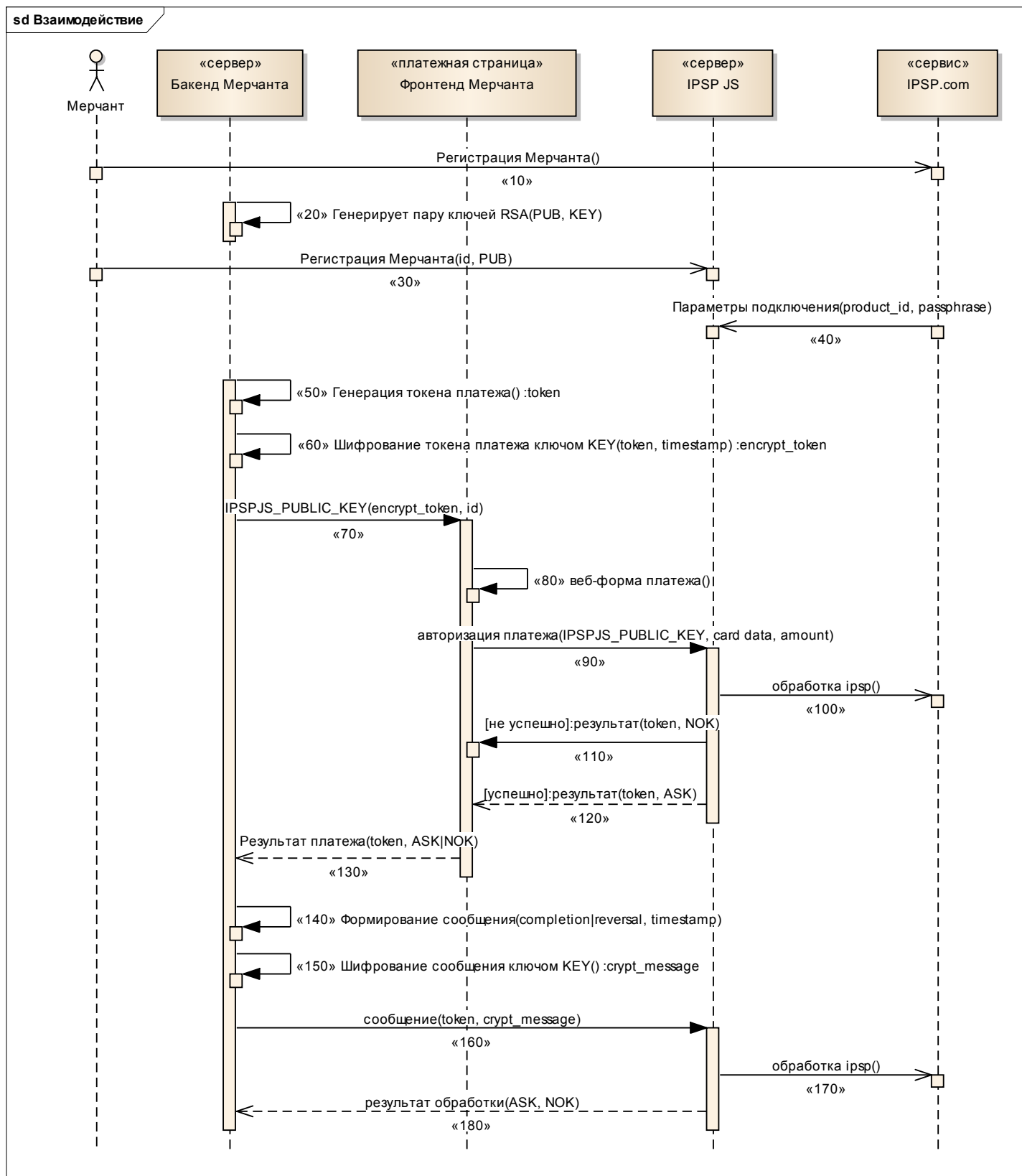


## Спецификация подключения (демо-версия)



## Опции подключения

Id мерчанта для IPSPJS	1 (дописывать в запрос - 000001)
Id продукта в IPSP	1723
Пара ключей	Письмом
Логин и пароль для <a href="https://test1.ipsp.com/merchant/">https://test1.ipsp.com/merchant/</a>	
Отладочная (готовая клиентская) форма платежа (auth)	
Тестовые данные банковских карт (Банк Москвы)	<a href="https://ipspjs.kredy.ru/test/">https://ipspjs.kredy.ru/test/</a>
Java script для платежной формы	<a href="https://ipspjs.kredy.ru/test/ipspjs.js">https://ipspjs.kredy.ru/test/ipspjs.js</a>
Отладочная форма подтверждения/отмены платежа (void / sale completion)	<a href="https://ipspjs.kredy.ru/test/complete.php">https://ipspjs.kredy.ru/test/complete.php</a>

