SPARROW

Transact, Grow, SPARROW.

Sparrow Client (Front-end) API

Service API

Version 3.6.0 (Build 8062)

Released May 2017

Revision History

Date	Revision	Comments	Author
2017-05-22	1.0	Initial document	Ilya Tretyakov

Table of Contents

Vocabulary

SCA	Sparrow Client (Front-end) API Initial document
TT	Payment's temporary token
EF	Embedded Form
CF	Custom Form

Overview

This guide was made for technical staff and other specialists who maintain sites, online shops and other sources, which provide payment opportunity through the "Sparrow global payment solution".

The “Sparrow Client (Front-end) API” (hereinafter referred to SCA) allows merchants to receive payments without storing payment information on their servers (PCI DSS license is not required) as well as without using "Payment Redirect". Therefore, this possibility suggests using JavaScript at the sales resource (merchant's site).

Important! At the current version, SCA provides only the simple credit card payment, which includes such fields as Credit Card Number, Cardholder, Expiration Date and CVV (CVC).

Interaction Scheme

The interaction scheme consists of two consecutive requests, each of which is directed into the same server "Sparrow".

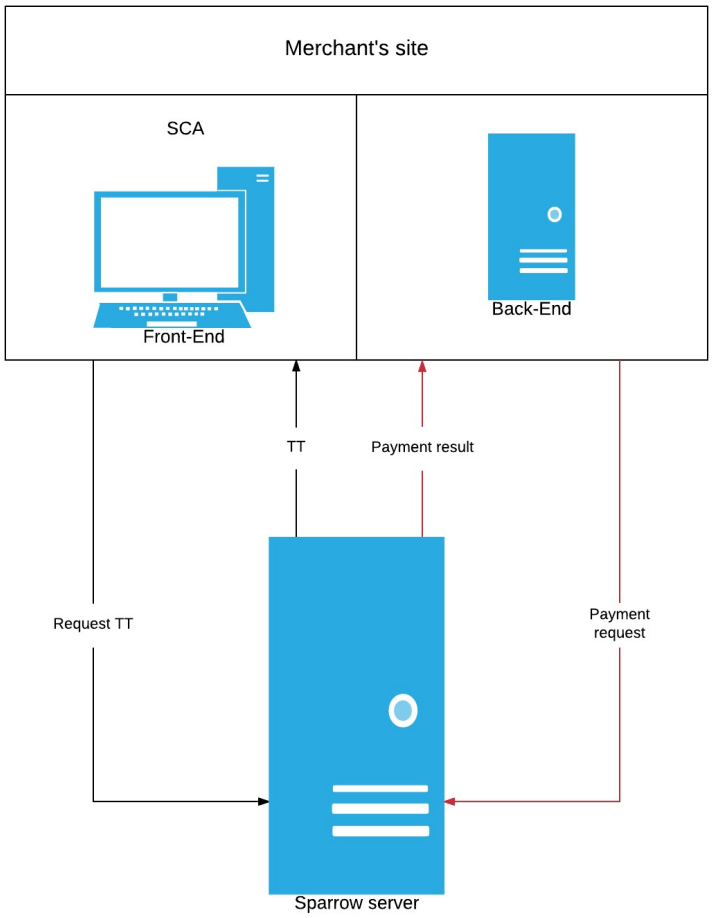
The first request:

- 'POST' from merchant's site front-end using JavaScript, the second request - 'POST' from merchant's server-side.
- Sends payment information (credit card number, cardholder name, CVV etc.) and receives payment temporary token (hereinafter referred to TT).

The second request:

- 'POST' from merchant's server-side.
- Makes a payment using TT and other special info.

The following scheme shows this interaction:



:

How does it work

There are two different ways to use it: Embedded Form (hereinafter referred EF) and Custom Form (hereinafter referred CF).

Embedded Form (EF)

SCA allows you not to think about html-layout of payment forms and JS-logic of receiving TT. If your task is to draw one or more payment buttons for each good (for example) and you do not want to spend your time on custom form layout - EF is your choice. SCA provides mechanism of drawing payment form based on its own JavaScript code and CSS "from the box". Your duties will by only add SCA CSS file on your page besides SCA JS, rendering particular tags with "data-" attributes as well and making an AJAX function for back-end logic of payment. You can also redefine our CSS styles in accordance with your site's theme.

