SOFT-LOGIC

Paylogic

Протокол взаимодействия между системой мерчанта и платежным виджетом Pay-logic. Программное обеспечение «PSP Pay-logic».

Руководство разработчика

АННОТАЦИЯ

Содержит описание принципов работы, настройки и интеграции в системы партнеров

Версия руководства: 1.8

Руководство актуально для кабинета мерчанта версий 1.11

ООО «Софт-Лоджик»

656006, г. Барнаул, Малахова ул., дом 146в

Тел: (3852) 72-27-27

ОГЛАВЛЕНИЕ

ИСТОРИЯ ИЗМЕНЕНИЙ 5

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.0 5
ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.1 5
ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.2 5
ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.3 6
ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.4 6
ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.5 6
ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.6 7
ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.7 7
ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.8 7

1 ТЕРМИНЫ И СОКРАЩЕНИЯ 9

2 ПРЕДПОЛАГАЕМАЯ АУДИТОРИЯ 10

3 ВВЕДЕНИЕ 11

4 ОБЩИЕ ПОЛОЖЕНИЯ 12

5 ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ 15

5.1 СЕТЕВОЕ ВЗАИМОДЕЙСТВИЕ 15
5.2 АВТОРИЗАЦИЯ 15

6 СОЗДАНИЕ API ТИПА ПЛАТЕЖНЫЙ ВИДЖЕТ 17

7 ИНТЕГРАЦИЯ В СИСТЕМУ ПАРТНЕРА 20

7.1 ЗАКАЗ НА ОПЛАТУ 20
7.1.1 ОБЩАЯ ИНФОРМАЦИЯ 20
7.1.2 СОЗДАНИЕ ЗАКАЗА НА ОПЛАТУ 23
7.1.3 ПРЕДЗАПОЛНЕНИЕ ДАННЫХ В ЗАКАЗЕ 27
7.1.4 ОПЛАТА ЗАКАЗА ЧЕРЕЗ SBERPAY 28

7.2 ЗАКАЗ НА ВЫПЛАТУ 29
7.2.1 ОБЩАЯ ИНФОРМАЦИЯ 29
7.2.2 СОЗДАНИЕ ЗАКАЗА НА ВЫПЛАТУ 31
7.3 БАЛАНС МЕРЧАНТА 35
7.4 УВЕДОМЛЕНИЕ ПАРТНЕРА О СТАТУСЕ ОПЕРАЦИИ 36

8 ПРИЛОЖЕНИЕ 41
A. ПРИМЕРЫ ГЕНЕРАЦИИ И ПРОВЕРКИ ЭП НА JAVA 41
B. ПРИМЕРЫ ЗАПРОСОВ И ОТВЕТОВ ДЛЯ МЕТОДА СОЗДАНИЯ ЗАКАЗА НА ОПЛАТУ 43
C. ПРИМЕР ЗАПРОСА ДЛЯ МЕТОДА СОЗДАНИЯ ЗАКАЗА НА ОПЛАТУ С ПРЕДЗАПОЛНЕННЫМИ ДАННЫМИ 44
D. ПРИМЕРЫ ЗАПРОСА И ОТВЕТА ДЛЯ МЕТОДА СОЗДАНИЯ ЗАКАЗА НА ВЫПЛАТУ 45
E. ПРИМЕР ОТВЕТА НА ЗАПРОС БАЛАНСА МЕРЧАНТА 46
F. ПРИМЕР WEBHOOK-ОПОВЕЩЕНИЯ 47

ИСТОРИЯ ИЗМЕНЕНИЙ

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.0

Дата публикации: 21.05.2025 г.

Изменение	Раздел
Общие улучшения:	
Документ создан	-

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.1

Дата публикации: 22.09.2025 г.

Изменение	Раздел
Общие улучшения:	
Добавлен раздел **«Предзаполнение данных в заказе»**	7.2.1

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.2

Дата публикации: 01.11.2025 г.

Изменение	Раздел
Общие улучшения:	
Обновлено описание **параметров ответа** на запрос создания заказа	7.2
Добавлено описание **параметров** при использовании способа оплаты	7.2.2

Изменение	Раздел
SberPay	

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.3

Дата публикации: 07.11.2025 г.

Изменение	Раздел
Общие улучшения:	
Обновлено описание создания API	6
Добавлено возможное значение **параметра source**	7.2

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.4

Дата публикации: 10.11.2025 г.

Изменение	Раздел
Общие улучшения:	
Обновлено название документа	-

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.5

Дата публикации: 26.11.2025 г.

Изменение	Раздел
Общие улучшения:	

Изменение	Раздел
Добавлено возможное значение **параметра source**	7.2

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.6

Дата публикации: 03.12.2025 г.

Изменение	Раздел
Общие улучшения:	
Общие улучшения документа	-

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.7

Дата публикации: 19.12.2025 г.

Изменение	Раздел
Общие улучшения:	
Добавлен раздел Создание заказа на выплату	7.2

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.8

Дата публикации: 16.02.2026 г.

Изменение	Раздел
Общие улучшения:	

Изменение	Раздел
Добавлено описание метода **запроса баланса мерчанта**	7.3

1 ТЕРМИНЫ И СОКРАЩЕНИЯ

Мерчант — юридическое лицо, занимающееся продажей товаров или предоставлением услуг и принимающее оплату через электронные платежные системы.

Платежный виджет — форма оплаты в системе партнера, с помощью которой клиент производит оплату.

ПЦ — процессинговый центр.

СБП — система быстрых платежей.

ЭП — электронная подпись.

PCI DSS (анг. Payment Card Industry Data Security Standard) — стандарт безопасности данных платежных карт, разработанный для защиты данных на протяжении всего жизненного цикла платежа, с использованием технологических решений, обесценивающих эти данные для кражи преступниками.

URL (англ. Uniform Resource Locator) — стандартизированный способ записи адреса ресурса в сети Интернет.

2 ПРЕДПОЛАГАЕМАЯ АУДИТОРИЯ

Данное руководство предназначено для использования в качестве спецификации для настройки и интеграции платежного виджета в систему мерчанта.

Документ адресован IT-специалистам (программистам и администраторам), обслуживающим информационную систему поставщика услуг и знакомым с технологиями разработки ПО, языками программирования, основами работы с XML и сетевым взаимодействием по HTTP/HTTPS, владеющим основами криптографии.

Необходимо знание языков программирования.

Web: https://soft-logic.ru/
Mail: info@soft-logic.ru

3 ВВЕДЕНИЕ

Документ предназначен для систем мерчантов и их разработчиков:

1. Для проведения доработки (разработки) программного обеспечения для систем мерчантов, разработки технических задании на доработку (разработку) соответствующих систем , обеспечивающих возможность подключения к системам мерчантов.

2. Для оценки качества проведенных доработок и соответствия доработанной системы общим техническим требованиям во время комплексных испытаний. Реализация требовании данного документа позволит подключить платежный виджет к системе приема платежей и обеспечить списания денежных средств.

Web: https://soft-logic.ru/
Mail: info@soft-logic.ru

4 ОБЩИЕ ПОЛОЖЕНИЯ

Платежный виджет представляет собой форму для приема платежей онлайн с использованием различных вариантов оплат (например, система быстрых платежей, банковская карта и т.д.). Виджет интегрируется в систему партнера и позволяет принимать оплаты заказов от клиента непосредственно на сайте партнера. Такой способ приема платежей обеспечивает удобство оплаты для клиента и безопасность платежных данных клиента.

Особенности использования платежного виджета:

- Безопасность платежных данных клиента со стороны программного обеспечения обеспечивается согласно стандарту безопасности PCI DSS;
- Платежный виджет интегрируется в систему партнера за короткое время, что позволяет быстро начать прием оплат;
- Единая точка позволяет принимать оплаты по всем источникам оплаты, настроенным в системе. Единая интеграция позволяет клиенту использовать любой удобный ему источник оплаты (рисунок 4.1);
- Для платежного виджета может быть применена кастомизация под визуальный стиль системы партнера.

