ALFEBA
PROCESSING SOLUTIONS

SEQUOIA MOSAIC 3000:

INTERNET-ACQUIRING PLATFORM

Merchant manual

User's manual for GPS merchants

Version 1.0

This page doesn’t contain any information

Content

Chapter 1. About the document 5
1.1. Purpose of the document 7
1.2. How to use this manual 7
1.3. Classification 7
1.4. Document sheet 7
1.5. Document contacts 7
1.6. Document history 7
Chapter 2. Getting started 9
2.1. General information 11
2.2. Access registration and recovery 11
2.3. Service creation 13
Chapter 3. Payment method setup 17
3.1. General information 19
3.2. Directing the user to a specific payment method 20
3.3. Payment button 22
3.4. Payment button with variables 24
3.5. Short link (order creation) 25
3.6. QR-code 29
3.7. Card payments tests 30
Chapter 4. Application programming interfaces 31
4.1. General information 33
4.2. Data sending format by the partner request 34
4.3. Getting additional parameters about a transaction (webhook notifications) 36
4.4. Transfer of extended transaction data (custom_fields, airline tickets) 38
Chapter 5. Request authentication 41
5.1. General information 43
5.2. Keys for the request authentication 44
5.3. Signature creation algorithm 46
5.4. System IP addresses 53
Chapter 6. Additional options 55
6.1. General information 57
6.2. Background initiation of payment 57
6.3. Requesting information about a transaction 66

6.4. Pre-authorization (additional request about the possibility of making a payment) 69
6.5. Two-stage payments (holding funds on a card) 70
6.6. Refund of payments (refund) 71
6.7. API for sending and receiving payment / refund checks 77
6.8. Payment cancellation (reversal) 78
6.9. Description of the recurring payment scheme (RP) 79
6.10. Saving data of the payer's card 83
6.11. Using tokens (card tokens) with libraries for mobile applications 84
Chapter 7. Reports 87
7.1. General information 89
7.2. Payments report 90
7.3. Transactions report 91
7.4. Configuring notifications about successful transactions 92
7.5. Sending of acts and details by e-mail 93
Chapter 8. Attachments 95
3.1. Terms and abbreviations 97
3.2. External documents references 99

Chapter 1. About the document

This chapter contains the next sections:

Section	Description	Page
1.1.	Purpose of the document	7
1.2.	How to use this manual	7
1.3.	Classification	7
1.4.	Document sheet	7
1.5.	Document contacts	7
1.6.	Document history	7

This page doesn’t contain any information

1.1. Purpose of the document

This document describes how to work with a SM 3000 Internet-acquiring platform as a merchant. This document was prepared for users of the SM 3000 Internet-acquiring platform.

1.2. How to use this manual

The manual is designed to provide the principal information to the User about service registration, payment options setup, light-api (based on the button payment) etc.

The terms, abbreviations and useful references to other documents about the SM 3000 system are provided at the final part of the document.

Terms and Abbreviations - a glossary of terms commonly used in the card processing and electronic funds transfer industry.

1.3. Classification

This document has been classified as External.

1.4. Document sheet

200201

1.5. Document contacts

In the case of questions or proposals about information presented in this document, you can contact Alfeba's Documentation Division by email doc@alfeba.com, by phone +598 2 208 31 42 or by mail, using the address: Av. Agraciada 2770, Montevideo, 11823, Uruguay.

1.6. Document history

Version	Date	Modification	Notes	Authors
1.0	29.07.2020	-	Init. Version	Natalia Bogorodskaya

This page doesn’t contain any information

Chapter 2. Getting started

This chapter contains the next sections:

Section	Description	Page
2.1.	General information	11
2.2.	Access registration and recovery	11
2.3.	Service creation	13

This page doesn’t contain any information

2.1. General information

In this chapter we provide the principal information how you can register independently a new account and immediately start testing payment with bank cards without going to the office, using a Sequoia Mosaic 3000 Internet-acquiring platform [SM3000 IAP].

