Техническое решение

по взаимодействию между <Шина> и <Клиент>

в части интеграции <Клиент\_1> и <Клиент\_2>

На 8 листах.

# Общая информация

Назначением <Шина> является оптимизация электронного взаимодействия в рамках <Интеграции>;

Взаимодействие между <Клиент\_1> и <Клиент\_2> должно осуществляться через <Шина> и представлять собой обмен сообщениями двух видов:

- Для направления «из <Клиент\_1> в <Клиент\_2>» это сообщение вида <Заявка>

- Для направления «из <Клиент\_2> в <Клиент\_1>» это сообщения вида <Статус>.

Схема взаимодействия предусматривает реализацию на стороне <Шина> основных интеграционных функций:

1. функцию транспорта сообщений с гарантированной доставкой
2. функцию маршрутизации сообщений

Эти функции <Шина> реализованы при помощи продукта Apache ActiveMQ, для чего на стороне <Шина> созданы входящая и исходящая очереди сообщений, используемые <Клиент\_1> и <Клиент\_2> как внешний буфер сообщений.

Настоящий документ описывает механизм взаимодействия между <Клиент> и <Шина>.

# Сценарии взаимодействия <Шина> и <Клиент>

Взаимодействие между <Шина> и <Клиент> представляет собой обмен сообщениями двух видов.

Для направления «из <Клиент> в <ШИНА>» это сообщение вида «Статус»

Для направления «из <ШИНА> в <Клиент>» это сообщения вида «Заявка».

# Описание технической реализации взаимодействия <Клиент> и «Шина)

Работа с менеджером очередей (создание соединения) производится с использованием расширения PHP Client реализации протокола STOMP.

Детальная информация о протоколе доступна по ссылке: <http://stomp.fusesource.org/documentation/index.html>

Расширение PHP Client, в том числе – в виде исходного кода, доступно для скачивания по ссылке: <http://stomp.fusesource.org/download.html>

# Взаимодействие <Клиент> с <Шина> (передача сообщения)

Для передачи сообщения из <Клиент> в <Шина> необходимо выполнить следующие шаги:

* + 1. создать подключение к менеджеру очередей <ШИНА>;
    2. отправить сообщение (поместить сообщение в соответствующую очередь);
    3. закрыть соединение, открытое на шаге 1

Эти шаги детально описаны ниже.

# Шаг 1: Подключение <Клиент> к менеджеру очередей <ШИНА>

Для создания соединения необходимо вызвать функцию, создающую и возвращающую объект типа «соединение» и выполнить метод connect() данного объекта.

Пример подключения к менеджеру очередей <ШИНА>:

// соединение

$con = new Stomp("tcp://etp.sm-soft.ru:61613");

$result = $con->connect();

echo("connect result: $result\n");

// При неудачной попытке соединения возникает исключительная ситуация или возвращается 0,

// При успешном соединении $result==1

# Шаг 2: Отправка сообщения из <Клиент> в <ШИНА>

Для передачи сообщения из <Клиент> в <ШИНА> необходимо поместить его в очередь входящих сообщений от <Клиент> в <ШИНА> – **CLIENT\_1.STATUS**.

Для этого необходимо выполнить метод send() объекта типа «соединение», который был создан и соединение с которым было установлено на первом шаге.

Пример кода передачи сообщения:

$result = $con->send("/queue/CLIENT\_1.STATUS", $msg, array('p'=>'p'), true);

где,

/queue/CLIENT\_1.STATUS - константа для всех передаваемых в менеджер очередей сообщений, определяющая очередь, в которую будут передаваться сообщения

$msg - тело сообщения (строка) в формате **<ссылка на XSD>**

array - атрибуты сообщения, которые могут использоваться для быстрой маршрутизации сообщения в исходящую очередь целевой системы, в текущей схеме обмена передается пустой массив (позднее сюда могут передаваться атрибуты существенные маршрутизации сообщения – номер заявления, тип услуги и т.д.)

true – значение true устанавливает режим отправки сообщения с получением ответа от адаптера STOMP, значение false – без получения ответа. Необходимо использовать значение параметра – true, в противном случае невозможно определить – помещено ли сообщение в очередь по результатам вызова метода.

# Шаг 3: Закрытие соединения

Для закрытия соединения необходимо вызвать метод disconnect() объекта типа «соединение», который был создан и соединение с которым было установлено на первом шаге.

Пример кода закрытия соединения:

$result = $con->disconnect();

# Обработка ошибок и исключительных ситуаций

При неудачной попытке создания соединения, отправки сообщения или закрытия соединения (при выполнении вызова соответствующего метода) возникает исключительная ситуация.

В противном случае метод возвращает код успешного завершения метода - **0**.

В журнале <Клиент> необходимо фиксировать:

* + 1. **все** возникающие в процессе выполнения сценария «Передача Заявки» исключительные ситуации;
    2. **все** случаи, когда любая функция расширения PHP Client реализации протокола STOMP возвращает код завершения, отличный от успешного.