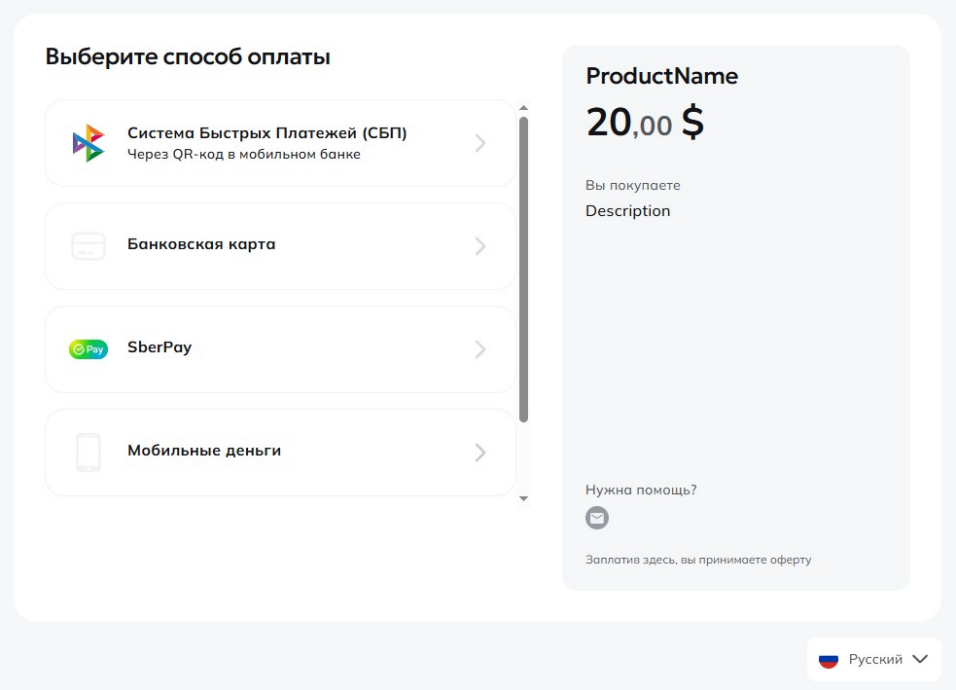

Рисунок 4.1 — Пример страницы выбора источника оплаты

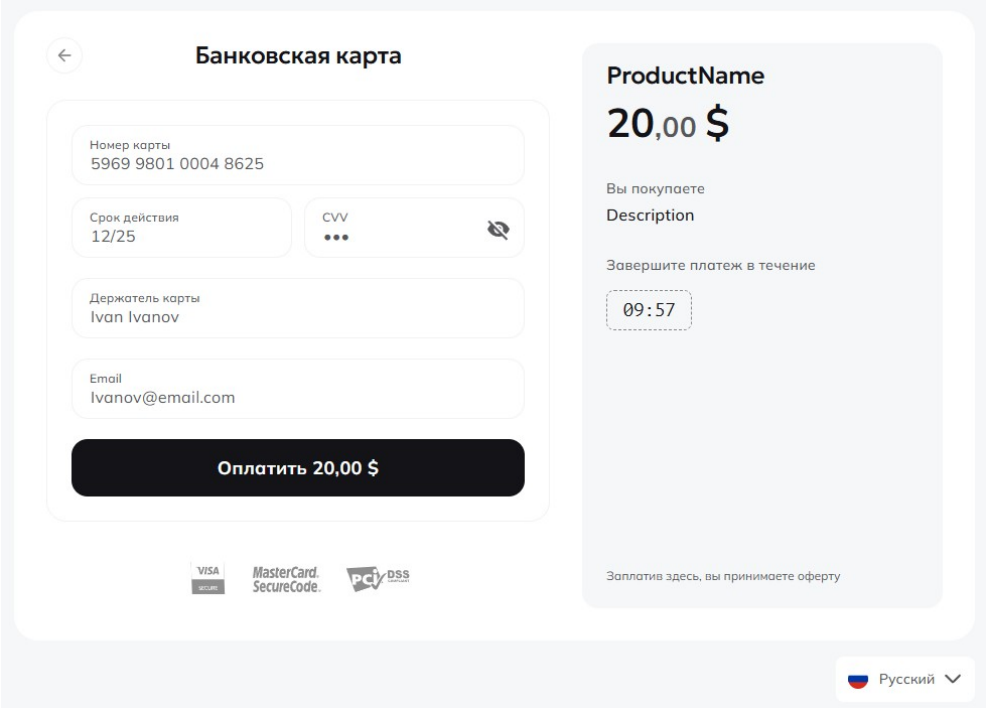

Рисунок 4.2 — Пример страницы ввода данных банковской карты

Схема создания заказа и списания средств с использованием платежного виджета приведена на рисунке 4.3.

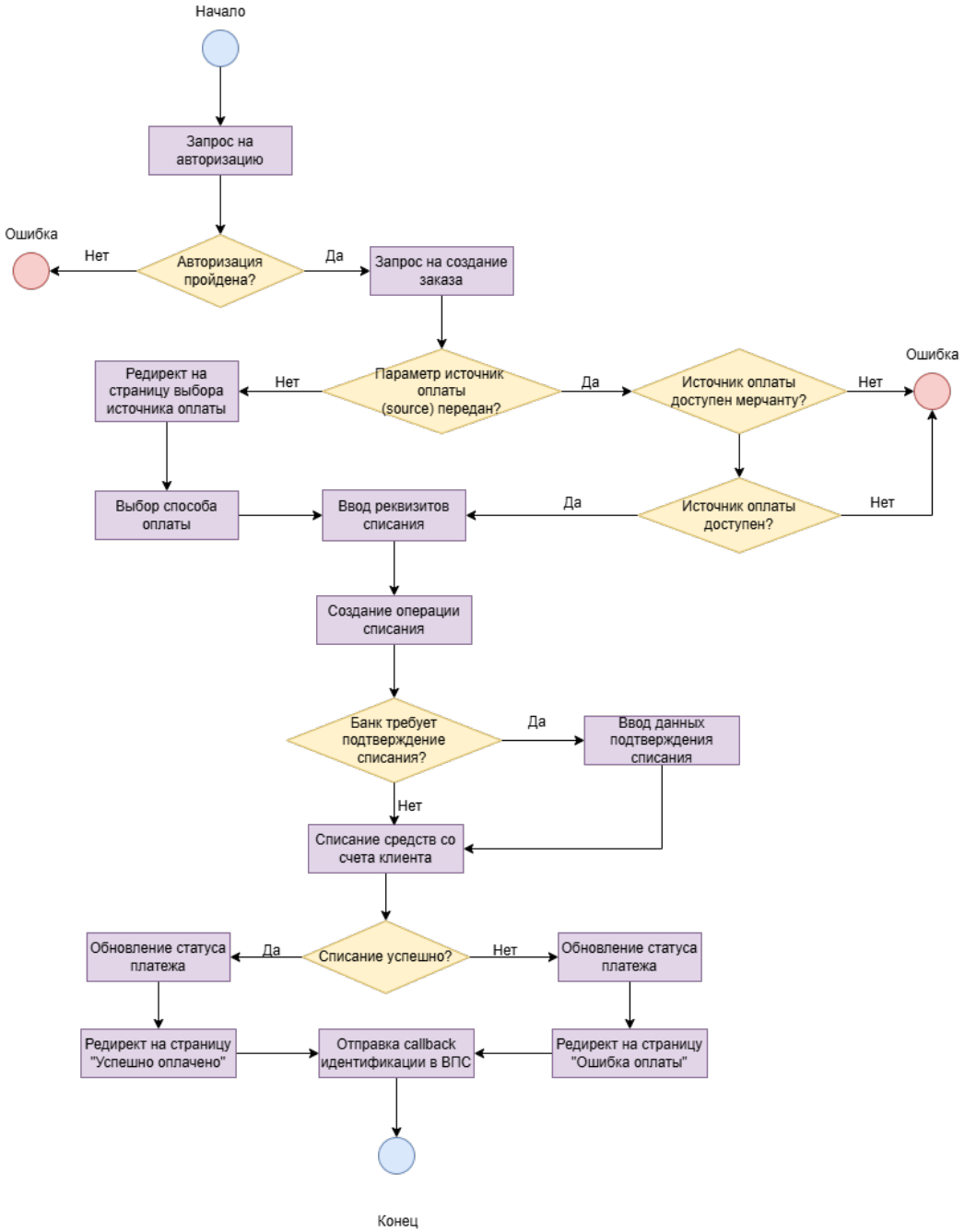

Рисунок 4.3 — Схема проведения оплаты заказа через платежный виджет

5 ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

5.1 СЕТЕВОЕ ВЗАИМОДЕЙСТВИЕ

Сетевое взаимодействие системы мерчанта и ПЦ должно происходить в рамках протокола HTTPS. Для взаимодействия с системой мерчанта ПЦ предоставляет адрес, на который будут отправляться запросы. Запросы отправляются методом POST.