---

10

---

Мерчант заключает соглашение с сервисом IPSP

---

20

---

Сервер Мерчанта генерирует пару ключей RSA: публичный (PUB) и приватный (KEY)

---

30

---

Мерчант передает свои данные в IPSPJS: идентификатор id и ключ PUB

(настроено id=1)

---

40

---

IPSPJS получает и конфигурирует параметры подключения Мерчанта к сервису IPSP согласно протоколам сервиса (product\_id, passphrase)

Сейчас настроено: product\_id = 1723 (логин и пароль от <https://test1.ipsp.com/merchant/> отправлю отдельно)

---

50

---

Сервер Мерчанта генерирует уникальный строковый идентификатор платежа (далее: **токен**). Токен должен быть уникальным и иметь длину ровно 32 символа.

Если в этом есть необходимость, сервер Мерчанта может сохранять **токен** в своей базе данных в связи с конкретным платежом и/или плательщиком.

Токен должен быть уникальным и иметь длину ровно 32 символа.

---

60

---

Ключи RSA должны быть сформированы по стандарту OpenSSL, т.е.:

- 1. Формат ключа — PEM, длина — не менее 2048 бит;
  - 2. Хэш — SHA1;
  - 3. Стандарт подписи — PKCS #1, padding.
- Ключ для формы запроса к IPSP JS (IPSPJS\_PUBLIC\_KEY) должен состоять из:
- 1. 6-ти символов ID публичного ключа RSA мерчанта (PUB) в системе IPSP JS. Значение ID дополняется нулями перед ним до 6 символов;
  - 2. 32-х символов токена;
  - 3. 10-и символов времени создания ключа (unix timestamp);
  - 4. Подписи этих данных (ID, token, timestamp) приватным ключом RSA мерчанта.

Последовательность параметров важна.

Пример ключа:

000001ac24b17cce7b4412b8312d918b8f797d14089506658a0a6471676a6929f79ff0dbce2ad5e0f9666acba04be9a0fdf08cf62ce7ec83f6ba09fbc2268a9b625dda916df776592c7fddcba929275cf094afe468bfee422ef9824677a150b846df066c3847ba14547cb981d8b865d0ef0bd6bb02c627147b5d560209deb99f1889afcb856d3d12064e86ef7e9a2887ffa1f4446a945a5c

Пример ID публичного ключа (PUB) в конце токена, если он 1:

000001

70-80

Генерируемая мерчантом веб-форма платежа должна содержать:

- 1. IPSPJS\_PUBLIC\_KEY как параметр JavaScript\*;
  - 2. Подключенный JavaScript-модуль ipsrjs.js;
  - 3. Платежную форму с обязательными полями;
  - 4. JavaScript-методы инициализации запроса к IPSP JS и обработки его результата.
- \* для реализации некоторых JavaScript-методов (на тестовой форме) используется библиотека jQuery.

Обязательные поля платежной формы:

Имя	Описание
amount	Сумма платежа
currency	Валюта платежа
number	Номер карты
exp_month	Месяц даты истечения действия карты
exp_year	Год даты истечения действия карты (4 цифры)
cvc	CVC карты
cardholder	Держатель карты

Пример инициализации IPSPJS\_PUBLIC\_KEY:

```
<script type="text/javascript">
    var IPSPJS_PUBLIC_KEY = '';
</script>
```

Пример подключения ipspjs.js:

```
<script src="https://ipspjs.kreddy.ru/test/ipspjs.js"></script>
```

Пример JavaScript-кода инициализации платежа:

```
var params = {
    amount: $('.card-amount-int').val(),
    currency: $('.card-currency').val(),
    number: $('.card-number').val(),
    exp_month: $('.card-expiry-month').val(),
    exp_year: $('.card-expiry-year').val(),
    cvc: $('.card-cvc').val(),
    cardholder: $('.card-holdername').val()
};

ipspjs.createToken(params, IspjsResponseHandler);
```

Пример JavaScript-кода обработчика результата авторизации платежа:

```
function IspjsResponseHandler(error, result) {
    if (error) {
        // обработка ошибки error.message
    } else {
        // обработка токена result.token
        // например, отправка токена на сервер мерчанта
    }
}
```

---

90-120

---

Скрипт ipspjs.js дополняет платежные данные и формирует запрос к IPSP JS, получает ответ и отдает его в метод-обработчик. Фактически требуется только указание адреса IPSP JS в коде ipspjs.js, вызов ipspjs.createToken и реализация IspjsResponseHandler().

Метод-обработчик может получить от IPSP JS 3 варианта ответа:

1. Авторизация платежа успешна;
2. Требуется пройти 3DS;
3. Ошибка.

