Руководство пользователя

“Сервис отправки уведомлений при изменении статусов операций”

Москва, 2023 г.

1. Введение

Наименование программного обеспечения “Сервис отправки уведомлений при изменении статусов операций”.

1.1 Область применения

“Сервис отправки уведомлений при изменении статусов операций” представляет собой техническое решение, *B2B сервис*, предоставляющий возможность отправлять уведомления пользователям при изменении статуса операций на предоставленный «url».

1.2 Краткое описание возможностей

- Возможность масштабирования благодаря наличию ресурсов для запуска нескольких экземпляров сервиса;
- Штатный функционал позволяет подписать *N инстансов* на одну и ту же очередь, сообщения из которой будут случайным образом приходит в тот или иной инстанс;
- Расшифровка и трансформация полученной информации;
- Передача данных в виде запросов в Базу данных и получение этих данных;
- Формирование запроса в конкретную платежную систему в определенном формате.

1.3 Уровень подготовки пользователей

Для интеграции API пользователь должен иметь квалификацию разработчика не ниже уровня Regular Middle.

1.4 Перечень программной документации

- “Сервис отправки уведомлений при изменении статусов операций”. Техническое задание (ГОСТ 19.201-78);
- “Сервис отправки уведомлений при изменении статусов операций”. Руководство пользователя;

- “Сервис отправки уведомлений при изменении статусов операций”. Инструкция по развертыванию экземпляра ПО;
- “Сервис отправки уведомлений при изменении статусов операций”. Описание жизненного цикла разработки ПО.

1.5 Термины и сокращения

API – описание способов взаимодействия одной компьютерной программы с другими.

БД – база данных – набор структурированных данных, хранящихся в виде таблицы.

Инстанс – экземпляр класса (объекта) в объектно-ориентированном программировании.

Мерчант – партнер, пользователь программного обеспечения “Сервис отправки уведомлений при изменении статусов операций”.

Очередь – это некая структура данных, которая обеспечивает хранение и передачу двоичных данных между различными участниками системы.

Сервис – независимо компилируемый программный модуль, динамически подключаемый к основной программе и предназначенный для расширения и/или использования её возможностей. Сервисы обычно выполняются в виде библиотек общего пользования.

Платежный провайдер – компания, которая предоставляет онлайн-сервисы по осуществлению электронных платежей различными способами. Некоторые поставщики платежных сервисов предоставляют различные инновационные сервисы.

ПО – программное обеспечение.

2. Назначение и требования для эксплуатации ПО

Программное обеспечение “Сервис отправки уведомлений при изменении статусов операций” предназначено для интеграции и дальнейшей возможности использования у любого Заказчика.

2.1 Функциональное назначение

ПО представляет собой техническое решение, позволяющее расшифровать полученные запросы, трансформировать полученную информацию и передать полученные данные в виде запроса в конкретную платежную систему в определенном формате.

2.2 Эксплуатационное назначение

ПО интегрируется и эксплуатируется на стороне Заказчика и предоставляет следующие преимущества при использовании сервиса:

- ***Независимость*** компонентов системы друг от друга. Благодаря использованию очереди, компоненты взаимодействуют через некий общий интерфейс, но ничего не знают о существовании друг друга.
- ***Экономия ресурсов*** достигается вследствие возможности разумно распределять информацию, поступающую в очередь от одних процессов, между другими процессами, осуществляющими ее обработку. Кроме того, благодаря тому, что нет необходимости промежуточного хранения необработанных данных, достигается дополнительная экономия ресурсов.
- ***Надежность*** очередей достигается благодаря возможности накапливать сообщения, амортизируя недостаток вычислительных возможностей системы, а также благодаря независимости компонентов. Помимо этого очередь может аккомодировать сбои отдельных компонентов, осуществляя доставку «опоздавших» сообщений после восстановления.
- И, наконец, ***гарантия последовательной обработки***, позволяющая точно контролировать потоки данных в системе и запускать асинхронную обработку там, где это необходимо, не беспокоясь, что одна операция выполнится раньше другой, от результата которой она зависит.