# Взаимодействие <Клиент> с <ШИНА> (получение сообщения)

<Клиент> получает заявки из исходящей очереди обработчиком, который должен запускаться при поступлении сообщения в очередь исходящих сообщений для <Клиент> в <ШИНА> – **CLIENT\_2.APPLICATION**.

# Шаг 1: Подключение <Клиент> к менеджеру очередей <ШИНА>

Для создания соединения необходимо вызвать функцию, создающую и возвращающую объект типа «соединение» и выполнить метод connect() данного объекта (аналогично п.3.1.1).

Пример подключения к менеджеру очередей <ШИНА>:

// соединение

$con = new Stomp("tcp://etp.sm-soft.ru:61613");

$result = $con->connect();

echo("connect result: $result\n");

// При неудачной попытке соединения возникает исключительная ситуация или возвращается 0,

// При успешном соединении $result==1

# Шаг 2: Получения сообщения в <Клиент> из <ШИНА>

Для получения сообщения, необходимо выполнить операцию «подписки» на соответствующую очередь « **CLIENT\_2.APPLICATION».**

$con->subscribe("/queue/**CLIENT\_2.APPLICATION**", array('ack' => 'client'), true);

Где, указан параметр ack=true (подтверждение прочтения сообщения производится клиентом шины в «ручном» режиме с помощью соответствующего вызова message->ack).

После прочтения тела сообщения и его успешной обработки (сохранения в БД <Клиент>), сообщение удаляется из очереди вызовом метода $con->ack($msg);

Пример получения сообщения:

// соединение

$con = new Stomp("tcp://etp.sm-soft.ru:61613");

$result = $con->connect();

echo("connect result: $result\n");

// При неудачной попытке соединения возникает исключительная ситуация или возвращается 0,

// При успешном соединении $result==1

$con->subscribe("/queue/**CLIENT\_2.APPLICATION** ", array('ack' => 'client'), true);

$con->readFrame(); // читаем тело сообщения

// проводим обработку сообщения, если при обработке не возникло никаких исключительных ситуаций подтверждаем получение сообщения из исходящей очереди

$con->ack($msg);

// отписываемся от очереди

$con->unsubscribe("/queue/**CLIENT\_2.APPLICATION**", array(), true);

# Шаг 3: Закрытие соединения

Для закрытия соединения необходимо вызвать метод disconnect() объекта типа «соединение», который был создан и соединение с которым было установлено на первом шаге (аналогично шагу 3.1.3).

Пример кода закрытия соединения:

$result = $con->disconnect();

# Рекомендации по реализации транспортной подсистемы <Клиент>

Соединение с менеджером очередей осуществляется с использованием протокола STOMP http://stomp.fusesource.org/documentation/php/book.html. Для соединения вызывается функция, которая возвращает объект соединения. Во время использования STOMP не требуется отдельного открытия и поддержания ссылок на открытые очереди, метод send получает имя очереди и тело сообщения для отправки.

Необходимо посылать сообщения в режиме с получением ответа от адаптера STOMP, иначе без получения подтверждения нет возможности понять, дошло ли сообщение во входящую очередь. 4-й параметр вызова функции send - обязательно должен быть выставлен в true.

Транспортный блок необходимо реализовать в виде отдельного модуля, который инкапсулирует все механизмы связи с очередью, чтобы обеспечить: единый механизм обработки ошибок, возможность менять константы и логику соединения в одном месте для всех взаимодействий <ШИНА> -> <Клиент>.

# Форматы данных

# Номер заявки и единый номер обращения

Запрос от <Клиент\_1> к <Клиент\_2> идентифицируется уникальным номером. При поступлении сообщения в очередь исходящих сообщений для <Клиент> в <ШИНА> – **CLIENT\_1.STATUS** (сообщения обновления статуса), <ШИНА> кроме номера заявки возвращает единый номер обращения, сформированный в следующем формате:

xxxx-yyyy-zzzzzzz-nnnnnnn/yy

xxxx - код ИС <Клиент\_1> как организации, где создано обращение

yyyy– код ИС <Клиент\_2> как получателя сообщения

zzzzzz – код процесса по реестру процессов шины

nnnnnnn – порядковый номер сообщения

yy - текущий год

Примеры:

0001-0002-000101-2242460/16

<Клиент>, при получении сообщения со статусом заявки, сохраняет в БД <Клиент> присвоенный заявке единый номер обращения.

Описание реализации сервиса по назначению единого номера обращения выполнено отдельным документом «Назначение единого номера обращения (ЕНО). Техническое решение».

# Формат передачи сообщений

Сообщения, которыми обмениваются <Клиент> и <ШИНА> должны быть сформированы в формате XML.

xsd-схема формата версии V0 приведена в Приложении 1 к настоящему документу.

# Доступ к стендам

# Тестовые стенды

Тестовый стенд <ШИНА>:

IP-адрес тестового стенда: etp.sm-soft.ru

Порт тестового стенда:61613

Авторизация отключена

Тестовый стенд принимает соединения по протоколу STOMP

Тестовый стенд <Клиент>:

??? внести параметры тестового стенда <Клиент>

# Приложение 1. xsd-схема описания формата

<?xml version="1.0" encoding="UTF-8"?>