните скрипт install.sql, находящийся в папке /root, на вашем сервере PostgreSQL. Обратите внимание, что установка и настройка сервера PostgreSQL, а также создание базы данных находятся вне компетенции этого документа и не будут тут описаны.
4. Отредактируйте файл docker-compose.yml, в соответствии с пунктами 3.2 данного документа
5. Смените текущую папку на /root и выполните в ней команду
   docker compose -up -d --build

3. Настройка Системы

3.1. Общие сведения

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

3.2. Модуль обработки запросов и доставки сообщений

3.2.1. Конфигурируемые параметры

Для корректной работы модуля обработки запросов и доставки сообщений, необходимо настроить для него следующие переменные окружения:

- HOST - адрес сервиса, который будет слушать модуль. На этот адрес следует пробросить внешний порт или настроить проксирующий сервер для поддержки протокола https.
- VK_TOKEN — токен сообщества VK. Ключ доступа, полученный в разделе настроек «Работа с API» сообщества
- VK_GROUP — имя сообщества VK.
- DB_HOST — адрес сервера PostgreSQL
- DB_PORT— порт сервера PostgreSQL, по умолчанию 5432
- DB_USER — пользователь базы данных
- DB_PASSWORD — пароль пользователя базы данных
- DB_NAME — имя базы данных
- DB_CONN_MAX_TIME — устанавливает максимальное время перед переиспользованием соединения, по умолчанию 3m
- DB_MAX_OPEN_CONNECTIONS — максимальное количество соединений с базой данных, по умолчанию 30
- DB_MAX_IDLE_CONNECTIONS — максимальное количество неиспользуемых соединений с базой данных, по умолчанию 30
- LOG_LEVEL - уровень логгирования. Поддерживаемые значения:
    - error
    - warn
    - info
    - debug
- TEMPLATES_PATH - путь к папке с шаблонами сообщений. По умолчанию /files
- ACCESS_KEY – Bearer-токен доступа к приложению

Пример настройки модуля:

```
  vkbot:
    build:
      context: ./vk/
    restart: always
    ports:
      - '80:80'
    environment:
      HOST: :80
      LOG_LEVEL: debug
      VK_TOKEN: vk1.a29Gh0quqdA
      VK_GROUP: testreed
      DB_HOST: host.docker.internal
      DB_PORT: 5432
      DB_USER: postgres
      DB_PASSWORD: LDP8brN7B8ZLzf5Q879QXuJYXCwm9nP
      DB_NAME: messages_db
      TEMPLATES_PATH: /files
      ACCESS_KEY: valid-key
    volumes:
      - ./vk/files:/files
    extra_hosts:
      - "host.docker.internal:host-gateway"
```

4. Описание API

4.1. Общие сведения

Модуль приема запросов отвечает за взаимодействие с Системой. Модуль принимает HTTP сообщения по протоколу JRPC. В случае, если требуется взаимодействие по протоколу HTTPS, это настраивается внешними, по отношению к Системе, способами. Система принимает сообщения с заголовком Content-Type равным application/json.

4.2. Управление типами сообщений

Система принимает, заранее сконфигурированные, типы сообщений. Для того, чтобы Система могла работать с сообщением, необходимо внести его в базу данных Системы и подготовить соответствующие шаблоны сообщений.

4.2.1. Шаблоны сообщений

Шаблоны сообщений используются для формирования окончательного вида сообщения, которое будет отправлено пользователю. Если соответствующий шаблон сообщения не будет найден — сообщение не будет доставлено. Шаблоны сообщений хранятся в папке, которую использует Система. Конкретное их расположение в файловой системе модуля регулируется переменной окружения TEMPLATES_PATH.

Шаблон представляет собой текстовый файл с любым именем, внутри которого находится сообщение, которое будет отправлено пользователю. Сообщение может содержать именованные параметры, окруженные двойными круглыми скобками. Например: {{test}}. При получении сообщения, Система будет заменять все параметры в шаблоне на пришедшие в сообщении параметры.

4.2.2. Создание типа сообщений

Для создания типа сообщений, надо отправить сообщение subjects.create. Например:

```
{
  "jsonrpc": "2.0",
  "method": "subjects.create",
  "id": 1244560000,
  "params": {
    "subject": "test",
    "params": {
       "vk_template": "test.tpl"
    }
  }
}
```

В качестве параметров сообщения отправляется объект, содержащий параметры:

- subject — тип сообщения или его «тема». Уникальный идентификатор, по которому Система идентифицирует типы сообщений.
- params — параметры типа сообщения. Объект, содержащий необходимые для отправки сообщения настройки.
    - vk_template - имя файла шаблона сообщения.

В качестве ответа, Система возвращает уникальный идентификатор созданного типа сообщений. Этот идентификатор, в дальнейшем, будет использоваться для управления этим типом сообщений. Например:

```
{
   "jsonrpc": "2.0",
   "result": "241b5680-4696-4e70-a8b6-14adcd68971d",
   "id": 1244560000
}
```

4.2.3. Чтение типа сообщений

Для чтения используется метод subjects.read принимающий идентификатор типа сообщений в качестве параметра. Например:

```
{
  "jsonrpc": "2.0",
  "method": "subjects.read",
  "id": 1244560000,
  "params": "241b5680-4696-4e70-a8b6-14adcd68971d"
}
```

В ответе Система возвращает свойства указанного типа сообщений. Например:

```
{
    "jsonrpc": "2.0",
    "result": {
        "subject": "test",
        "params": {
            "vk_template": "test.tpl"
        }
    },
    "id": 1244560000
}
```