При ответе на полученный пакет ПЦ возвращает пакет с подписью, в котором содержатся результаты обработки данных пакета. Данные должны помещаться в теле страницы, генерируемой в ответ на обращение со стороны системы мерчанта.

Сервер ПЦ имеет возможность обрабатывать параллельно несколько запросов.

Авторизация на сервере ПЦ производится с использованием электронной подписи.

5.2 АВТОРИЗАЦИЯ

Для авторизации запросов в системе используется проверка электронной подписи, для этого система ожидает от мерчанта передачу специальных HTTP-заголовков:

- **PSP-Point** — идентификатор API мерчанта, присвоенный в PSP;
- **PSP-Sign** — подпись запроса.

Ключи для формирования подписи генерируются на сервере ПЦ при создании API с типом «Платежный виджет» (создание API и ключей для формирования подписи описано в разделе 6).

Электронная подпись формируется от строки <REQUEST METHOD>+<REQUEST FULL URI>+<Request Body> по алгоритму SHA256+RSA в формате Base64 (UTF8).

Пример подписываемой строки:

```
POST/psp/payment-widget/
register{"client":"2437583234","currency":"RUB","date":"2024-11-
07T08:29:16.784Z","description":"Война и
Мир","id":"20241112030","params":[{"code":"client-
agentName","value":"cypix.ru"}],"product":"Покупка
книги","redirectUrlError":"https://processing.pay2.pro/result/error","red
irectUrlSuccess":"https://processing.pay2.pro/result/
success","service":8,"sum":100}
```

Получившаяся подпись будет иметь вид:

```
mkF2ZzPFU0tdnASr2ZDtCXCrPh3IvPo2vhBl33fkvEX7vNSkVvkT2G+sqf2jMAUTQ7c/T2s/
ppuz/7aLXZWAQbAPYIatgRXIwXSJW8y8SO99h4jQrenVly1abPLSgC3SK0scmhBwkMGA/iI/
i/5S+q+tIfDcnbCRltI1Fu8cf1A=
```

Пример генерации и проверки подписи приведен в приложении А. Примеры генерации и проверки ЭП на java.

Web: https://soft-logic.ru/
Mail: info@soft-logic.ru

6 СОЗДАНИЕ API ТИПА ПЛАТЕЖНЫЙ ВИДЖЕТ

Для запуска платежного виджета необходимо создать в личном кабинете мерчанта API с типом «Платежный виджет» (раздел «API — Список API»).

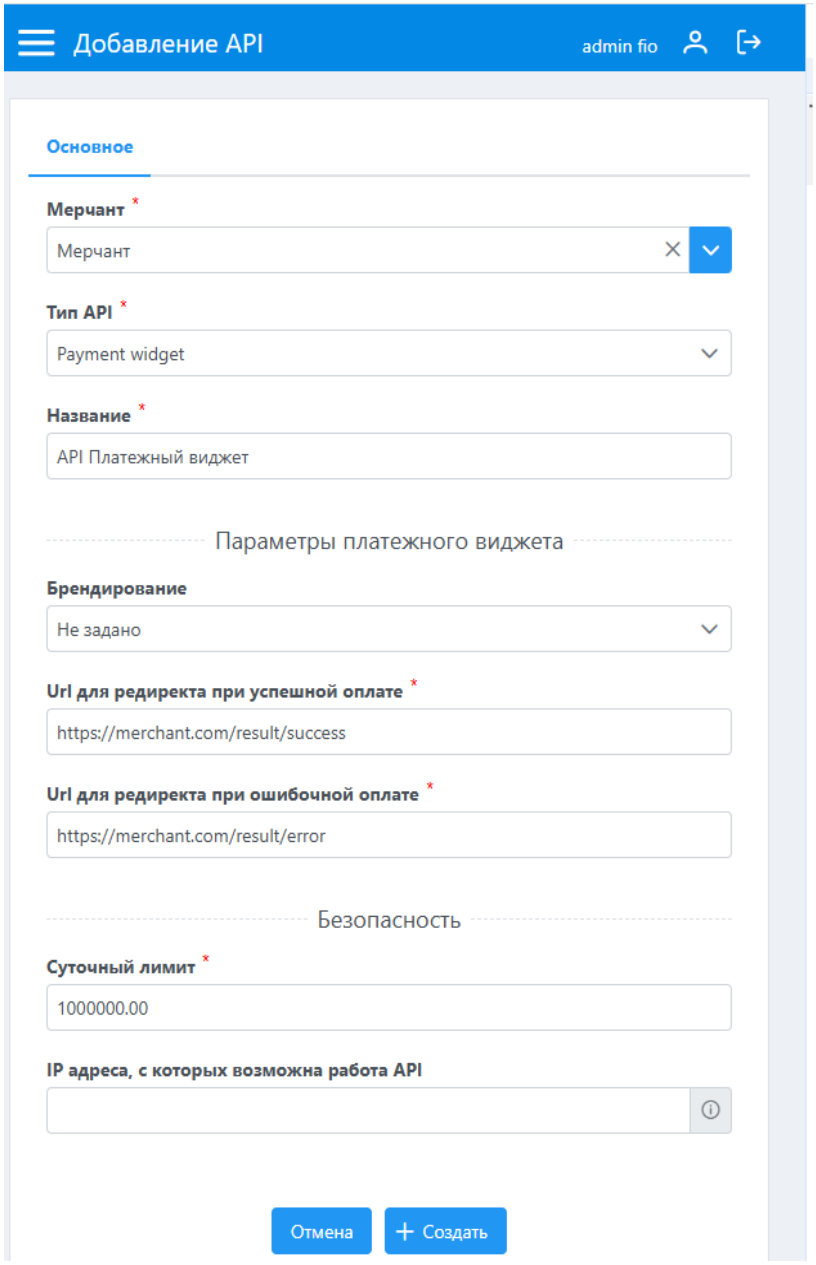

Рисунок 6.1 — Создание точки с типом "Платежный виджет"

1. Для создания API заполните следующие поля:

Блок **Основное**:

- **Мерчант** — мерчант, которому принадлежит API;
- **Тип API** — Payment widget;
- **Название** — будет использоваться в отчетах и выпадающих списках в личном кабинете;

Блок **Платежный виджет**:

- **Стиль платежного виджета** — стиль кастомизации, который будет применен к платежному виджету.
- **Url для редиректа при успешной оплате** — URL-адрес, на который будет перенаправлен клиент при успешном проведении платежа;
- **Url для редиректа при ошибочной оплате** — URL-адрес, на который будет перенаправлен клиент в случае возникновении ошибки при проведении платежа;

Внимание!
В дальнейшем параметры **«Url для редиректа»** могут быть переопределены в запросе на создание заказа.

Блок **Безопасность**:

- **Суточный лимит** — максимальная сумма оплат в сутки через платежный виджет;
- **IP-адреса, с которых возможна работа API** — IP-адреса сайтов, на которых будет разрешено размещение платежного виджета. При указании, работа с других IP-адресов будет невозможна, IP указываются через запятую.

2. При успешном создании API будет сформировано уведомление о необходимости генерации ключей. Ключи можно сгенерировать сразу, либо позже на вкладке «Безопасность». Набор ключей, открытая и закрытая часть, будет доступен для скачивания на этой же вкладке (рисунок 6.2).

