**ROC INTEGRATION**

**Содержание**

1. Техническое описание ПО (Описание, архитектура, дистрибутивы, требования к окружению) 2 **стр**.

2. Руководство пользователя; **9 стр.**

3. Руководство по настройке (развертыванию) и установке; **стр.12**

4. Описание процессов, обеспечивающих поддержание жизненного цикла платформы; **стр. 23**

5. Информация о персонале, необходимом для обеспечения поддержки работоспособности платформы; **стр.24**

6. Прайс-лист; **стр. 25**

1. **Техническое описание ПО**

Roc Integration или платформа интеграции предназначена для построения решений по координации совместной работы информационных систем предприятий и организаций различных сфер деятельности, а также государственных органов власти и местного самоуправления и позволяет обеспечить целостность и надежность доставки данных.

Платформа предоставляет стандартизированный интерфейс для подключения приложений и обеспечивает возможность создания правил маршрутизации и преобразования потоков данных. Встроенная система мониторинга предоставляет инструменты сквозного контроля над функционированием платформы и смежных приложений.

Для эффективной эксплуатации и поддержки платформы необходимы следующие категории персонала:

* администраторы платформы − выполняют администрирование компонентов платформы.

Перед эксплуатацией обслуживающему персоналу необходимо ознакомиться со следующими документами:

* Руководство по развертыванию и настройке Roc Integration;
* Руководство пользователя Roc Integration;
* документация по платформе виртуализации Docker (https://docs.docker.com/).

**Архитектура решения**

В состав платформы входят следующие компоненты:

коннектор входящих соединений – получает обращения от информационной системы источника, преобразовывает обращения в транспортные сообщения и отправляет в поток обработки обращений, при наличии ответа преобразовывает его из транспортного сообщения в протокол взаимодействия с информационной системой и возвращает ответ информационной системе источнику;

коннектор исходящих соединений – получает транспортные сообщения от потока обработки обращений, преобразовывает их из транспортного сообщения в протокол взаимодействия с информационной системой получателем и производит взаимодействие с информационной системой, в случае наличия ответа преобразовывает его в транспортное сообщение и отправляет в поток обработки ответов;

поток обработки обращений – получает от коннектора входящих соединений транспортное сообщение с данными обращения, получает у интерфейса администратора платформы по протоколу SOAP правила по маршрутизации, валидации и трансформации, после чего производит операции маршрутизации, валидации и трансформации обращения и отправляет его в коннектор исходящих соединений;

поток обработки ответов – получает от коннектора исходящих соединений транспортное сообщение с данными ответа, производит операции валидации и трансформации ответа и отправляет его в коннектор входящих соединений;

поток журналирования – получает от потока обработки обращений и потока обработки ответов транспортные сообщения с журналами обработки взаимодействий, преобразует их в формат для хранения и сохраняет в хранилище журналов и конфигураций;

интерфейс администратора платформы – предоставляет возможность управления платформой и просмотра журналов взаимодействия через браузер;

хранилище журналов и конфигураций – система управления базами данных (далее СУБД), в которой на постоянной основе хранится конфигурация платформы и журналы взаимодействия;

транспортная подсистема – менеджер очередей, обеспечивает надежную доставку транспортных сообщений между компонентами платформы.

Коннекторы входящих и исходящих соединений реализованы с использованием Spring Boot + Java 8, потоки обработки обращений, ответов, журналирования реализованы с использованием Spring Boot + Java 8 + Apache Camel. Интерфейс администратора платформы и хранилище журналов и конфигураций представляют собой трехзвенное приложение Spring Boot + Java 8 + jXFW + PostgreSQL. В качестве транспортной подсистемы используется Apache ActiveMQ Artemis.

Схема взаимодействия между компонентами платформы показана на рисунке (Рисунок 1), а также описана в таблице (Таблица 1).