2.3 Требования к эксплуатации серверной части

Обеспечение функционирования ПО серверной части “Сервис отправки уведомлений при изменении статусов операций” реализовано на базе серверной операционной системы Linux. Минимальной конфигурацией аппаратной составляющей являются:

- Современная ОС: Linux;
- Оперативная память: 2 ГБ;
- Свободное дисковое пространство: не менее 20 Gb;

- Количество логических ядер процессора: 2;
- Частота процессора: 3.50 GHz.

Возможно разворачивание экземпляра ПО и на других ОС, поддерживающих платформу для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации Docker, например Windows 10 (Профессиональная или Корпоративная).

3. Подготовка к работе

3.1 Подготовка

Необходимо предоставить `url` адрес в платежном сервисе, на который будет приходить уведомления.

3.2 Формат запросов

Оповещения передаются на POST запросов на url, предоставляемый при подключении платформы. Важно проверять X-Data-Hash запроса.

3.2.1 HTTP-заголовки

Обязательные к передаче заголовки HTTP запроса:

`Content-Type: application/json`

`X-Data-Hash`

3.2.2 X-Data-Hash

Используется для подтверждения легитимности и подлинности запроса. Передается в HTTP заголовке `X-Data-Hash`.

`X-Data-Hash = sha512(%request_body% + %secret_key%)`

- `request_body` - объект запроса, сериализованный в виде JSON строки (берем исходную строку).
- `secret_key` - секретный ключ, выдаваемый при подключении приложения.

На выходе получаем ЭЦП к запросу.

Проверка производится следующим образом:
`isValidRequest = X-Data-Hash == sha512(%request_body% + %secret_key%)`

- `isValidRequest == true` - запрос легитимный;
- `isValidRequest == false` - запрос следует проигнорировать.

3.3 Параметры

Параметр	Тип	Обязательный	Описание
`method`	string	+	Название метода (payment.update)
`params`	object	+	Объект с результатом изменения статуса транзакции
`params.request_id`	string	-	Идентификатор запроса
`params.payment`	Payment	+	Информация о транзакции
`params.payment.service_id`	integer	+	Идентификатор платежной системы
`params.payment.amount`	Amount	+	Объект суммы и валюты платежа
`params.payment.amount.value`	integer	+	Сумма платежа

`params.payment.amount.currency`	string(3)	+	Валюта платежа
`params.payment.identifiers`	Identifier	+	Объект с идентификаторами платежа
`params.payment.identifiers.c_id`	integer	+	Идентификатор платежа на стороне customer
`params.payment.identifiers.h_id`	integer	+	Идентификатор платежа на стороне PayHub
`params.payment.identifiers.p_id`	string	-	Идентификатор платежа на стороне платежной системы
`params.payment.payer`	Payer	-	Объект с платежными данными пользователя
`params.payment.payer.card`	Card	-	Карточные данные при платеже
`params.payment.payer.card.id`	string	-	Карточный токен при платеже
`params.payment.payer.card.mask`	string	-	Маска карты при платеже
`params.payment.payer.card.holder`	string	-	Имя Фамилия владельца карты при платеже
`params.payment.payer.payment_system_account`	Account	-	Объект с аккаунтом пользователя платежной системы при платеже
`params.payment.payer.payment_system_account.id`	string	+	Идентификатор аккаунта пользователя

			платежной системы при платеже
`params.payment.receiver`	Receiver	-	Объект с данными пользователя при выплате
`params.payment.receiver.card`	Card	-	Карточные данные при выплате
`params.payment.receiver.card.id`	string	-	Карточный токен при выплате
`params.payment.receiver.card.mask`	string	-	Маска карты при выплате
`params.payment.receiver.card.holder`	string	-	Имя Фамилия владельца карты при выплате
`params.payment.receiver.payment_system_account`	string	-	Объект с аккаунтом пользователя платежной системы при выплате
`params.payment.receiver.payment_system_account.id`	string	+	Идентификатор аккаунта пользователя платежной системы при выплате
`params.payment.status`	Status	+	Объект статуса транзакции
`params.payment.status.status`	string	+	Статус транзакции
`params.payment.status.final`	boolean	+	Финальный / не финальный статус транзакции
`params.payment.status.success`	boolean	+	Успешный / неуспешный статус транзакции