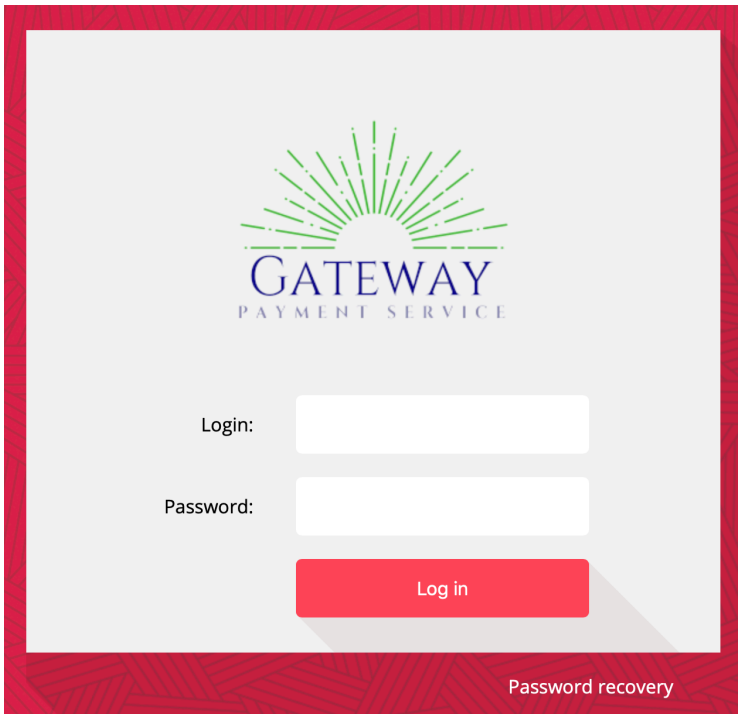
2.2. Access registration and recovery

2.2.1. Registration

To registrar your account with a Platform you have to contact your Payment operator/ Facilitator.

2.2.2. Authorization

To enter to the Platform you have to input your login and password in the authorization form:

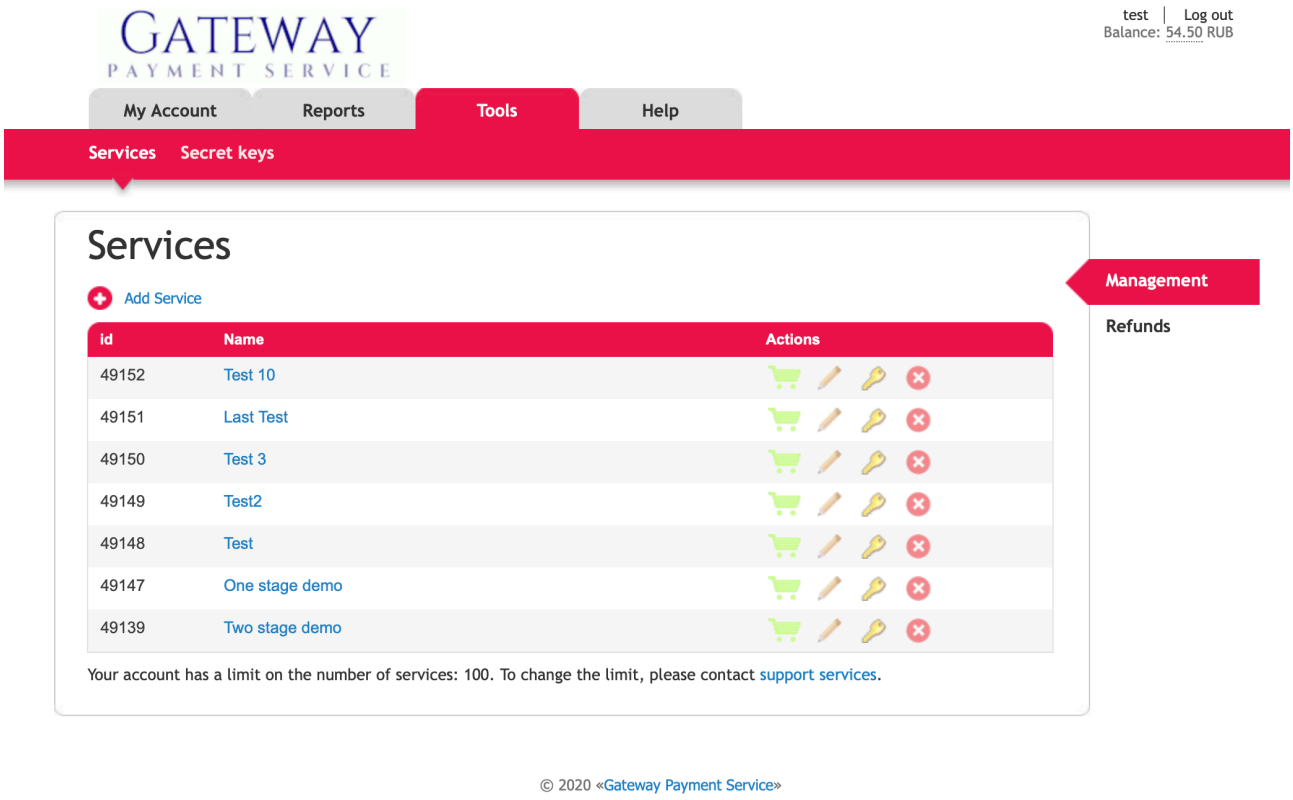


2.2.3. Recovery

To recover your password you have to press the **Password recovery** link on the main page and to input your login name and e-mail:

2.3. Service creation

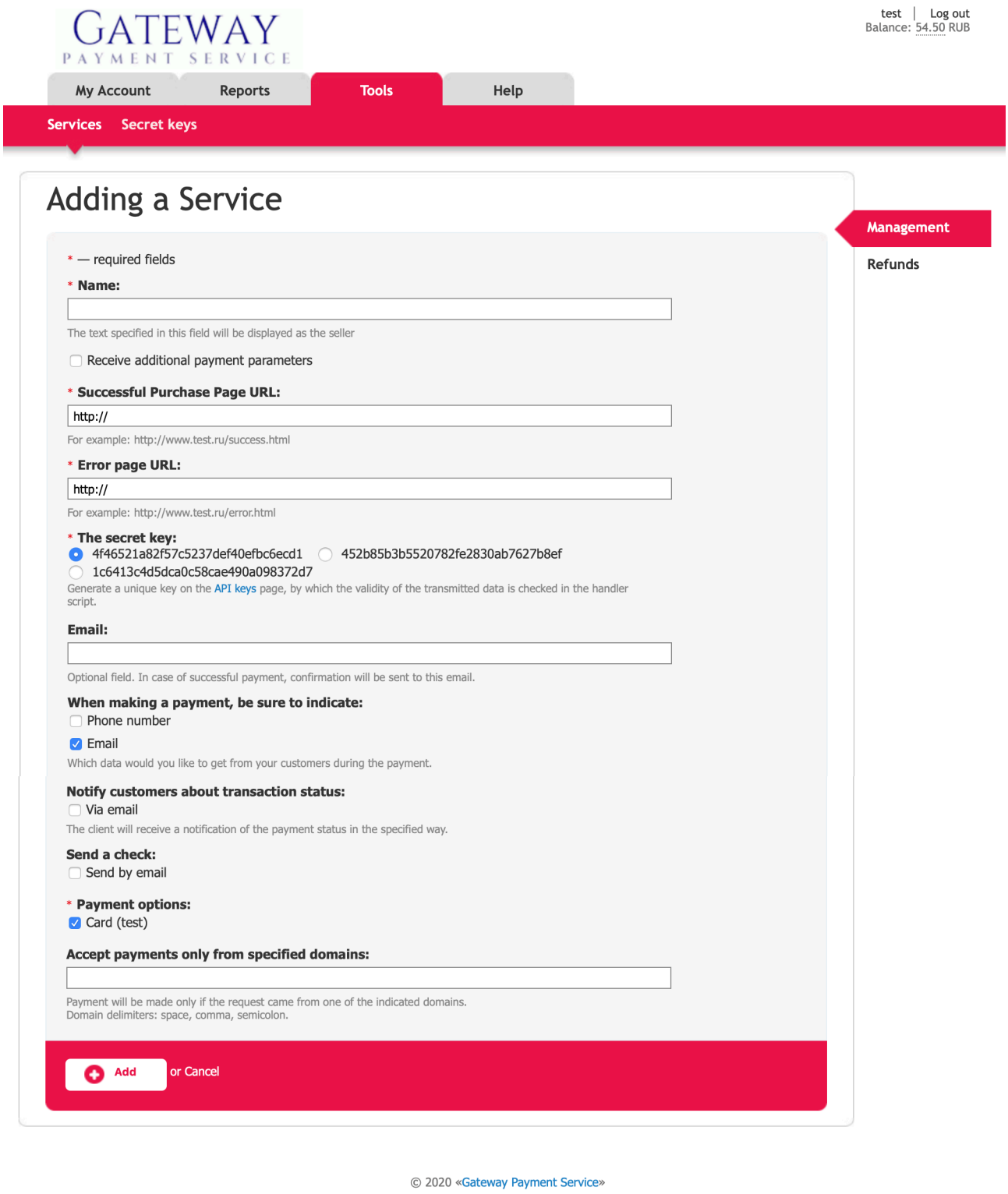
If you have several stores hosted on different servers, we suggest that you set up a separate Service for each site. You will be able to flexibly manage the settings of each Service, while the accounting of transactions will be general.