Рисунок 1 – Схема взаимодействия между компонентами платформы интеграции

Таблица 1 – Взаимодействия между компонентами платформы интеграции

| ***№ п/п*** | ***Описание взаимодействия*** |
| --- | --- |
| 1 | Обращение к платформе интеграции от информационной системы инициатора, опционально получение ответа от информационной системы получателя |
| 2 | Передача по транспортной подсистеме в поток обработки обращений преобразованного в транспортное сообщение обращения |
| 3 | Передача по транспортной подсистеме обработанного сообщения в коннектор исходящих соединений |
| 4 | Обращение от платформы интеграции к информационной системе получателю, опционально получение ответа для информационной системы инициатора |
| 5 | Передача по транспортной подсистеме в поток обработки ответов преобразованного в транспортное сообщение ответа (в случае его наличия) |
| 6 | Передача по транспортной подсистеме обработанного ответа (в случае его наличия) в коннектор входящих соединений |
| 7 | Сохранение через поток журналирования основной информации об обработке обращения |
| 8 | Сохранение через поток журналирования основной информации об обработке ответа (в случае его наличия) |
| 9 | Передача изменений конфигурации в поток обработки обращений |
| 10 | Передача изменений конфигурации в поток обработки ответов |
| 11 | Передача журналов об интеграционных взаимодействиях на консоль управления, сохранение в СУБД конфигурации платформы |

**Состав дистрибутива**

Компоненты платформы поставляются в виде подготовленных образов для платформы Docker. Дистрибутив представляет собой один файл в формате zip архива.

В состав дистрибутива входят следующие директории и файлы:

images – директория с образами компонентов платформы для платформы Docker;

config – директория с файлами конфигурации, необходимыми для запуска платформы;

source – директория с исходными кодами HTTP-коннекторов;

readme.txt – текстовый файл с кратким описанием порядка развертывания платформы.

**Требования к окружению**

Дистрибутив должен устанавливаться на выделенный сервер, удовлетворяющий перечисленным в таблице 2 требованиям.

Таблица 2 – Требования к выделенному серверу

| ***Параметр*** | ***Описание*** |
| --- | --- |
| Операционная система | Linux, Windows |
| Процессор | 4хCPU |
| Память | 8GB |
| Диск | 70GB |
| Программное обеспечение | docker 17.03+, docker-compose 1.14.0+ |

# Руководство пользователя

# Коннекторы

Коннекторы входящих и исходящих соединений взаимодействуют с потоками обработки обращений и ответов через транспортную подсистему, представляющую из себя менеджер очередей Apache Artemis версии 1.5.6. Каждый коннектор должен быть подключен к очереди входящих и исходящих транспортных сообщений.

Схема взаимодействия коннекторов с потоками обработки обращений и ответов приведена на рисунке 1.

1

3

1

2

3

 4

2

4

Поток обработки ответов

Тр анспортная

подсистема

Коннектор входящих соединений

Поток обработки обращений

Коннектор исходящих соединений

Рисунок 1 – Схема взаимодействия коннекторов с потоками обработки Требования к взаимодействию между коннекторами и потоками обработки

представлены в таблице 1.

Таблица 1 – Требования к взаимодействию между коннекторами и потоками обработки платформы интеграции

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Название** | **Очередь** | **Требования к сообщениям** |
| 1 | Взаимодействие между коннектором входящих соединений и потоком обработки обращений | Очередь исходящих сообщений коннектора входящих соединений– в данную очередь поступают преобразованные в транспортные сообщения обращения от системы инициатора | Сообщения должны иметь следующие обязательные JMS заголовки:* contentType –тип данных стандарта HTTP, в значении которого символ «/» заменен на символ «\_»;
* responseConnectorUri – адрес входящей очереди коннектора в формате

jms:queueName;* messageType –тип сообщения (может быть пустым), используется при валидации, трансформации и маршрутизации.

Сообщения могут иметь следующие дополнительные заголовки:* JMSCorrelationID – используется для корреляции обращения и ответа системы получателя
 |
| 2 | Взаимодействие между потоком обработки ответов и коннектором входящих соединений | Очередь входящих сообщений коннектора входящих соединений- в данную очередь поступают преобразованные в транспортные сообщения ответы от системы получателя (в случае их наличия) | Сообщения должны иметь следующие обязательные JMS заголовки:* contentType – описание типа данных стандарта HTTP, в котором символ «/» заменен на символ «\_»;
* JMSCorrelationID – используется для корреляции обращения и ответа системы получателя
 |
| 3 | Взаимодействие между потоком обработки обращений и коннектором исходящих соединений | Очередь входящих сообщений коннектора исходящих соединений - в данную очередь поступают преобразованные в транспортные сообщения обращения от системы инициатора | Сообщения должны следующие обязательные JMS заголовки:* contentType –тип данных стандарта HTTP, в значении которого символ «/» заменен на символ «\_»;
* communicationType – тип обращения, request или notification;
* messageType – тип сообщения (может быть пустым)
 |