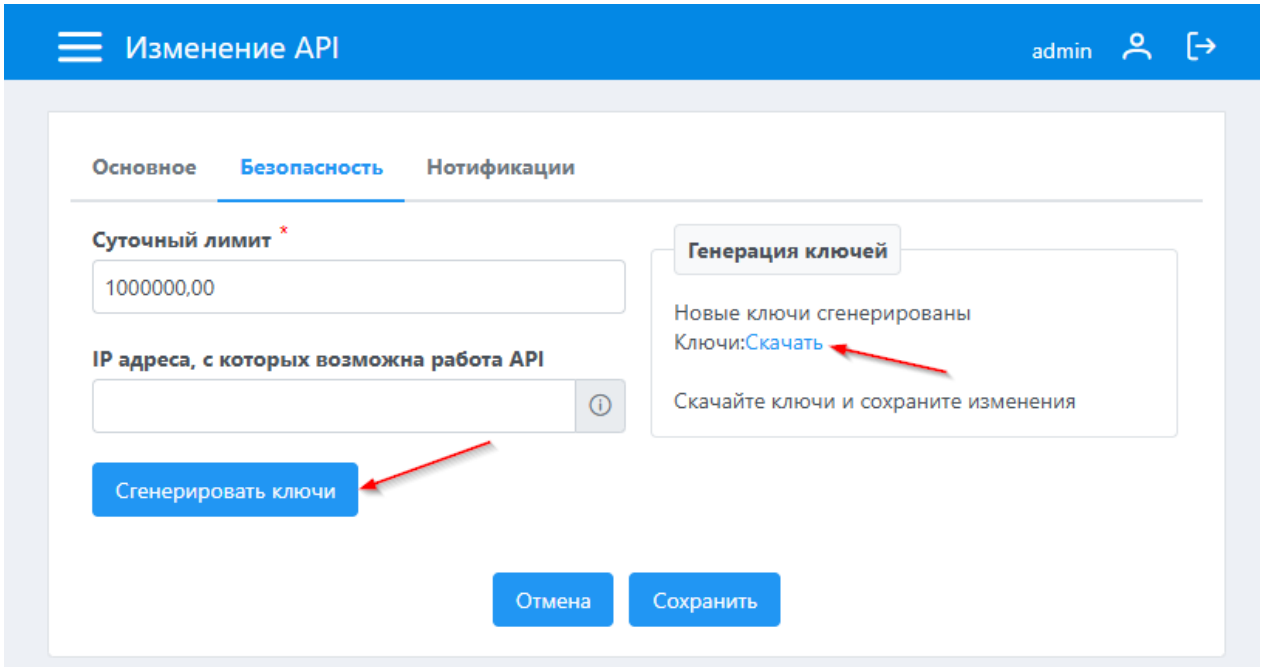

Рисунок 6.2 — Вкладка «Безопасность»

7 ИНТЕГРАЦИЯ В СИСТЕМУ ПАРТНЕРА

7.1 ЗАКАЗ НА ОПЛАТУ

7.1.1 ОБЩАЯ ИНФОРМАЦИЯ

Схема взаимодействия участников приведена на рисунке 7.1.1.

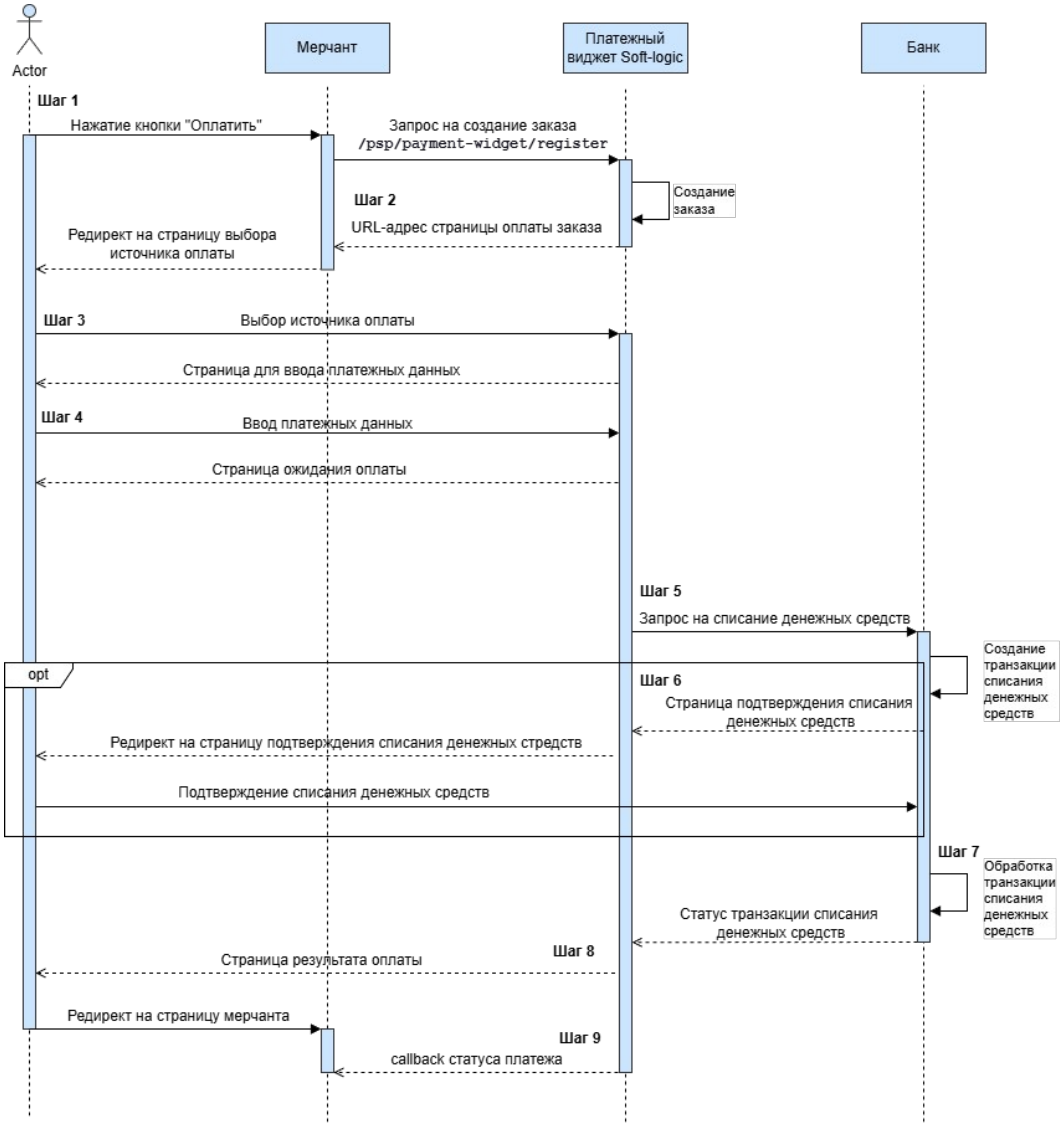

Рисунок 7.1.1 — Диаграмма взаимодействия участников процесса оплаты через платежный виджет

Web: https://soft-logic.ru/
Mail: info@soft-logic.ru

Шаг 1: Для создания операции списания денежных средств мерчант вызывает POST-запрос создания заказа **`/psp/payment-widget/register`**. При вызове данного метода заказ будет зарегистрирован в системе.

Шаг 2: Система возвращает мерчанту URL-адрес страницы оплаты заказа. На которую мерчант должен перенаправить клиента. На странице клиенту будет доступен выбор способа оплаты из списка разрешенных, согласно профилю API.

Шаг 3: Клиентом производится выбор предпочтительного для него способа оплаты. Платежный виджет возвращает клиенту страницу ввода платежных данных клиенту.

Шаг 4: Клиент вводит реквизиты списания денежных средств, в ответ система возвращает страницу подтверждения платежа.

Шаг 5: Платежный виджет отправляет запрос на списание денежных средств банку. Банк создает транзакцию списания денежных средств.

Шаг 6: Опционально банк может запросить подтверждение списания денежных средств и отправит платежному виджету страницу для подтверждения. Платежный виджет произведет редирект клиента на страницу для подтверждения списания денежных средств. Клиент производит подтверждение списания денежных средств, способом в зависимости от выбранного источника оплаты.

Шаг 7: Банк производит обработку транзакции списания денежных средств и возвращает ее результат платежному виджету.

Шаг 8: В соответствии с полученным результатом списания денежных средств от банка платежный виджет отправляет клиенту страницу результата оплаты (Успех или Ошибка). Затем клиент будет возвращен на страницу мерчанта.

Шаг 9: Система платежного виджета отправляет callback-нотификацию статуса платежа в систему мерчанта.

Дополнительные возможности платежного виджета:

- При необходимости мерчантом может быть дополнительно запрошен статус заказа повторным вызовом метода создания заказа **`/psp/payment-widget/register`**. В системе предусмотрена защита от дубликатов заказов, проверка на дубли производится по уникальному идентификатору операции мерчанта. При повторном вызове метода создания заказа по уже зарегистрированному в системе заказу с одинаковым значением параметра id, метод вернет в ответ результате обработки заказа и его текущий статус. Рекомендуется запросить статус если автоматически не был получен ответ в течении 5 минут.