`params.payment.status.error`	Object	-	Объект с ошибкой
`params.payment.status.error.code`	integer	+	Код ошибки PayHub
`params.payment.status.error.message`	string	+	Сообщение ошибка
`params.payment.status.history[]`	TransactionHistory	+	Объект с историей статуса транзакций
`params.payment.status.history[].status`	string	+	Объект с историей статуса транзакций
`params.payment.status.history[].final`	boolean	+	Финальный / не финальный статус транзакции
`params.payment.status.history[].success`	boolean	+	Успешный / неуспешный статус транзакции
`params.payment.status.history[].created`	string	+	Дата и время установление статуса транзакции
`params.payment.status.history[].reason`	Object	-	Объект с ошибкой
`params.payment.status.history[].reason.code`	integer	+	Код ошибки PayHub
`params.payment.status.history[].reason.message`	string	+	Сообщение ошибка
`params.payment.timestamps`	Timestamps	+	Объект с датой и времени создания, изменения и финализации транзакции
`params.payment.timestamps.created`	string	+	Дата и время создание

			транзакции
`params.payment.timestamps.updated`	string	+	Дата и время изменения транзакции
`params.payment.timestamps.finished`	string	-	Дата и время финализации транзакции
`params.payment.operations[]`	Operation	+	Объект с данными операций транзакции
`params.payment.operations[].id`	integer	+	Идентификатор операции
`params.payment.operations[].operation_type`	string	+	Тип операции
`params.payment.operations[].amount`	Amount	+	Объект суммы и валюты операции
`params.payment.operations[].amount.value`	integer	+	Сумма операции
`params.payment.operations[].amount.currency`	string(3)	+	Валюта операции
`params.payment.operations[].timestamps`	Timestamps	+	Объект с датой и времени создания, изменения и финализации операции
`params.payment.operations[].timestamps.created`	string	+	Дата и время создание операции
`params.payment.operations[].timestamps.updated`	string	+	Дата и время изменения операции
`params.payment.operations[].timestamps.finished`	string	-	Дата и время финализации

			операции
`params.payment.operations[].status`	Operation Status	+	Объект статуса операции
`params.payment.operations[].status.status`	string	+	Статус операции
`params.payment.operations[].status.final`	boolean	+	Финальный / не финальный статус операции
`params.payment.operations[].status.success`	boolean	+	Успешный / неуспешный статус операции
`params.payment.operations[].status.error`	Object	-	Объект с ошибкой
`params.payment.operations[].status.error.code`	integer	+	Код ошибки PayHub
`params.payment.operations[].status.error.message`	string	+	Сообщение ошибка
`params.payment.operations[].fees[]`	Fee	-	Массив суммы и валюты платежа
`params.payment.operations[].fees[].value`	integer	+	Сумма комиссии
`params.payment.operations[].fees[].currency`	string(3)	+	Валюта комиссии
`params.payment.operations[].fees[].type`	string	+	Тип комиссии
`params.operation`	Operation	+	Объект с данными текущей операции транзакции
`params.operation.id`	integer	+	Идентификатор текущей операции

`params.operation.operation_type`	string	+	Тип текущей операции
`params.operation.amount`	Amount	+	Объект суммы и валюты текущей операции
`params.operation.amount.value`	integer	+	Сумма текущей операции
`params.operation.amount.currency`	string(3)	+	Валюта текущей операции
`params.operation.timestamps`	Timestamps	+	Объект с датой и времени создания, изменения и финализации текущей операции
`params.operation.timestamps.created`	string	+	Дата и время создание текущей операции
`params.operation.timestamps.updated`	string	+	Дата и время изменения текущей операции
`params.operation.timestamps.finished`	string	-	Дата и время финализации текущей операции
`params.operation.status`	Operation Status	+	Объект статуса текущей операции
`params.operation.status.status`	string	+	Статус текущей операции
`params.operation.status.final`	boolean	+	Финальный / нефинальный статус текущей операции
`params.operation.status.success`	boolean	+	Успешный / неуспешный статус текущей операции