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Название** | **Очередь** | **Требования к сообщениям** |
| 4 | Взаимодействие между коннектором исходящих соединений и потоком обработки ответов | Очередь исходящих сообщений коннектора исходящих соединений - в данную очередь поступают преобразованные в транспортные сообщения ответы от системы получателя (в случае их наличия) | Сообщения должны иметь следующие обязательные JMS заголовки:* contentType –тип данных стандарта HTTP, в значении которого символ «/» заменен на символ «\_»;
* messageType –тип сообщения (может быть пустым), используется при валидации и трансформации;
* adapterName – имя адаптера обработки ответов, который будет обрабатывать ответ системы получателя (в случае его наличия)
 |

# Руководство пользователя. Диагностика коннекторов

Для диагностики подключения вновь разрабатываемых коннекторов входящих и исходящих соединений с потоками обработки обращений и ответов необходимо:

* + - 1. Проверить на наличие ошибок журналы вновь разрабатываемых коннекторов входящих и исходящих соединений в соответствии с их документацией;
			2. Проверить на наличие ошибок журналы компонентов платформы как описано в подразделе 7 документа «Руководство по развертыванию и настройке»
			3. Проверить корректность настройки маршрутизации, валидации и трансформации обращений в интерфейсе администратора платформы как описано в подразделе 9.3 документа «Руководство по развертыванию и настройке

**3. Развертывание и начало работы**

В процессе развертывания утилита docker-compose создает и запускает контейнеры с компонентами платформы интеграции.

Для развертывания платформы необходимо выполнить следующие действия (ОС Linux):

Подключиться к серверу при помощи ssh-клиента с учетной записью, состоящей в группе docker.

Создать директорию для работы с docker-compose (например, /opt/cip) и скопировать в нее файлы из архива дистрибутива.

При необходимости отредактировать файл docker-compose.yml.

Запустить импорт образов Docker:

 find /opt/cip/images/ -maxdepth 1 -type f -exec docker load -i {} \;

Перейти в директорию /opt/cip/config командой:

 cd /opt/cip/config

Запустить платформу из директории /opt/cip/config:

 docker-compose up -d

Для развертывания платформы необходимо выполнить следующие действия (ОС Windows):

1. Подключиться к серверу с учетной записью с правами локального администратора.

Создать директорию для работы с docker-compose (например, C:\cip) и скопировать в нее файлы из архива дистрибутива.

При необходимости отредактировать файл docker-compose.yml.

Запустить импорт образов Docker в командной оболочке cmd:

 for %i in ("C:\cip\images\"\*) do docker load -i "%i"

Перейти в директорию C:\cip\config командой:

 cd C:\cip\config

1. Запустить платформу из директории C:\cip\config:

 docker-compose up –d

Для упрощения процедуры установки и запуска контейнеров был создан скрипт cip.sh, который входит в поставку.

Для запуска процедуры установки и запуска необходимо запустить скрипт с параметром start

 Sudo ./cip.sh start

Для остановки контейнеров, входящих в поставку необходимо запустить скрипт с параметром stop

 Sudo ./cip.sh stop

**Запуск и остановка компонентов платформы**

Контроль компонентов платформы осуществляется при помощи утилиты docker-compose, при этом все команды выполняются из директории с конфигурационными файлами платформы.

Для запуска платформы необходимо выполнить команду:

docker-compose start

Для остановки платформы необходимо выполнить команду:

docker-compose stop

Для проверки статуса платформы необходимо выполнить команду:

docker ps –a

Все контейнеры должны находиться в статусе «Up».