В первом случае ipspjs.js просто вернет токен в метод-обработчик.

Во втором пользователь будет переадресован на страницу прохождения 3DS. После прохождения пользователь будет перенаправлен обратно на страницу веб-формы платежа с GET-параметрами, зависящими от результата прохождения 3DS:

Пример ответа (успех):

```
?token=c2a579535510b4424858cfcd8d70ae19
```

Пример ответа (ошибка):

```
?token=c2a579535510b4424858cfcd8d70ae19&message=fatal%20error
```

Данные параметры должны быть получены JavaScript-методом из строки адреса и переданы методу-обработчику `IpsjsResponseHandler()`.

Пример кода, обрабатывающего результат прохождения 3DS:

```
var UrlVars = getUrlVars();
if (UrlVars.message) {
    UrlVars.message = decodeURIComponent(UrlVars.message);
    IpsjsResponseHandler(UrlVars, null);
    return;
} else if (UrlVars.token) {
    IpsjsResponseHandler(null, UrlVars);
    return;
}
function getUrlVars() {
    var vars = [], hash;
    var hashes = window.location.href.slice(
        window.location.href.indexOf('?') + 1
    ).split('&');
    for (var i = 0; i < hashes.length; i++) {
        hash = hashes[i].split('=');
        vars.push(hash[0]);
        vars[hash[0]] = hash[1];
    }
    return vars;
}
```

Методы инициализации платежа (при использовании jQuery) и обработки результата должны быть внутри:

```
$(document).ready(function () { ... })
```

---

140-160

---

Запрос на завершение платежа должен содержать имя действия, токен, timestamp, тип действия (подтверждение — completion, отмена — reversal) и подпись этих данных приватным ключом RSA мерчанта.

Пример данных сообщения (подтверждение):

```
action=authPaymentCompletion&token=ac24b17cce7b4412b8312d918b8f797d&type=completion&timestamp=1408952868&sign=82
2bce7ceb9368bca8ec3a16848ef420a09e1a2e56a3cc998aa4c7cad28f4c7f10ddb883b3eda515337be919a7154fc85a364846309bd7380
67f90d0b7798e8a
```

---

180

---

Ответ IPSP JS будет в формате JSON. Может содержать: код ответа (ASK/NOK), код ошибки и сообщение об ошибке.

Пример ответа (успех):

```
{
    "result": "ACK"
}
```

Пример ответа (ошибка):

```
{
  "result": "NOK",
  "error": {
    "code": 400,
    "message": "Wrong token"
  }
}
```

## Регистрация продавца

Система продавца генерирует пару ключей RSA, и регистрируется на IPSPJS, передавая ему публичный ключ и какие-то дополнительные данные для идентификации.

## Начальная платежная форма на стороне клиента

Это является описанием уже работающего прототипа в интеграции с ipsp (по Банку Москвы). Схема подключения скриптов на платежную страницу – полностью аналогична пэймилу (работает собственно, их скрипт).

Условия:

- Форма должна быть создана и загружена в веб-браузере на какой-то странице «продавца».
  - Подразумевается, что сама ссылка на страницу, идентифицирует плательщика и «знает», за какие услуги/товары производится оплата
- Это должен быть https-протокол
- В исходном коде должен быть указан публичный ключ продавца
- Страница подключает файл bridge.js с сервера IPSPJS