- Доступна возможность использовать необходимый источник оплаты, минуя этап выбора клиентом источника оплаты. Для этого при вызове метода **`/psp/payment-widget/register`** необходимо задать параметр **`sourse`**. Доступные коды источников оплаты требуется уточнить у менеджера проекта.

7.1.2 СОЗДАНИЕ ЗАКАЗА НА ОПЛАТУ

POST метод **`/psp/payment-widget/register`** предназначен для регистрации заказа на оплату в системе.

Структура запроса:

```
{
  "client": "string",
  "currency": "string",
  "date": "2025-02-07T08:05:47.191Z",
  "description": "string",
  "id": "string",
  "params": [
    {
      "code": "string",
      "value": "string"
    }
  ],
  "product": "string",
  "redirectUrlError": "string",
  "redirectUrlSuccess": "string",
  "service": 0,
  "source": "string"
  "sum": 0
}
```

Таблица 7.1.2.1 — Атрибуты запроса на создание заказа на оплату

Название	Описание	Обязательность
`id`	Уникальный идентификатор операции мерчанта	Да
`currency`	Код валюты транзакции в формате ISO 4217	Да

sum	Стоимость заказа в минорной единице валюты	Да
date	Дата создания заказа на стороне мерчанта	Да
client	Идентификатор клиента на стороне мерчанта	Да
service	Идентификатор сервиса на стороне системы. Предоставляется мерчанту при интеграции	Да
description	Описание заказа, которое передано мерчантом	Нет
product	Наименование товара или услуги	Да
redirectUrlSuccess	URL мерчанта для редиректа при успешной оплате	Нет
redirectUrlError	URL мерчанта для редиректа при ошибочной оплате	Нет
params[]	Массив дополнительных параметров заказа. Требования к заполнению дополнительных параметров предоставляется мерчанту при интеграции	Нет
params[].code	Код параметра заказа	Нет

params[].value	Значение параметра заказа	Нет
source	Код источника оплаты. Если параметр передан в запросе, то для клиента сразу будет открыт экран ввода данных для указанного в **source** источника оплаты, минуя страницу выбора источника оплаты. Допустимые значения: • SBP • SBERPAY • MOBILE_MONEY • BANKCARD-PS • HALYK_QR • ELQR	Нет

Структура ответа:

```
{
  "date": "string",
  "error": 0,
  "errorDetail": 0,
  "errorMessage": "string",
  "id": "string",
  "orderId": 0,
  "redirectUrl": "string",
  "state": 0
}
```

Таблица 7.1.2.2 — Атрибуты ответа на запрос создания заказа на оплату

Параметр	Описание	Обязательность
id	Уникальный идентификатор операции мерчанта	Да если error=0, иначе Нет

orderId	Идентификатор заказа на стороне ПЦ	Да если error=0, иначе Нет
state	Код статуса состояния заказа, может принимать значения, описанные в таблице 7.1.2.3	Да если error=0, иначе Нет
stateDescription	Текстовое описание статуса состояния заказа, может принимать значения, описанные в таблице 7.1.2.3	Да если error=0, иначе Нет
date	Дата создания заказа на стороне платежной системы	Да если error=0, иначе Нет
redirectUrl	URL для редиректа на платежную страницу	Да если error=0, иначе Нет
error	Код статуса обработки запроса, может принимать значения, описанные в таблице 7.1.2.4	Да
errorMessage	Описание ошибки	Нет

Таблица 7.1.2.3 — Значения параметров **state** и **stateDescription**

Значение параметра **state**	Значение параметра **stateDescription**	Описание	Финальность
0	NEW	Заказ создан, данные для оплаты не введены	Нет
40	PROCESS	Заказ обрабатывается на стороне банка	Нет
60	SUCCESS	Успех	Да
80	ERROR	Ошибка	Да
81	ERROR_EXPIRED	Ошибка по таймауту (если по	Да

		заказу не был создан платеж)	

Таблица 7.1.2.4 — Значения параметра **error**

Код	Описание
0	Ошибок нет, запрос обработан успешно. Необходимо анализировать значение параметра **state.**
!=0	Ошибка выполнения запроса, требуется обратиться к сотруднику системы для локализации причины.

Примеры запроса и ответа приведены в приложении В. Примеры запросов и ответов для метода создания заказа на оплату.

7.1.3 ПРЕДЗАПОЛНЕНИЕ ДАННЫХ В ЗАКАЗЕ

При использовании источника оплаты **MOBILE_MONEY** доступно использование функционала предзаполнения данных клиента на форме платежного виджета. Для этого мерчанту необходимо передать в запросе на создание заказа в массиве дополнительных параметров заказа **params[]** данные о ФИО клиента, номере телефона и e-mail адресе (см. таблицу 7.1.3.1). Страна определяется автоматически по коду номера телефона.

Данные, переданные в запросе, недоступны для редактирования клиентом. Если какой-то из параметров не передан при создании заказа, то поле останется редактируемым для клиента.

Таблица 7.1.3.1 — Дополнительные параметры заказа для предзаполнения данных клиента

Код параметра	Описание	Обяз.
customerName	Имя клиента	Нет
customerEmail	Электронная почта клиента	Нет
phoneNumber	Номер телефона клиента	Нет

Пример создания заказа с предзаполненными данными приведен в приложении С.Пример запроса для метода создания заказа на оплату с предзаполненными данными.

7.1.4 ОПЛАТА ЗАКАЗА ЧЕРЕЗ SBERPAY

Если в системе для оплаты заказа используется источник оплаты **SBERPAY** в секции **params** дополнительно указываются параметры, описанные в таблице 7.1.4.1.

Таблица 7.1.4.1 — Параметры, передаваемые в секции **params**, при использовании источника оплаты SBERPAY

Код параметра	Описание	Обяз.
deepLink	Ссылка для возврата в приложение мерчанта	Да

Пример создания заказа приведен в приложении В. Примеры запросов и ответов для метода создания заказа на оплату.