**Диагностика неисправностей компонентов платформы**

В случае возникновения неисправностей необходимо проверить журналы компонентов платформы. Для вывода журналов всех компонентов необходимо из директории с конфигурационными файлами платформы выполнить следующую команду:

docker-compose logs

Для вывода журналов одного компонента, например, коннектора входящих соединений httpinbound, необходимо из директории с конфигурационными файлами платформы выполнить следующую команду:

docker-compose logs httpinbound

**Доступ к интерфейсам платформы**

**Интерфейс администратора**

Интерфейс администратора реализован на инструментальной среде «ROC Java eXtendable FrameWork»

Доступ к интерфейсу администратора осуществляется по адресу: http://host:port, где host – адрес сервера, port – первое значение параметра ports для сервиса ui в файле docker-compose.yml (по умолчанию 8000). По умолчанию используются учетные данные admin/admin.

**Консоль мониторинга**

Консоль мониторинга реализована на платформе с открытым исходным кодом Grafana, предназначенной для визуализации и анализа данных.

Доступ к консоли мониторинга осуществляется по адресу: http://host:port, где host – адрес сервера, port – первое значение параметра ports для сервиса grafana в файле docker-compose.yml (по умолчанию 3000). По умолчанию используются учетные данные admin/monitoring.

После входа в консоль и выбора при помощи верхнего меню Home страницы с данными мониторинга (JVM или PostgreSQL) консоль мониторинга будет отображать панели с метриками мониторинга компонента платформы, как показано на рисунке 2.



Рисунок 2 – Панели с метриками мониторинга СУБД

**Описание пользовательских настроек платформы**

**Коннектор входящих соединений**

Система – инициатор информационного обмена обращается к коннектору входящих соединений по протоколу HTTP методом POST. Порт коннектора входящих соединений по умолчанию указан в файле docker-compose.yml как первое значение параметра ports для сервиса httpinbound (по умолчанию 8080). В зависимости от URI обращения коннектор выбирает тип обращения. Соответствие URI и типа обращения приведено в таблице 3.

Таблица 3 – Соответствие URI и типа обращения

| **URI** | **Тип обращения** | **Описание** |
| --- | --- | --- |
| /sync/\* | Запрос | Обращение типа запрос предусматривает ответ, коннектор входящих соединений будет ожидать ответа от системы получателя в течение RESPONSE\_TIMEOUT миллисекунд (параметр сервиса httpinbound в файле docker-compose.yml) и вернет ответ системе инициатору |
| /async/\* | Нотификация | Обращение типа нотификация не предусматривает ответ, коннектор входящих соединений закроет соединение с системой инициатором сразу после получения обращения |

Для формирования транспортного сообщения коннектор входящих соединений использует следующие заголовки протокола HTTP:

обязательный HTTP-заголовок обращений Message-Type определяет параметр MessageType транспортного сообщения, который позволяет разбивать обращения на категории и настраивать отдельную логику обработки для каждой из категорий;

обязательный HTTP-заголовок обращений Content-Type определяет тип данных передаваемого тела транспортного сообщения.

**Коннектор исходящих соединений**

Коннектор исходящих соединений при получении транспортного сообщения выполняет запрос по протоколу HTTP методом POST к системе получателю по адресу, указанному в файле docker-compose.yml в качестве параметра CONNECTOR\_URL сервиса httpoutbound, и ожидает ответа от системы-получателя в течение времени, указанного в параметре RESPONSE\_TIMEOUT сервиса httpoutbound в файле docker-compose.yml.

В случае получения ответа от системы-получателя коннектор преобразует ответ в транспортное сообщение и отправляет в поток обработки ответов.

**Настройка потоков обработки обращений и ответов**

Для обеспечения взаимодействия между информационной системой инициатором и информационной системой получателем необходимо настроить правило маршрутизации, которое определяется на странице Маршрутизация. После первоначальной установки платформы правила отсутствуют. Для добавления нового правила необходимо:

1. Открыть страницу «Маршрутизация» и нажать кнопку «Создать». Откроется интерфейс добавления правила маршрутизации (Рисунок 3).



Рисунок 3 – Интерфейс добавления правила маршрутизации

Нажатием на соответствующий пункт контекстного меню создать настройку потока обработки запросов, имеющего название совпадающее по значению с параметром ADAPTER\_NAME сервиса request в файле docker-compose.yml (по умолчанию adapter1), выбрать для него адрес, совпадающий с адресом в параметре INPUT\_QUEUE (по умолчанию jms:requestInputQueue), и сохранить изменения кнопкой ОК.