`params.operation.status.error`	Object	-	Объект с ошибкой
`params.operation.status.error.code`	integer	+	Код ошибки PayHub
`params.operation.status.error.message`	string	+	Сообщение ошибка
`params.operation.fees[]`	Fee	-	Массив суммы и валюты платежа
`params.operation.fees[].value`	integer	+	Сумма комиссии
`params.operation.fees[].currency`	string(3)	+	Валюта комиссии
`params.operation.fees[].type`	string	+	Тип комиссии
`params.payment.redirect`	Redirect	-	Объект с информацией о переадресациях пользователя
`params.payment.redirect.to`	string	-	ссылка для редиректа для совершения платежа / выплаты
`params.payment.redirect.on_success`	string	-	ссылка для редиректа в случае успеха
`params.payment.redirect.on_fail`	string	-	ссылка для редиректа в случае ошибки/отказа/

3.3.1 Пример успешного уведомления


```
{ "method": "payment.update", "params": { "payment": { "receiver":
{ "card": { "id":
"a669fcbcdda5fc802624a01f3cc38d1c86f522521d309ab3fa8f330fc84cc71a",
"mask": "111111******1234", "holder": "TEST HOLDER" } }, "amount":
{ "value": 70000, "currency": "USD" }, "description": null,
"identifiers": { "c_id": 9914, "h_id": 12645, "p_id": null },
"status": { "status": "success", "final": true, "success": true,
"error": null, "history": [ { "status": "created", "final": false,
"success": null, "created": "2020-04-09T09:57:02.064Z", "reason":
null, "amount": 70000 }, { "status": "processing", "final": false,
"success": null, "created": "2020-04-09T09:57:02.360Z", "reason":
null, "amount": 70000 }, { "status": "success", "final": true,
"success": true, "created": "2020-04-09T09:57:06.435Z", "reason":
null, "amount": 70000 } ] }, "timestamps": { "created": "2020-04-
09T09:57:02.064Z", "updated": "2020-04-09T09:57:06.437Z",
"finished": "2020-04-09T09:57:06.437Z" }, "destination": "out",
"operations": [ { "id": "22981", "operation_type": "payment.out",
"amount": { "value": 70000, "currency": "USD" }, "timestamps": {
"created": "2020-04-09T09:57:02.105Z", "updated": "2020-04-
09T09:57:06.464Z", "finished": "2020-04-09T09:57:06.464Z" },
"status": { "status": "success", "final": true, "success": true,
"error": null }, "fees": [ { "type": "FROM_PROVIDER_TO_CUSTOMER",
"value": 1332, "currency": "USD" } ] } ], "service_id": 201 },
"operation": { "id": "22981", "operation_type": "payment.out",
"amount": { "value": 70000, "currency": "USD" }, "timestamps": {
"created": "2020-04-09T09:57:02.105Z", "updated": "2020-04-
09T09:57:06.464Z", "finished": "2020-04-09T09:57:06.464Z" },
"status": { "status": "success", "final": true, "success": true,
"error": null }, "fees": [ { "type": "FROM_PROVIDER_TO_CUSTOMER",
"value": 1332, "currency": "USD" } ] } } }
```


4. Аварийные ситуации

В случае аварийных ситуаций обращаться в техническую поддержку.

5. Рекомендации по освоению

Для успешной работы с ПО “Сервис отправки уведомлений при изменении статусов операций”

необходимо:

- Иметь квалификацию разработчика не ниже уровня Regular Middle;
- Иметь оборудованное рабочее место с подключением к сети Интернет;
- Рабочее место должно соответствовать минимальным требованиям, указанным в разделе “2.3 Требования к эксплуатации серверной части” данного документа;
- Ознакомиться с документом “Руководство пользователя “Сервис отправки уведомлений при изменении статусов операций”;
- Ознакомиться с документом “Разворачивание экземпляра ПО “Сервис отправки уведомлений при изменении статусов операций”.