4.2.4. Чтение списка типов сообщений в Системе

Метод subjects.list, не принимающий параметров, возвращает список всех имеющихся в системе типов сообщений. Например:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "id": "fa791266-d945-406c-9be5-a74ff95f7e60",
            "subject": "test",
            "params": {
                "vk_template": "test.tpl"
            }
        }    ],
    "id": 1244560000
}
```

4.2.5. Изменение типа сообщений

Для изменения используется метод subjects.update, принимающий парметры параметры:

- id — идентификатор типа сообщения
- subject — тип сообщения или его «тема». Уникальный идентификатор, по которому Система идентифицирует типы сообщений.
- params — параметры типа сообщения. Объект, содержащий необходимые для отправки сообщения настройки.
    - telegram_template - имя файла шаблона сообщения.

Например:

```
{
  "jsonrpc": "2.0",
  "method": "subjects.update",
  "id": 1244560000,
  "params": {
    "id": "241b5680-4696-4e70-a8b6-14adcd68971d",
    "subject": "test1",
    "params": {
       "vk_template": "vk_test.tpl"
    }
  }
}
```

4.2.6. Удаление типа сообщений

Для удаления используется метод subjects.delete принимающий идентификатор типа сообщений в качестве параметра. Например:

```
{
  "jsonrpc": "2.0",
  "method": "subjects.delete",
  "id": 1244560000,
  "params": "241b5680-4696-4e70-a8b6-14adcd68971d"
}
```

4.3. Управление получателями сообщений

Система рассылает сообщения только, заранее настроенным, получателям. Для того, чтобы Система могла доставить сообщение, необходимо найти конкретного пользователя в списке подписчиков сообщества VK и разрешить отправлять ему сообщения.

4.3.1. Чтение списка подписчиков сообщества VK

Метод recipients.list, не принимающий параметров, возвращает список всех имеющихся в сообществе подписчиков. Например:


```
{
   "jsonrpc": "2.0",
   "result": [
      {
         "id": 378131853,
         "caption": "Иванов Иван",
         "approved": false
      },
      {
         "id": 875457255,
         "caption": "Николай Камышов",
         "approved": true
      }
   ],
   "id": 1244560000
}
```


В ответе возвращается массив подписчиков, каждый элемент которого имеет поле id – идентификатор подписчика в VK и булево поле approved, означающее может подписчик принимать сообщения от Системы или нет.

4.3.2. Разрешить подписчику сообщества получать сообщения от Системы

Для записи разрешения получать сообщения, надо отправить сообщение recipients.approve. Например:


```
{
  "jsonrpc": "2.0",
  "method": "recipients.approve",
  "id": 1244560000,
  "params": 875457255
}
```


В качестве параметров сообщения отправляется идентификатор подписчика.

В качестве ответа, Система возвращает булевый результат операции:


```
{
   "jsonrpc": "2.0",
   "result": true,
   "id": 1244560000
}
```

4.3.3. Запрет подписчику сообщества получать сообщения от Системы

Для запрета подписчику получать сообщения, используется метод recipients.disapprove принимающий идентификатор подписчика в качестве параметра. Например:

```
{
  "jsonrpc": "2.0",
  "method": "recipients .read",
  "id": 1244560000,
  "params": 875457255
}
```

В качестве ответа, Система возвращает булевый результат операции:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1244560000
}
```

4.4. Управление маршрутами сообщений

Систему можно настроить таким образом, чтобы часть сообщений отправлялась одной группе пользователей, а часть другой. Это соответствие между типом сообщений и пользователем в Системе называется «маршрут».

4.4.1. Создание или модификация маршрута сообщений

Для создания маршрута, надо отправить сообщение routes.set. Например:

```
{
  "jsonrpc": "2.0",
  "method": "routes.set",
  "id": 1244560000,
  "params": {
    "subject_id": "241b5680-4696-4e70-a8b6-14adcd68971d",
    "recipient_id": 875457255
  }
}
```

В качестве параметров сообщения отправляется объект, содержащий параметры:

- subject_id — идентификатор типа сообщений
- recipient_id — идентификатор получателя сообщений

4.4.2. Чтение списка маршрутов сообщений в Системе

Метод routes.list, не принимающий параметров, возвращает список всех имеющихся в системе маршрутов сообщений. Например:

```
{
   "jsonrpc": "2.0",
   "result": [
      {
         "subject_id": "fa791266-d945-406c-9be5-a74ff95f7e60",
         "recipient_id": 875457255
      }
   ],
   "id": 1244560000
}
```

4.4.3. Удаление маршрута сообщений

Для удаления используется метод routes.delete принимающий идентификатор маршрута сообщений в качестве параметра. Например:

```
{
  "jsonrpc": "2.0",
  "method": "routes.delete ",
  "id": 1244560000,
  "params": {
    "subject_id": "241b5680-4696-4e70-a8b6-14adcd68971d",
    "recipient_id": 875457255
  }
}
```

5.5. Отправка сообщений

После того, как система была настроена, а пользователи авторизованы, можно приступать к отправке сообщений. Для этого надо отправить сообщение message. Например:

```
{
  "jsonrpc": "2.0",
  "method": "message",
  "id": 1244560000,
  "params": {
    "subject": "test",
    "payload": {
       "test": "test"
    }
  }
}
```

В качестве параметров сообщения отправляется объект, содержащий параметры:

- subject — наименование типа сообщений.
- payload — параметры сообщения. Внутри этого объекта должны содержаться параметры, которые будут использованы при формировании текста сообщения на основании шаблона сообщения.