В поле «Тип сообщения» ввести значение типа отправляемого сообщения, в поле «Класс сообщения» выбрать класс. Доступны следующие варианты выбора:

«Запрос» (предусматривает ответ);

«Нотификация» (не предусматривает ответа).

Нажатием на соответствующий пункт контекстного меню создать настройку нового коннектора, имеющего в качестве URI значение, совпадающее с параметром INPUT\_QUEUE сервиса httpoutbound в файле docker-compose (по умолчанию jms:requestOutputQueue), и сохранить изменения кнопкой ОК.

Сохранить изменения соответствующей кнопкой.

Процессы трансформации и валидации обращений происходят при помощи скриптов трансформации и валидации на языке программирования Groovy. Примеры скриптов приведены в подразделе 9.4.

Для обеспечения трансформации и валидации сообщений необходимо загрузить скрипты валидации и трансформации на странице «Скрипты». После первоначальной установки платформы скрипты отсутствуют.

Для добавления нового скрипта необходимо выполнить следующие действия:

1. Открыть страницу «Скрипты» и нажать кнопку «Создать». Откроется интерфейс добавления скрипта (Рисунок 4).



Рисунок 4 – Интерфейс добавления скрипта

В поле «Название» ввести название скрипта, в полях «Метод» и «Тип» выбрать подходящие значения.

В поле «Время обновления» выбрать время хранения скомпилированного скрипта в кэше платформы, после которого будет происходить его повторная компиляция. Если оставить поле пустым скрипт будет компилироваться при каждом вызове.

Загрузить файл с исходным кодом скрипта.

Для выполнения привязки загруженного скрипта к потоку, в котором он должен выполняться, и к типу сообщений, к которым он должен применяться, необходимо нажатием на соответствующий пункт контекстного меню создать новый объект ValidatorMapping (для привязки скрипта валидации) или TransformerMapping (для привязки скрипта трансформации), выбрав соответствующей кнопкой существующий поток, и заполнить поле «Тип сообщения», после чего сохранить изменения кнопкой ОК.

Сохранить изменения соответствующей кнопкой.

В случае, если информационная система-инициатор не может самостоятельно инициировать запрос, платформа интеграции может по расписанию инициировать запрос к информационной системе-инициатору, получить от нее ответ, после чего этот ответ перенаправить информационной системе-получателю.

Для включения функциональности запросов по расписанию необходимо добавить их на странице «Запросы по расписанию». После первоначальной установки платформы запросы отсутствуют. Для добавления нового запроса необходимо:

1. Открыть страницу «Запросы по расписанию» и нажать кнопку «Создать». Откроется интерфейс добавления запроса по расписанию (Рисунок 5).



Рисунок 5 – Интерфейс добавления запроса по расписанию

В поле «Запрос» загрузить файл с телом запроса, в поле «Название» ввести название запроса.

В поле «Тип контента» ввести значение HTTP заголовка Content-Type, который будет передаваться в систему инициатор.

В поле «Тип сообщения» ввести тип сообщения, который будет передаваться в систему получатель.

В поле «Расписание» ввести значение расписания в следующем формате:

минуты часы дни месяцы дни\_недели

минуты – любое целое число от 0 до 59, или \* если параметр игнорируется;

часы – любое целое число от 0 до 23, или \* если параметр игнорируется;

дни – любое целое число от 1 до 31, или \* если параметр игнорируется;

месяцы – любое целое число от 1 до 12, или \* если параметр игнорируется;

дни\_недели – любое целое число от 0 до 7, где 0 или 7 означает воскресенье, или \* если параметр игнорируется;

Например, расписание:

1 \* \* \* \* \*

будет вызвать запрос каждую минуту.

Нажатием на соответствующий пункт контекстного меню выбрать необходимый коннектор инициатор нотификации, соответствующий коннектору системы инициатора.

Нажатием на соответствующий пункт контекстного меню выбрать поток который будет обрабатывать нотификацию.

Сохранить изменения соответствующей кнопкой.

**Примеры скриптов трансформации и валидации**

Для обеспечения процессов валидации (проверке правильности сообщения), трансформации (преобразования сообщения, например, в другой формат) используются скрипты на языке Groovy, выполняемые в потоках обработки обращений и потоках обработки ответов.