Custom Form (CF)

In this case, you have to write all clients and servers logic independently. To obtain a TT you use SCA function "getToken" (see "API description" section).

Using Embedded Form (EF)

1. Add following JS- and CSS-files at the "head" of your page (**data-name="sparrowApi" attribute is necessary!**):

```
<link                href="https://secure.5thdl.com/clientapi/sparrow-api.css?v=1.1"
rel="stylesheet" />
<script                 src="https://secure.5thdl.com/clientapi/sparrow-api.js?v=1.1"
type="text/javascript" data-name="sparrowApi"></script>
```

   Make sure these files are loaded successful.

2. Configure payment buttons at the "body" section of your page:

```
<div data-pkey="[Your public key]"
     data-buttontext="Pay $200.4"
     data-title="One thing by $200.4"
     data-tokencallback="embeddedFormCode.getTokenCallback"
     data-purchasebuttontext="Purchase $200.4"
     data-amount="200.4">
</div>

<br />

<div data-pkey="[Your public key]"
     data-buttontext="Pay &#8364;3.45"
     data-tokencallback="embeddedFormCode.getTokenCallback2"
     data-amount="3.45">
</div>
```

The assignment of the attributes see at the “API description” section. “embeddedFormCode.getTokenCallback” and “embeddedFormCode.getTokenCallback2” are defined below.

3. Now let’s write some JS-code (with comments along the text). **Be careful, your custom Ajax function for payment should return "Promise" object (such as "iQuery.Deferred").** Deferred object has to involve either “done” or “then” methods. It's necessary for right "loading-animation" behavior EF:

```
<script type="text/javascript">
      // Configure EF if it is necessary (see “API description” section)
      Sparrow.modals.disableClose = false;
...
```

```
// Embedded form code example7
        var embeddedFormCode = (function () {

            var embeddedFormCallback = function (response) {
                Sparrow.closePopups(); // you may close popup window after purchase
                // use response here
            },
            // User purchase function should returns Deferred object (jQuery Deferred
for example)
            embeddedPurchase = function (token, amount) {
                return [Your Purchase function](token, amount)
                    .done(embeddedFormCallback);
            }

            // callbacks which you use at 'data-tokencallback' attributes
            return {
                getTokenCallback: function (token) {
                    return embeddedPurchase(token, 200.4);
```

Using Custom Form (CF)

The example involves only “Sparrow.getToken” function call. Do it where you want. Then use callback for further payment operation.