7.2 ЗАКАЗ НА ВЫПЛАТУ

7.2.1 ОБЩАЯ ИНФОРМАЦИЯ

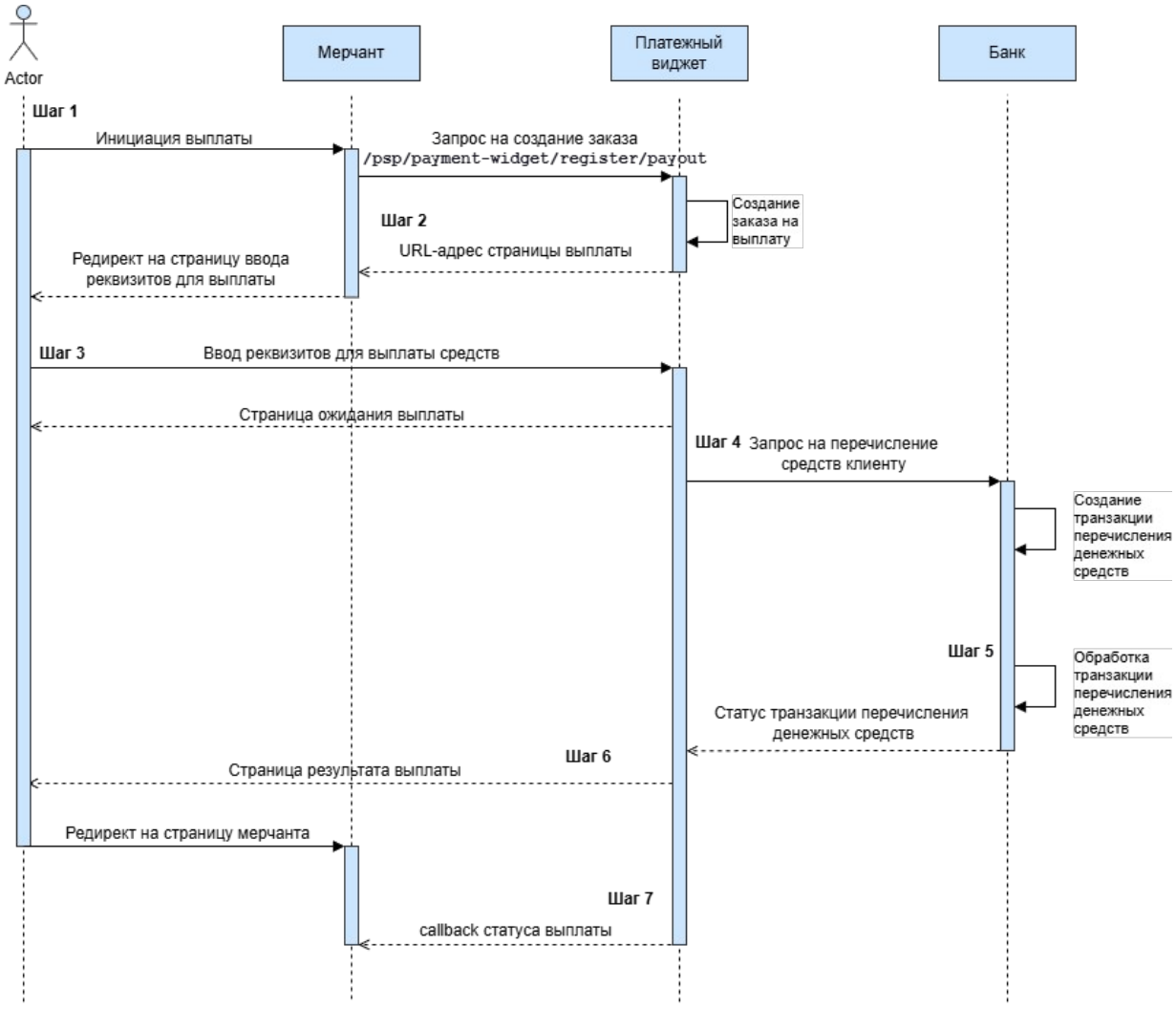

Рисунок 7.2.1 — Диаграмма взаимодействия участников процесса выплаты через платежный виджет

Шаг 1: Для создания операции выплаты денежных средств клиенту мерчант вызывает POST-запрос создания заказа **`/psp/payment-widget/register/payout`**.

При вызове данного метода заказ на выплату будет зарегистрирован в системе.

Шаг 2: Система возвращает мерчанту URL-адрес страницы заказа на выплату, на которую мерчант должен перенаправить клиента.

Шаг 3: Клиент вводит реквизиты для выплаты денежных средств.

Шаг 4: Платежный виджет отправляет запрос банку на перечисление денежных средств. Банк создает транзакцию перечисления денежных средств клиенту.

Шаг 5: Банк производит обработку транзакции перечисления денежных средств и возвращает ее результат платежному виджету.

Шаг 6: В соответствии с полученным результатом перечисления денежных средств от банка платежный виджет отправляет клиенту страницу результата выплаты (Успех или Ошибка). Затем клиент будет возвращен на страницу мерчанта.

Шаг 7: Система платежного виджета отправляет callback-нотификацию статуса выплаты в систему мерчанта.

Дополнительные возможности платежного виджета:

- При необходимости мерчантом может быть дополнительно запрошен статус операции выплаты повторным вызовом метода **`/psp/payment-widget/register/payout`**. В системе предусмотрена защита от дубликатов, проверка на дубли производится по уникальному идентификатору операции мерчанта. При повторном вызове метода создания заказа на выплату по уже зарегистрированному в системе заказу с одинаковым значением параметра id, метод вернет в ответ результат обработки заказа и его текущий статус. Рекомендуется запросить статус если автоматически не был получен ответ в течении 5 минут.

7.2.2 СОЗДАНИЕ ЗАКАЗА НА ВЫПЛАТУ

POST метод **/psp/payment-widget/register/payout** предназначен для регистрации в системе заказа на выплату средств клиенту.

Структура запроса:

```
{
  "client": "string",
  "currency": "string",
  "date": "2025-02-07T08:05:47.191Z",
  "description": "string",
  "id": "string",
  "params": [
    {
      "code": "string",
      "value": "string"
    }
  ],
  "product": "string",
  "redirectUrlError": "string",
  "redirectUrlSuccess": "string",
  "service": 0,
  "sum": 0
}
```

Таблица 7.2.2.1 — Атрибуты запроса на создание заказа на выплату

Название	Описание	Обязательность
id	Уникальный идентификатор операции мерчанта	Да
currency	Код валюты транзакции в формате ISO 4217	Да
sum	Сумма выплаты в минорной единице	Да

	валюты	
date	Дата создания заказа на стороне мерчанта	Да
client	Идентификатор клиента на стороне мерчанта	Да
service	Идентификатор сервиса на стороне системы. Предоставляется мерчанту при интеграции	Да
description	Описание назначения выплаты	Нет
product	Финансовый продукт, по которому производится выплата	Да
redirectUrlSuccess	URL мерчанта для редиректа при успешной выплате	Нет
redirectUrlError	URL мерчанта для редиректа при ошибочной выплате	Нет
params[]	Массив дополнительных параметров заказа. Требования к заполнению дополнительных параметров предоставляется мерчанту при интеграции	Нет
params[].code	Код параметра заказа	Нет
params[].value	Значение параметра заказа	Нет

Структура ответа:

```
{
  "date": "string",
  "error": 0,
  "errorDetail": 0,
  "errorMessage": "string",
  "id": "string",
  "orderId": 0,
  "redirectUrl": "string",
  "state": 0
}
```

Таблица 7.2.2.2 — Атрибуты ответа на запрос создания заказа на оплату

Параметр	Описание	Обязательность
id	Уникальный идентификатор операции мерчанта	Да если error=0, иначе Нет
orderId	Идентификатор заказа на стороне ПЦ	Да если error=0, иначе Нет
state	Код статуса состояния заказа, может принимать значения, описанные в таблице 7.2.2.3	Да если error=0, иначе Нет
stateDescription	Текстовое описание статуса состояния заказа, может принимать значения, описанные в таблице 7.2.2.3	Да если error=0, иначе Нет
date	Дата создания заказа на стороне платежной системы	Да если error=0, иначе Нет
redirectUrl	URL для редиректа на платежную страницу	Да если error=0, иначе Нет
error	Код статуса обработки запроса, может принимать значения, описанные в таблице	Да

	7.2.2.4	
errorMessage	Описание ошибки	Нет

Таблица 7.2.2.3 — Значения параметров **state** и **stateDescription**

Значение параметра **state**	Значение параметра **stateDescription**	Описание	Финальность
0	NEW	Заказ создан, данные для оплаты не введены	Нет
40	PROCESS	Заказ обрабатывается на стороне банка	Нет
60	SUCCESS	Успех	Да
80	ERROR	Ошибка	Да
81	ERROR_EXPIRED	Ошибка по таймауту (если по заказу не был создан платеж)	Да

Таблица 7.2.2.4 — Значения параметра **error**

Код	Описание
0	Ошибок нет, запрос обработан успешно. Необходимо анализировать значение параметра **state.**
!=0	Ошибка выполнения запроса, требуется обратиться к сотруднику системы для локализации причины.

Примеры запроса и ответа приведены в приложении D. Примеры запроса и ответа для метода создания заказа на выплату.

7.3 БАЛАНС МЕРЧАНТА

Для запроса баланса необходимо реализовать поддержку метода **`/psp/external/api/v2/balance`**.

В ответе на запрос будет возвращена информация о счетах, включая: валюту, общий баланс, а также детализацию суммы общего баланса.

Структура запроса:

```
{
  "accounts": [
    {
      "awaitingEnrollment": 0,
      "balance": 0,
      "available": 0,
      "currency": "string",
      "rollingReserve": 0
    }
  ]
}
```

Таблица 7.3.1 — Параметры ответа на запрос баланса мерчанта

Параметр	Описание
`currency`	Валюта счета
`balance`	Общий баланс
`rollingReserve`	Удерживаемый резерв *(роллинг резерв)*
`awaitingEnrollment`	Блокированный баланс по операциям

`available`	Доступный для взаиморасчёта баланс *(Рассчитывается по формуле: Общий баланс – Блокированный баланс по операциям – Удерживаемый резерв – Взаиморасчеты в обработке)*

Пример ответа на запрос приведен в приложении Е. Пример ответа на запрос баланса мерчанта.

7.4 УВЕДОМЛЕНИЕ ПАРТНЕРА О СТАТУСЕ ОПЕРАЦИИ

В системе платежного виджета предусмотрен функционал callback-нотификации мерчанта о переходе операции в финальный статус (успех или ошибка). Функционал реализован с использованием технологии отправки webhook-оповещений.