В скрипт валидации передается переменная body с типом данных String, содержащая тело обращения и структура данных headers с типом данных Map, содержащая заголовки транспортного сообщения. На выходе из скрипта ожидается текстовая строка с значениями «true», если сообщение валидно и может быть отправлено дальше по маршруту или «false», если сообщение должно быть проигнорировано.

Пример скрипта валидации, проверяющим наличие тела сообщения:

if (!body)

 {return “false”}

else

 {return “true”}

В скрипт трансформации аналогично передается переменная body с типом данных String, содержащая тело обращения и структура данных headers с типом данных Map, содержащая заголовки транспортного сообщения. Значение, возвращаемое скриптом, помещается в тело сообщения.

Пример скрипта трансформации, преобразующего сообщение c полями name, version, description из формата JSON в формат XML:

import groovy.json.\*
import groovy.xml.\*

def xml = ""

xml = new JsonSlurper().parseText(body).with { j ->    new StringWriter().with { sw ->
        new MarkupBuilder(sw)."$name"(version: version, description:description) {
            params {
                parameters.each { p ->
                    if(p.value instanceof List) {
                        "$p.name"(description:p.description) {
                            p.value.each { v ->
                                "$v.name"(description: v.description, v.value)
                            }
                        }
                    }
                    else {
                        "$p.name"(description:p.description, p.value)
                    }
                }
            }
        }
        sw.toString()
    }
}

return xml

**Перечень условных обозначений, терминов и сокращений**

|  |  |  |
| --- | --- | --- |
| Платформа | − | Roc Integration, платформа интеграции  |
| Docker | − | система виртуализации уровня операционной системы |
| Транспортное сообщение | − | совокупность элементов информации, оформленных в соответствии с требованиями транспортного протокола |
| SOAP | − | Simple Object Access Protocol – [протокол](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB) обмена структурированными сообщениями в распределённой вычислительной среде |
| Маршрутизация | − | процесс определения системы получателя обращения поступившего от системы инициатора |
| Валидация | − | процесс проверка соответствия обращения установленным для него требованиям |
| Трансформация | − | процесс преобразования обращения, например, в другой формат |
| СУБД | − | система управления базами данных |
| Менеджер очередей | − | программный компонент, предоставляющий сервисы очередей сообщений посредством API |
| API  | − | набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением для использования во внешних программных продуктах |
| Очередь | − | именованное хранилище сообщений, управляемое посредством менеджера очередей |
| HTTP | − | HyperText Transfer Protocol – протокол прикладного уровня передачи данных |
| ОС | − | операционная система |
| URI | − | Uniform Resource Identifier – унифицированный (единообразный) идентификатор ресурса |
| JMS | − | Java Message Service, стандарт промежуточного ПО для рассылки сообщений. Используется версия 2.0 |
| Тип сообщения | − | обязательный заголовок транспортного сообщения определяющий |
| Groovy | − | объектно-ориентированный язык программирования, разработанный для платформы Java |
| JSON | − | JavaScript Object Notation – текстовый формат обмена данными, основанный на языке программирования JavaScript |
| XML | − | eXtensible Markup Language – расширяемый язык разметки документов |

**4.Описание процессов, обеспечивающих поддержание жизненного цикла платформы**

Поддержание жизненного цикла Платформы интеграции осуществляется за счёт сопровождения платформы в течении всего периода эксплуатации. Сопровождение платформы также включает проведение модернизаций программного кода платформы по заявкам заказчика с целью расширения функциональности или исправления ошибок в платформе, в случае если это установлено договором, восстановление данных и консультации по вопросам эксплуатации, установки и переустановки платформы.

Сопровождение платформы необходимо для:

* отсутствия простоя в работе организации Заказчика по причине невозможности функционирования платформы (аварийная ситуация, ошибки в работе платформы);
* обеспечения гарантий корректного функционирования платформы и дальнейшего развития её функциональности.

Обозначенные цели достигаются следующим путём:

* консультирования администраторов платформы по вопросам эксплуатации и обновления платформы (по телефону, факсу, электронной почте или при помощи удаленного доступа);
* обеспечение Заказчика новыми версиями платформы по мере их выхода;
* обеспечение Заказчика актуальными версиями эксплуатационной документации;
* устранение ошибок в случае выявления их при работе с платформой.

