CROC INTEGRATION PLATFORM

Руководство по развертыванию и настройке

Листов 19

Содержание

1 Введение

CROC Integration Platform или Платформа интеграции КРОК (далее платформа) 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 CROC Integration Platform;
- Руководство по доработке CROC Integration Platform;
- документация по платформе виртуализации Docker (https://docs.docker.com/).

2 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.

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

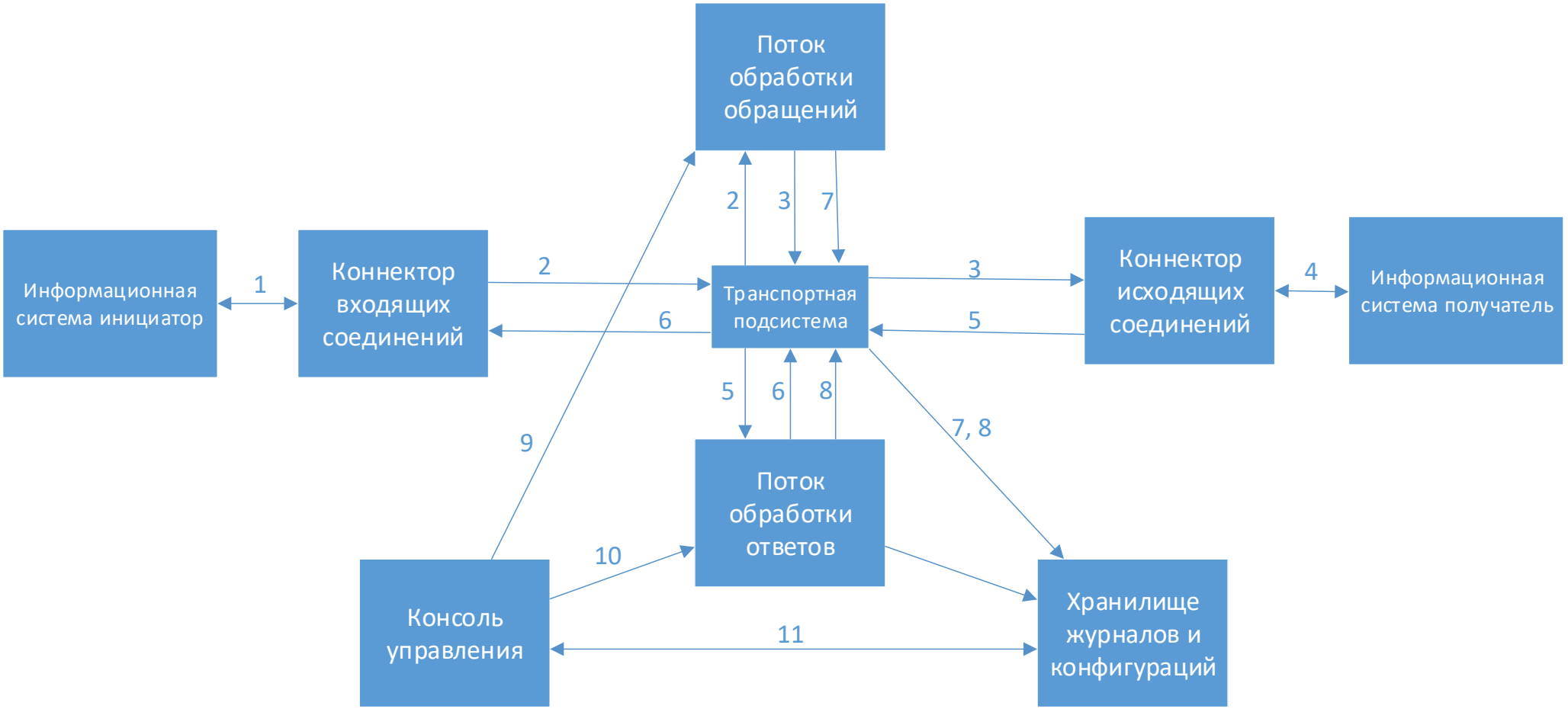

Рисунок 1 – Схема взаимодействия между компонентами платформы интеграции КРОК

Таблица 1 – Взаимодействия между компонентами платформы интеграции КРОК

№ 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

3 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.