Для настройки функционала, необходимо указать URL-адрес, на который будут отправляться webhook-оповещения. В настройках API на вкладке Нотификации (рисунок 7.4.1) необходимо заполнить параметр **«URL-адрес для отправки уведомлений о статусе платежа»**.

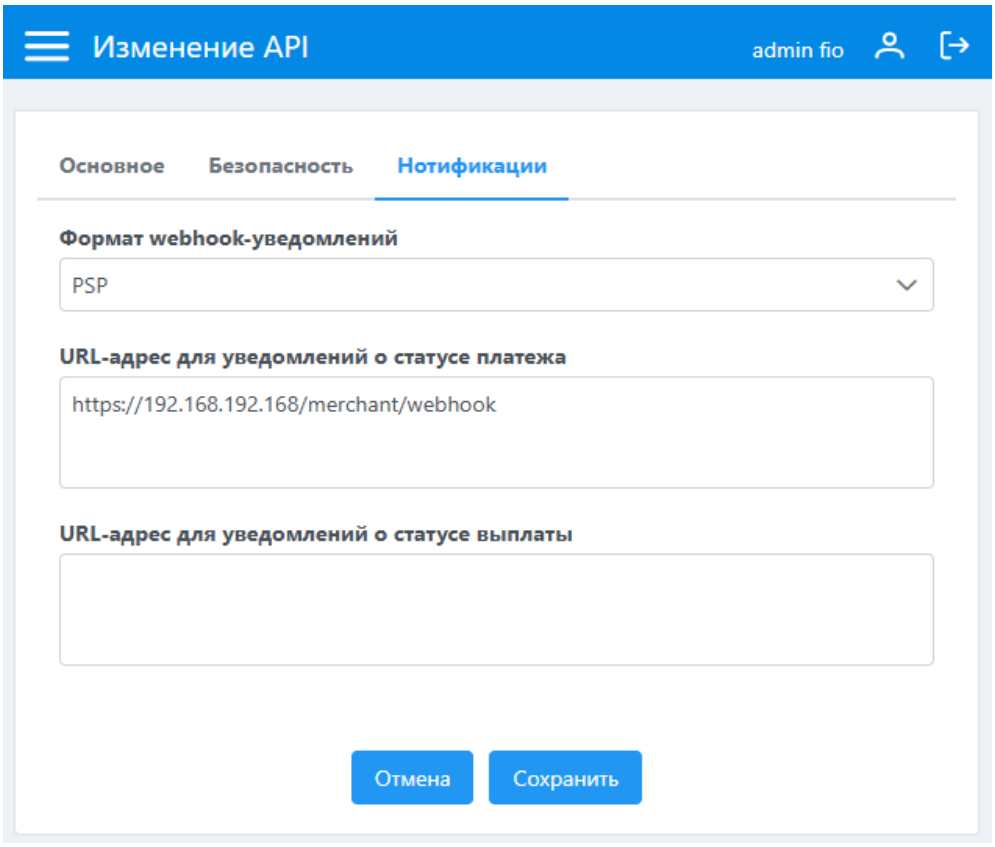

Рисунок 7.4.1 — Заполнение URL-адреса для отправки webhook-оповещений в настройках API

Структура webhook-оповещения:

```
{
  "id" : string,
  "transaction" : 21799382,
  "status" : string,
  "errorDetail" : null,
  "error" : 0,
  "message" : null,
  "bankPayment" : null,
  "date" : string,
  "attribute" : [ {
    "code" : string,
    "value" : string,
    "title" : null,
    "valueTitle" : null,
    "flags" : null
  } ],
  "menu" : {
    "service" : 8,
    "serviceName" : string,
    "category" :  string
  },
  "sumOutcome" : 10000
}
```

Таблица 7.4.1 — Атрибуты webhook-оповещения

Параметр	Описание	Обязательность
id	Уникальный идентификатор транзакции мерчанта	Да
transaction	Уникальный идентификатор транзакции системы	Да
status	Статус операции, может принимать значения: • SUCCESS — Успех • ERROR — Ошибка	Да

sumOutcome	Сумма операции, в копейках	Да
error	Код ошибки	Нет
errorDetail	Детализирующий код ошибки	Нет
message	Описание	Нет
bankPayment	Объект данных о банковской транзакции. Описание объекта приведено в таблице 7.4.2	Нет
date	Дата транзакции	Нет
attribute	Объект данных атрибутов основной операции. Описание объекта приведено в таблице 7.4.3	Нет
menu	Объект данных сервиса. Описание объекта приведено в таблице 7.4.4	Нет

Таблица 7.4.2— Атрибуты объекта **bankPayment**

Параметр	Описание	Обязательность
status	Статус операции, может принимать значения: • SUCCESS — Успех • ERROR — Ошибка	Нет
transaction	Уникальный идентификатор транзакции на стороне банка	Нет
attributes	Атрибуты банковской операции	Нет

Таблица 7.4.3— Атрибуты объекта **attribute**

Параметр	Описание	Обязательность
code	Код атрибута	Нет
value	Значение	Нет
title	Заголовок	Нет
valueTitle	Отображаемый заголовок	Нет
flags	Флаг	Нет

Таблица 7.4.4— Атрибуты объекта **menu**

Параметр	Описание	Обязательность
cervice	Код сервиса	Нет
cerviceName	Название сервиса	Нет
category	Категория сервиса	Нет

В ответ от мерчанта системой будет ожидаться HTTP-ответ:

- **200** — webhook-оповещение успешно принято, система прекращает отправку;
- **401, 403** — доступ запрещен, система прекращает отправку;
- **404** — URL, система прекращает отправку;

- **любой код, отличный от 200, 401, 403 и 404** — ошибка, система повторит отправку позже. Webhook-оповещение будет находиться в очереди на отправку 24 часа и будет периодически отправляться повторно. Если по истечению 24 часов, от мерчанта не будет получен успешный ответ, то система прекратит отправку.

Пример оповещения приведен в приложении F. Пример webhook-оповещения.

Для авторизации запросов webhook-оповещений используется проверка электронной подписи. Для этого в запросе передается в HTTP-заголовок **X-Sign,** содержащий в себе идентификатор API мерчанта, присвоенный в системе PSP.

Электронная подпись формируется от строки <REQUEST METHOD>+<REQUEST FULL URI>+< REQUEST BODY> BODY> по алгоритму SHA256+RSA в формате Base64 (UTF8).

Пример:

Webhook-оповещение отправляется мерчанту по адресу — **`https://192.168.192.168/psp/webhook`**

Подпись в данном примере будет иметь вид:

```
signBase64(POST/psp/webhook{"body":"body"})
```

Проверка подписи осуществляется по публичной части ключа, полученной от системы платежного виджета.

Пример генерации и проверки подписи приведен в приложении A. Примеры генерации и проверки ЭП на java.

8 ПРИЛОЖЕНИЕ

A. ПРИМЕРЫ ГЕНЕРАЦИИ И ПРОВЕРКИ ЭП НА JAVA

Пример генерации и проверки подписи API:

```
import org.apache.commons.codec.binary.Base64;
import org.apache.commons.codec.binary.Hex;
import org.bouncycastle.openssl.PEMReader;
import org.bouncycastle.openssl.PasswordFinder;
import java.io.FileReader;
import java.io.IOException;
import java.security.*;

public class Signer {
    public static final String defaultAlgoritm = "SHA256withRSA";
    public static final String defaultCharset = "UTF-8";

    private PrivateKey privateKey;
    private PublicKey publicKey;
    private String algoritm;
    private String charset;

    public Signer(String publicKeyPath, String privateKeyPath) throws IOException
{
        try (FileReader fr = new FileReader(publicKeyPath);
             PEMReader pr = new PEMReader(fr)) {
            publicKey = (PublicKey) pr.readObject();
        }

        try (FileReader fr = new FileReader(privateKeyPath);
             PEMReader pr = new PEMReader(fr)) {

            KeyPair kp = (KeyPair) pr.readObject();
            privateKey = kp.getPrivate();
        }
    }
    public boolean verifyBase64(String message, String signature) throws
SignatureException {
        try {
            Signature sign = Signature.getInstance(defaultAlgoritm);
            sign.initVerify(publicKey);
            sign.update(message.getBytes(defaultCharset));
            return sign.verify(Base64.decodeBase64(signature.getBytes(charset)));
```

```
        } catch (Exception ex) {
        throw new SignatureException(ex);
        }
    }

    public String signBase64(String message) throws SignatureException {
        try {
            Signature sign = Signature.getInstance(defaultAlgoritm);
            sign.initSign(privateKey);
            sign.update(message.getBytes(defaultCharset));
            return new String(Base64.encodeBase64(sign.sign()), charset);
        } catch (Exception ex) {
            throw new SignatureException(ex);
        }
    }
}
```