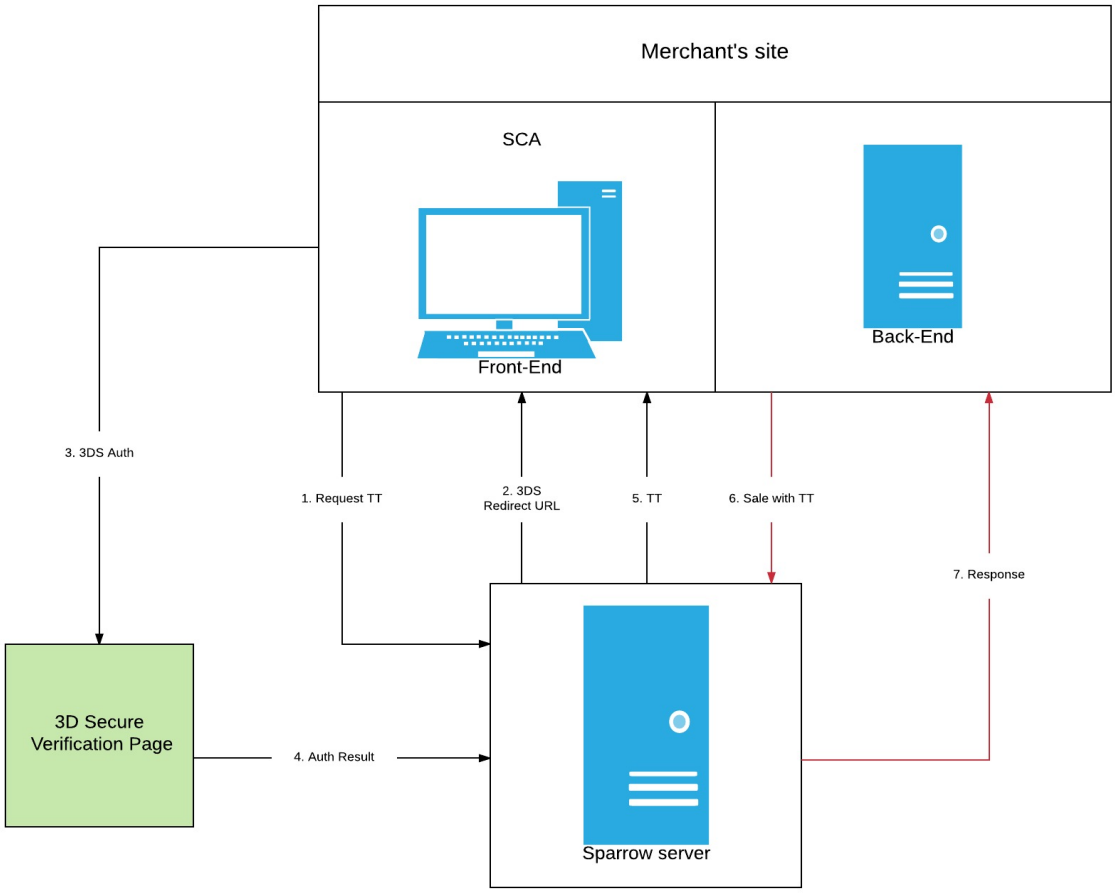
```
Sparrow.getToken([Your pkey], {
    cardnum: [Buyer Card Number],
    cardholder: [Cardholder],
    cardexp: [Month + Year], // Format string “MMYY”
    cvv: [CVC],
    amount: [Amount] // Required for 3D secure verification
```

3D Secure Verification

3-D Secure is an authenticated payment system to improve online transaction security and encourage the growth of e-commerce payments.

The interaction scheme with credit card verification consist of following steps:

1. Merchant sends the 'POST' request from merchant's site using JavaScript.
2. Sparrow sends to merchant response with redirect URL for 3D secure verification.
3. Sparrow API redirects consumer to the 3D secure verification page, which will open in the popup window.
4. Consumer pass 3D secure verification and card issues send to Sparrow 3D secure authentication data.
5. Sparrow generate and send to merchant site TT.
6. Merchant sends to sparrow sale request with TT.
7. Sparrow sends to merchant response with sale operation result.



Notes:

- Please, contact administrator to enable 3D secure verification feature
- The field **'amount'** is required for 3D secure verification. Don't forget to add "data-amount" attribute to Embedded form (example on page 7), or "amount" parameter to "data" object for Custom Form (example on page 9)
- To avoid caching, please, use postfix with Client API current version for JS- and CSS-files at the "head" of your page (example on page6).

API description

EF popup settings

Property	Description	Values	Default
`Sparrow.modals`	Object, which stores popup window behavior	JSON	
`Sparrow.modals.disableClose`	Allows/forbids closing the popup if the target control is disabled	true / false	false
`Sparrow.modals.bgClose`	Allows/forbids closing the popup onto background (overlay) click	true / false	false
`Sparrow.modals.escapeClose`	Allows/forbids closing the popup using Escape button	true / false	false
`Sparrow.modals.fadeOutOpen`	Switch on / off the simple animation during the popup window opening	true / false	true

EF tag attributes description

Attribute	Description	Default	Required
data-pkey	Merchant Public Key		yes
data-buttontext	Caption on the button which opens popup window	Purchase	no
data-title	The title of the popup window	Purchase	no
data-purchasebuttontext	Caption on the purchase button	Purchase	no
data-amount	Purchase amount, required for 3D secure verification		Only for 3D secure verification
data-tokencallback	Callback function that will be called after receiving the token. **Has to return Promise (jQuery.Deferred for example)**		yes

"getToken" function description (CF)

Parameter	Description	Required
pkey	Merchant Public Key string	yes
data	JSON object – data which will be required for payment holding	yes
success	Success function that will be called after receiving the token	yes
error	Error function that will be called if the error appears	no
complete	Complete function. It will be called after the ajax request finish anyway	no