Модернизация платформы осуществляется путем модификации программного обеспечения по заявкам пользователей, с выпуском новых версий программного обеспечения, полученных в результате модификации, и предоставления Заказчику неисключительных прав на использование новых версий платформы, полученных в результате модернизации.

1. **Информация о персонале, необходимом для обеспечения поддержки работоспособности платформы**

Администраторы системы должны обладать общими знаниями по работе с корпоративными сервисными шинами, навыками администрирования операционных систем семейств Linux или Windows, работе с контейнерами платформы виртуализации Docker.

Для работы достаточно 2 (двух) штатных администраторов системы, которые должны пройти обучение, предоставляемое разработчиком системы при выполнении этапа пуско-наладки платформы, а также изучить следующую документацию:

* Руководство по развертыванию и настройке;
* Руководство по доработке;
* документация по платформе виртуализации Docker (https://docs.docker.com/).
1. **Прайс-лист на продукты и услуги для России**

Действителен на 2023 г.

|  |  |  |
| --- | --- | --- |
| **Наименование программного обеспечения** | **Цена, руб.** | **Примечание\*** |
| Roc Integration | По запросу | Стоимость решения рассчитывается индивидуально |

1. Прайс-лист не является публичной офертой. Указанное программное обеспечение предоставляется конечному пользователю путем заклюения Лицензионного договора (ст. 1235 ГК РФ), согласно пп.1 п.1 ст.1236 ГК РФ в виде права использования (простая (неисключительная) лицензия), в объеме, устанавливаемом ст.1280 ГК РФ.

2. Цены указаны в российских рублях, действуют для расчетов на территории РФ. НДС не облагается на основании пп.26, п.2, ст.149 Налогового кодекса РФ

# Перечень условных обозначений, терминов и сокращений

Платформа − Roc Integration, платформа интеграции

Docker − система виртуализации уровня операционной системы

Коннектор входящих соединений

Коннектор исходящих соединений

Поток обработки обращений

− компонент платформы, получает обращения от информационной системы источника, преобразовывает обращения в транспортные сообщения и отправляет в поток обработки обращений, при наличии ответа преобразовывает его из транспортного сообщения в протокол взаимодействия с информационной системой и возвращает ответ информационной системе источнику

− компонент платформы, получает транспортные сообщения от потока обработки обращений, преобразовывает их из транспортного сообщения в протокол взаимодействия с информационной системой получателем и производит взаимодействие с информационной системой, в случае наличия ответа преобразовывает его в транспортное сообщение и отправляет в поток обработки ответов

− компонент платформы, получает от коннектора входящих соединений транспортное сообщение с данными обращения, получает у интерфейса администратора платформы по протоколу SOAP правила по маршрутизации валидации, трансформации, после чего производит операции маршрутизации, валидации и трансформации обращения и отправляет его в коннектор исходящих соединений

SOAP − Simple Object Access Protocol – протокол обмена структурированными сообщениями в распределённой вычислительной среде

Поток обработки ответов

Транспортная подсистема

− компонент платформы, получает от коннектора исходящих соединений транспортное сообщение с данными ответа, производит операции валидации и трансформации ответа и отправляет его в коннектор входящих соединений

− компонент платформы, менеджер очередей, обеспечивает надежную доставку транспортных сообщений между компонентами платформы интеграции

Менеджер очередей − программный компонент, предоставляющий сервисы очередей

сообщений посредством API

API − набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением для использования во внешних программных продуктах

Очередь − именованное хранилище сообщений, управляемое посредством

менеджера очередей

Транспортное сообщение

− совокупность элементов информации, оформленных в соответствии с требованиями транспортного протокола.

JMS − Java Message Service, стандарт промежуточного ПО для рассылки сообщений. Используется версия 2.0.

Тип сообщения − обязательный заголовок транспортного сообщения определяющий

HTTP − HyperText Transfer Protocol – протокол прикладного уровня передачи данных

Валидация − процесс проверка соответствия обращения установленным для него

требованиям

Маршрутизация − процесс определения системы получателя обращения поступившего от системы инициатора

Трансформация − процесс преобразования обращения, например, в другой формат

URI − Uniform Resource Identifier — унифицированный (единообразный)

идентификатор ресурса