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8">
    <link rel="stylesheet" href="https://netdna.bootstrapcdn.com/twitter-bootstrap/2.2.1/css/bootstrap.min.css">
    <script type="text/javascript">
      var PAYMILL_PUBLIC_KEY = '1733954763024fa6ea94587027b0f49d';
    </script>
  </head>
  <body>
    <div class="container span8">...</div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js"></script>
    <script src="https://netdna.bootstrapcdn.com/twitter-bootstrap/2.2.1/js/bootstrap.min.js"></script>
    <script src="bridge.js"></script>
    <script language="javascript" type="text/javascript">...</script>
  </body>
</html>
```

IPSPJS, получив запрос, создает токен и сохраняет к этому токенту публичный ключ и всю информацию из запроса. Потом IPSPJS передает запрос в IPSP и получает ответ – либо сразу токен, либо «надо пройти 3ds» (и токен после прохождения 3ds).

(Одновременно по токenu может быть определен конкретный «продавец», но это здесь не играет роли)

## Вариант с включенным на карте 3DS

Это отличается от пэймила только потому, что Банк Москвы информирует браузер о том, что данная страница не должна открываться во фрейме (!), и «послушные» браузеры ее не открывают во фрейме. Если для платежной страницы может использоваться браузер, которому «наплевать» на это (я пока не знаю как с этим будут работать браузеры в смартфонах или компоненты webview), то отличий от подключения «как у пэймила» не будет совсем.

Итак, плательщик нажимает «оплатить» и перенаправляется на страницу 3DS

Joint-Stock Commercial Bank - Bank of Moscow OJSC [RU] <https://3ds2.mmbank.ru/acs2/pa?id=380676700468646>

Приложения Bootstrap-wysiwyg Redmine admin HTML5 Please - Use ... pma.inspector.kred... Kreddy - Личный ка... Mail.ru

**VERIFIED by VISA** **MyBank**

Введите ваш пароль

Пожалуйста, введите ваш Verified by VISA пароль в поле снизу для подтверждения покупки. Эта информация не доступна интернет магазину.

Магазин: Test\_creditpilot  
Описание: Description  
Сумма: RUB 22.87  
Дата: 07/31/2014  
Номер карты: \*\*\*\* \* 1342  
Личное приветствие: HIITEST

Пароль:

[Изменить](#) [Забыли пароль?](#)

[Выйти](#) [Отправить](#) [Помощь](#)

В IPSP появляется платеж типа «авторизация», ожидающий прохождения 3DS:

Id	Date	R...	Type	Status	Amount	Card
615...	2014-07-31 12:20:42		Authorization	Antifraud Passed	22.87 RUB	VISA 465206*****1342

Плательщик вводит правильный код и перенаправляется обратно на страницу платежа.

В IPSP платеж является «approve»:

Id	Date	R...	Type	Status	Amount	Card
615101	2014-07-31 12:20:42		Authorization	Approve	22.87 RUB	VISA 465206*****1342

Обратный переход содержит токен для идентификации платежа и дальнейшего его подтверждения:



The screenshot shows a web browser window with the URL `ipsp-js-bridge.local/test/?token=6dd81b56d1ec94febfed926f859dea6f`. The token is highlighted with a red box. Below the browser window is a payment form with the following fields:

- Card type: **Mastercard (Non 3DS)** / Visa (3DS)
- Card number: 5417150774572399
- CVC: 759
- Card holder: TEST CARDHOLDER
- Valid until (MM/YYYY): 11 / 2014
- Amount: 22.87
- Currency: RUB
- Button: **Buy now**

Below the form is a green box labeled "Token" containing the token value: 6dd81b56d1ec94febfed926f859dea6f, which is also highlighted with a red box.

## Если 3DS на карте нет

То платательщик никуда не перенаправляется и сразу в ответе от IPSPJS (через js-скрипт) получает токен.

## Что теперь делать продавцу

Подразумевается, что система «продавца», получив обратный переход на ссылку, и зная исходя из этой ссылки, «кто» и «за что» платит, сохраняет токен в своей системе, в связи с конкретным платежом.

Далее, платеж может быть отменен или подтвержден из интерфейса IPSP:

Id	Date	R...	Type	Status	Amount	Card
615101	2014-07-31 12:20:42		Authorization	Approve	22.87 RUB	VISA 465206*****1342

Details

Sales Completion

Void

Или, может быть составлен запрос от продавца, зашифрованный приватным ключом:

token = 6dd81b56d1ec94febfed926f859dea6f

message = AKJUI&\*&^X\*CVGIUT\*Q&%#&\$^IUGIV \*(&^(^#(TIUGFIGKV ===


IPSPJS, получая такой запрос

- 1) Проверяет наличие токена – найдя токен – понимает, по какому платежу пришел запрос.
- 2) Расшифровывает сообщение, используя публичный ключ продавца

Сообщение может содержать две команды –подтвердить платеж или отклонить.

В случае команды «подтвердить», платежу проводится операция Sale Completion.

В IPSP – платеж успешно завершен:

Id	Date	R...	Type	Status	Amount	Card
615101	2014-07-31 12:20:42		Authorization	Authorization completed	22.87 RUB	 465206*****1342

## Безопасность

Платежная форма может быть размещена на любом ресурсе. В данном варианте реализации IPSPJS будет обеспечивать условную безопасность – определять – откуда приходят запросы, сравнивать их с информацией о продавце (url, referrer, user-agent и т.п.). При любом подозрении, что это идет запрос с «неправильного» ресурса – отказывать на самом старте. Также могут быть реализованы ограничения и лимиты на количество операций в период времени, и при «нехороших» тенденциях – отключение «продавца» до принятия ручного решения. При таком подходе, тем не менее, всегда есть риск получить «левые» платежи, но – в относительно небольшом количестве.

100% достоверность источника платежа – может обеспечить только зашифрованный приватной частью ключа «продавца» токен. То есть, токен не создается «IPSPJS», а создается «продавцом». «IPSPJS» идентифицирует продавца в случае успешной расшифровки токена. Для реализации данной схемы, система «продавца» должна при генерации платежной страницы, создавать зашифрованный токен и размещать его на странице (см. выше) вместо публичного ключа.