Руководство администратора ПО «Шлюз между REST и SOAP сервисами»

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 «Шлюз между REST и SOAP сервисами» (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 обладать знаниями Javascript, 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.

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.
- 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)

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. Отредактируйте файл docker-compose.yml, в соответствии с пунктом 3.2 данного документа
4. Создайте и отредактируйте файл настроек, в соответствии с пунктаом 3.3 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, а также возможная настройка использования систем оркестрации, находятся вне компетенции этого документа и не будут тут описаны.

3.2. Модуль приема и обработки запросов

Для корректной работы модуля приема и обработки запросов, 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.
- 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.
- WSDL_URL - содержит строку, представляющую URL к файлу WSDL (Web Services Description Language). Этот файл определяет структуру SOAP-сервиса, включая доступные методы, параметры, типы данных и конечные точки.
- SOAP_TIMEOUT — таймаут на исполнение SOAP-запроса.
- LOG_LEVEL - уровень логгирования. Поддерживаемые значения:
    - error
    - warn
    - info
    - debug
    - trace

Пример настройки модуля:

```
 rest2soap:
   build:
    context: ./rest2soap/
   restart: always
   ports:
    - "8080:8080"
   environment:
    WSDL_URL: https://www.crcind.com/csp/samples/SOAP.Demo.CLS?WSDL=1
    SOAP_TIMEOUT: 3
    HOST: :8080
    SETTINGS_FILE: /config.json
    LOG_LEVEL: debug
   volumes:
    - ./config.json:/config.json:ro
   extra_hosts:
    - "host.docker.internal:host-gateway"
```

3.3. 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, назначение которого – обработка поступающих запросов и подготовка параметров к отправке к стороннему SOAP-серверу.

  JavaScript-код должен заполнить хэш таблицу parameters значениями, требуемыми для запроса к конкретному методу SOAP

  JavaScript-код также должен установить значение параметра command названием метода SOAP

  В JavaSctipt-код передаются параметры:

  - body – тело запроса
  - method – HTTP-метод
  - query_params – параметры запроса
  - url – URL запроса

- 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.

  JavaSctipt-код должен заполнить переменную raw_data значением, которое

вернется в теле ответа Системы.

JavaSctipt-код должен заполнить переменную contenttype значением, которое вернется в качестве заголовка Content-type ответа Системы. Значение по умолчанию application/json.

JavaSctipt-код должен заполнить переменную status_code значением, которое вернется в качестве статуса ответа от Системы. Значение по умолчанию 200.

В JavaScript-код передаются параметры:

- raw_data – тело ответа от SOAP-сервера, предварительно преобразованное из xml в json

Пример настройки маршрутизации:

```
{
   "requests": {
      "GET|/numberconversion/:id":"command='AddInteger';
parameters['Arg1']=query_params['id'];  parameters['Arg2']='1';
console.log(JSON.stringify(parameters));"
    },
   "responses": {
      "GET|/numberconversion/:id":"obj = JSON.parse(raw_data); raw_data =
obj['AddIntegerResponse']['AddIntegerResult']; console.log(raw_data); "
   }
}
```