4 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	4xCPU
Память	8GB
Диск	70GB
Программное обеспечение	docker 17.03+, docker-compose 1.14.0+

5 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):

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

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

3) При необходимости отредактировать файл docker-compose.yml.

4) Запустить импорт образов Docker:

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

5) Перейти в директорию /opt/cip/config командой:

```
cd /opt/cip/config
```

6) 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.

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

3) При необходимости отредактировать файл docker-compose.yml.

4) Запустить импорт образов Docker в командной оболочке cmd:

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

5) Перейти в директорию C:\cip\config командой:

```
cd C:\cip\config
```

6) Запустить платформу из директории C:\cip\config:

```
docker-compose up -d
```

6 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».

7 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
```

8 Доступ к интерфейсам платформы

8.1 Интерфейс администратора

Интерфейс администратора реализован на инструментальной среде «CROC Java eXtendable FrameWork» (регистрационный номере Едином реестре российских программ для электронных вычислительных машин и баз данных – 4309).

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.

8.2 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.

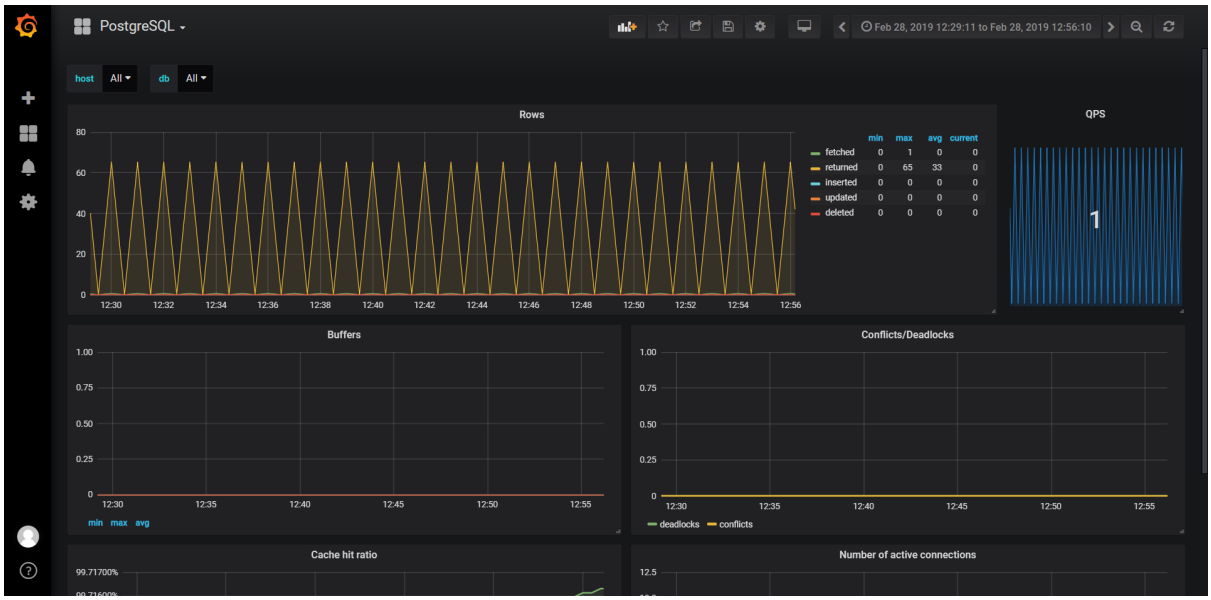

Рисунок 2 – Панели с метриками мониторинга СУБД

9 Описание пользовательских настроек платформы

9.1 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.

9.2 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.

9.3 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).

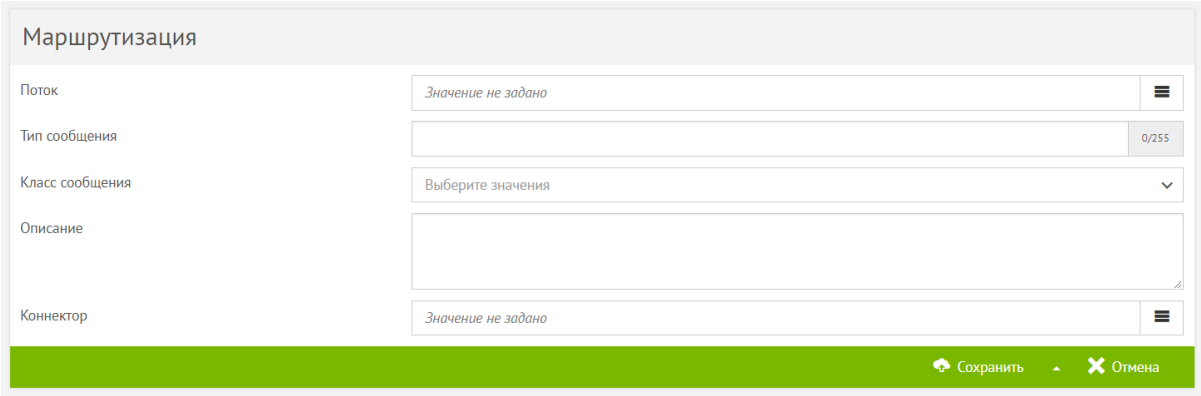

Рисунок 3 – Интерфейс добавления правила маршрутизации

2) 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.

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

а) «Запрос» (предусматривает ответ);
б) «Нотификация» (не предусматривает ответа).

4) 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.

5) 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).

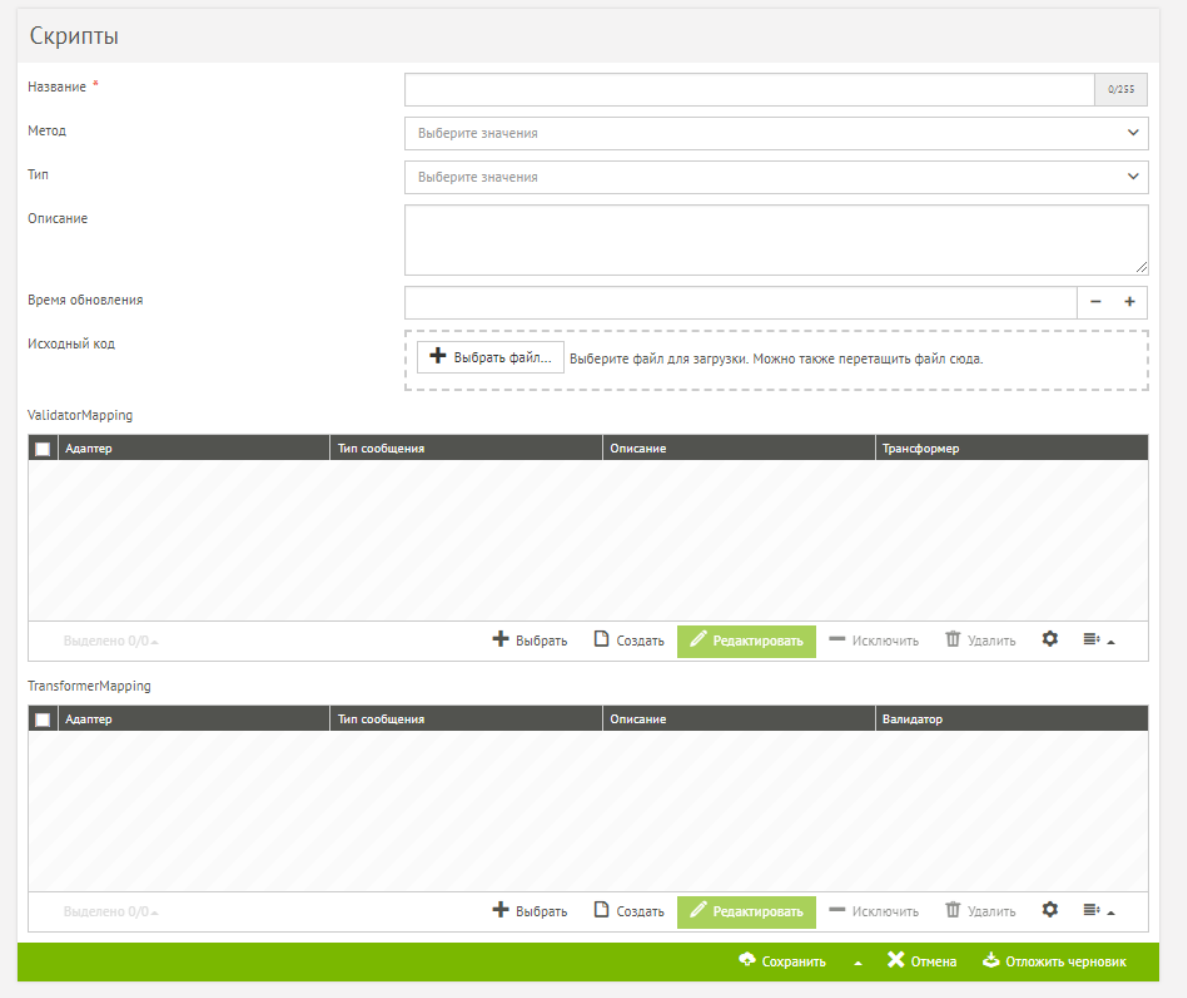

Рисунок 4 – Интерфейс добавления скрипта

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