Пример подписи генерации и проверки подписи при отправке webhook-оповещения:

```
import org.apache.commons.codec.binary.Base64;
import org.apache.commons.codec.binary.Hex;
import org.bouncycastle.openssl.PEMReader;
import org.bouncycastle.openssl.PasswordFinder;

import java.io.FileReader;
import java.io.IOException;
import java.security.*;

public class Signer {

    public static final String defaultAlgoritm = "SHA256withRSA";
    public static final String defaultCharset = "UTF-8";

    private PrivateKey privateKey;
    private PublicKey publicKey;
    private String algoritm;
    private String charset;

    public Signer(String publicKeyPath) throws IOException {
        try (FileReader fr = new FileReader(publicKeyPath);
             PEMReader pr = new PEMReader(fr)) {
            publicKey = (PublicKey) pr.readObject();
        }
    }
```

```
    public boolean verifyBase64(String message, String signature) throws
SignatureException {
        try {
            Signature sign = Signature.getInstance(defaultAlgoritm);
            sign.initVerify(publicKey);
            sign.update(message.getBytes(defaultCharset));
            return sign.verify(Base64.decodeBase64(signature.getBytes(charset)));
        } catch (Exception ex) {
            throw new SignatureException(ex);
        }
    }
}
```

В. ПРИМЕРЫ ЗАПРОСОВ И ОТВЕТОВ ДЛЯ МЕТОДА СОЗДАНИЯ ЗАКАЗА НА ОПЛАТУ

Пример запроса на платеж:

```
{
  "client": "2437583234",
  "currency": "RUB",
  "date": "2024-11-07T08:29:16.784Z",
  "description": "Собрание сочинений А.С. Пушкина",
  "id": "20241112030",
  "params": [
    {
      "code": "client-agentName",
      "value": "merchant.ru"
    }
  ],
  "product": "Покупка книги",
  "redirectUrlError": "https://merchant.ru/result/error",
  "redirectUrlSuccess": "https://merchant.ru/result/success",
  "service": 8,
  "sum": 157950
}
```

Пример успешного ответа на запрос:

```
{
  "error": 0,
  "id": "20241112030",
  "state": 0,
  "orderId": 56,
  "redirectUrl":
"https://app.merchant.ru/psp/payment-widget/order/93ef4f589dfbf5963fc94f422a6f282
```

```
a12579a015a6e974e9669748ed15a36441c6116b82adcfbc337f5efc68e8d54f2fe1191bcb4ae23ce
d93be69c514d9c87a932d6a03005000a0e27b6bcb300020d5bad549bb763dd50f4ba386br98741913
cecbd488u265437851142fc26175b5d319787f5778e917cd0de9a7272e2a4ea",
  "date": "2024-11-12T10:07:03.368+0000"
}
```


Пример ошибочного ответа на запрос:


```
{
  "error": 1,
  "errorMessage": "Ошибочное значение параметра: должно быть задано [Ошибочное
поле: 'id' : 'null']; "
}
```


Пример запроса на платеж с использованием источника оплаты SBERPAY:


```
{
    "client": "79600000000",
    "currency": "RUB",
    "date": "2025-08-20T08:29:16.784Z",
    "description": "Война и Мир",
    "id": "20251028011",
    "params": [{
        "code": "deepLink",
        "value": "https://merchant.ru/deepLink/"
    }],
    "product": "Покупка книги",
    "redirectUrlError": "https://merchant.ru/error/",
    "redirectUrlSuccess": "https://merchant.ru/success/",
    "service": 8,
    "sum": 100
}
```


C. ПРИМЕР ЗАПРОСА ДЛЯ МЕТОДА СОЗДАНИЯ ЗАКАЗА НА ОПЛАТУ С ПРЕДЗАПОЛНЕННЫМИ ДАННЫМИ


```
{
    "client": "0900000001",
    "currency": "UGX",
    "date": "2025-08-27T08:29:16.784Z",
    "description": "William Shakespeare. The Tragedy Of Hamlet, Prince Of
Denmark",
```

```
    "id": "20250827004",
    "params": [{
        "code": "customerName",
        "value": "Johnson John"
    }, {
        "code": "customerEmail",
        "value": "test@mail.com"
    }, {
        "code": "phoneNumber",
        "value": "229714748036"
    }],
    "product": "Покупка книги",
    "redirectUrlError": "https://merchant.ru/server/error",
    "redirectUrlSuccess": "https://merchant.ru/server/success",
    "service": 2,
    "sum": 100
}
```

D. ПРИМЕРЫ ЗАПРОСА И ОТВЕТА ДЛЯ МЕТОДА СОЗДАНИЯ ЗАКАЗА НА ВЫПЛАТУ

Пример запроса на выплату:

```
{
  "client": "79998887766",
  "currency": "RUB",
  "date": "2025-12-18T03:47:35.942Z",
  "description": "Выплата денежных средств по кредитному договору №22-
126845/59",
  "id": "122",
  "params": [
    {
      "code": "userId",
      "value": "546543274"
    }
  ],
  "product": "Кредит",
  "redirectUrlError": "http://error",
  "redirectUrlSuccess": "http://success",
  "service": 4341,
  "sum": 100000
}
```


Web: https://soft-logic.ru/
Mail: info@soft-logic.ru

Пример ответа на запрос на выплату:


```
{
  "error": 0,
  "id": "122",
  "state": 0,
  "stateDescription": "NEW",
  "orderId": 868,
  "redirectUrl":
"http://localhost:8080/psp/payment-widget/order/49f33088c00dca7ebb8711e63
377729a876c1f1b96160ddd831541d791f88d9170cc8128ef96a947cc62152b86081ed52c
2619f5096409ce6092b1b6d13d7ef16b9474841f387ac8f3aff69d44dd58f45b5c4f36ce3
d461b1fcb0af9755cf6143bff71320966c9aa13f5888843fb82c69018f2b2a2d1486588f0
534478586c3f",
  "date": "2025-12-18T08:02:51.396+0000"
}
```


Е. ПРИМЕР ОТВЕТА НА ЗАПРОС БАЛАНСА МЕРЧАНТА


```
{
  "accounts": [
    {
      "awaitingEnrollment": 0,
      "balance": 5793817,
      "available": 3898167,
      "currency": "RUB",
      "rollingReserve": 20000
    },
    {
      "awaitingEnrollment": 0,
      "balance": 1605200,
      "available": 1605200,
      "currency": "USD",
      "rollingReserve": 0
    },
    {
      "awaitingEnrollment": 0,
      "balance": 10000,
      "available": 10000,
      "currency": "USDT",
      "rollingReserve": 0
```

```
    },
    {
      "awaitingEnrollment": 0,
      "balance": 0,
      "available": 0,
      "currency": "USDC",
      "rollingReserve": 0
    }
  ]
}
```

F. ПРИМЕР WEBHOOK-ОПОВЕЩЕНИЯ

```
{
  "id" : "external-id-123456",
  "transaction" : 1391191,
  "status" : "SUCCESS",
  "errorDetail" : null,
  "error" : 0,
  "message" : null,
  "bankPayment" : {
    "status" : "SUCCESS",
    "transaction" : "ps_13755949",
    "attributes" : null
  },
  "date" : "2025-05-18",
  "attribute" : [ {
    "code" : "merch-url",
    "value" : "https://test.ru",
    "title" : null,
    "valueTitle" : null,
    "flags" : null
  }, {
    "code" : "id1",
    "value" : "",
    "title" : null,
    "valueTitle" : null,
    "flags" : null
  }, {
    "code" : "id2",
    "value" : null,
    "title" : null,
    "valueTitle" : null,
    "flags" : null
  } ],
```

```
  "menu" : {
    "service" : 5,
    "serviceName" : "Оплата через СБП",
    "category" : "Списание СБП"
  },
  "sumOutcome" : 24796
}
```