3) 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.

4) Загрузить файл с исходным кодом скрипта.

5) 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.

6) 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).

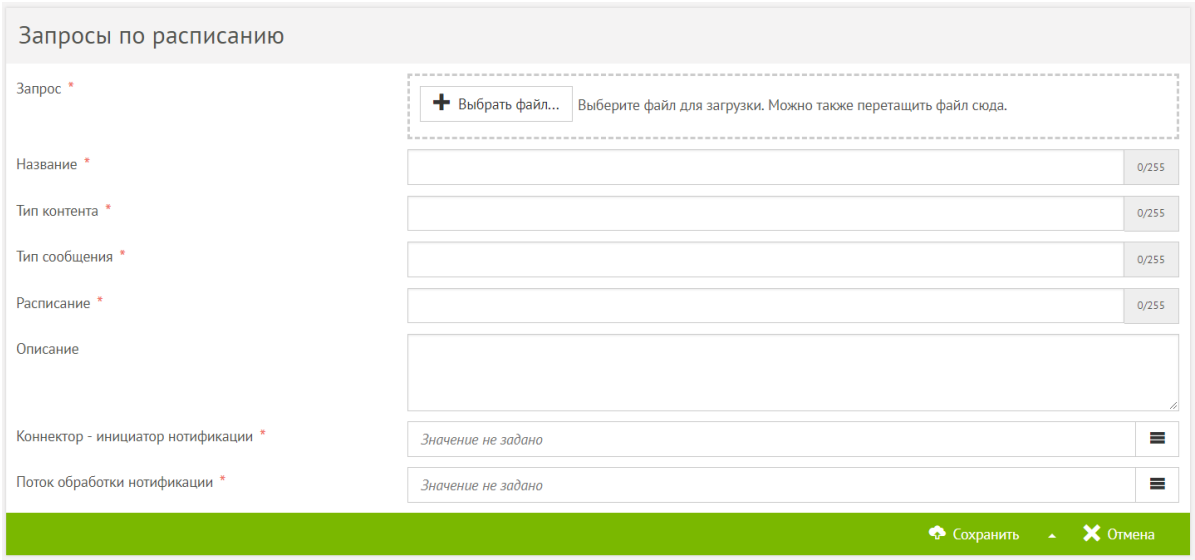

Рисунок 5 – Интерфейс добавления запроса по расписанию

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

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

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

5) 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.

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

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

8) Сохранить изменения соответствующей кнопкой.

9.4 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 с 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	–	CROC Integration Platform, платформа интеграции КРОК
Docker	–	система виртуализации уровня операционной системы
Транспортное сообщение	–	совокупность элементов информации, оформленных в соответствии с требованиями транспортного протокола
SOAP	–	Simple Object Access Protocol – протокол 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