When a service is identified, its ID (indicated on the left in the list of services) and secret_key are involved.

To create a service, go to the Merchant profile website homepage in your personal account. Then select the section "Tools", subsection "Services" and click on the button "Add service".

You can do it by clicking on the name of the service also.



The fields description is provided in the Table 2.3.0.0.

Table 3.2.0.0. The new service page fields description

Field name	Field format	Data format	Description
Name	Text	Text	The text specified in this field will be displayed as the seller
Receive additional payment parameters	Bool	Bool	Yes/ No option to receive additional payment parameters
Script URL for additional payment options	Text	Script	This script will be called to transfer additional payment parameters. You can receive information on partial payment, etc. If you do not need to perform actions related to accounting for money, then the handler script can be omitted.
Confirmation on product reservation	Logic	Logic	Is an option to confirm on product reservation: • No, • Once, • Every attempt to pay
Successful Purchase Page URL	Link	Link	Is a page to which the user will be redirected after successful payment with the transfer of all payment parameters
Error page URL	Link	Link	Is a page to which the user will be redirected if an error occurs during the payment
The secret key	Text	Text	Is a key by which you can confirm the accuracy of the transmitted data. If you specified the address of the handler script, then you should also specify the secret key
E-mail	Text	Text	Optional field. In case of successful payment, confirmation will be sent to this email
When making a payment, be sure to indicate: Phone number	Bool	Bool	if you need to receive phone numbers from users (to send a username and password, notify, etc.), then using this parameter you can make these fields mandatory.
When making a payment, be sure to indicate: E-mail	Bool	Bool	if you need to receive email from users (to send a username and password, notify, etc.), then using this parameter you can make these fields mandatory.
Notify customers about transaction status: Via e-mail	Bool	Bool	The payer will receive a letter when initiating a transaction, as well as at the end of the transaction.
Send a check: send by e-mail	Bool	Bool	If marked - the customer will receive the ticket of the bought by e-mail
Payment options	Bool	Bool	To choose the payment methods that you want to offer your customers. You cannot choose less than one method.
Accept payments only from specified domains	Bool	Bool	Payment will be made only if the request came from one of the indicated domains. Domain delimiters: space, comma, semicolon.

This page doesn’t contain any information

Chapter 3. Payment method setup

This chapter contains the next sections:

Section	Description	Page
3.1.	General information	19
3.2.	Directing the user to a specific payment method	20
3.3.	Payment button	22
3.4.	Payment button with variables	24
3.5.	Short link (order creation)	25
3.6.	QR-code	29
3.7.	Card payments tests	30

This page doesn’t contain any information

3.1. General information

In this chapter we provide the principal information how you can setup the payment method.

Depending on the specifics of the merchant, you can select one or several options to initiate a payment transaction.

Possible options are provided in the Picture 3.1.0.0. below.

Picture 3.1.0.0. Payment methods options

PAY	Payment button - HTML form with data required for payment	Use on websites of our own design. You can transfer your own payment parameters (amount, purpose, etc.) to HTML form fields.
https://	URL or ShortenURL, after clicking on which the form for choosing a payment method is displayed.	Sending payment links through any messaging systems. In the URL parameters, you can dynamically change the parameters of the payment transaction.
	Link to dynamic graphic image (QR code)	Placing the QR code online on websites or offline on any physical media
	Plug-in module	If your site uses any CRM system, using ready-made modules, you can connect payment in a very short time
	External API - Web service for initiating a payment transaction	This method is suitable for merchants who form their own choice of payment method. When using this mechanism, you can create a payment transaction without displaying the pages of the payment system

3.2. Directing the user to a specific payment method

You can direct the user immediately to the interface of the selected payment system, bypassing the payment selection window. To do this, add the required parameters to the parameters listed above. We describe them in the Table 3.2.0.0.

Table 3.2.0.0. Additional parameters description

Parameter's name	Value	Description
payment_type	the type of payment to which the payer should be sent	The available parameter values are shown in the table 3.2.0.1.
phone_number	payer's telephone number	For example 9841135147
verbose	the parameter specifies what to do in the case of an error, if there is no data about the user (mail or phone for payment methods, where they are required)	1 - shows error 0 - shows the payment selection page
email	customer email	this field is required for recurring payments, for other payment options, the need for input is regulated in the service settings

Available values for the payment_type parameter are presented in the Table 3.2.0.1.

Table 3.2.0.1. Available values for the payment_type parameter

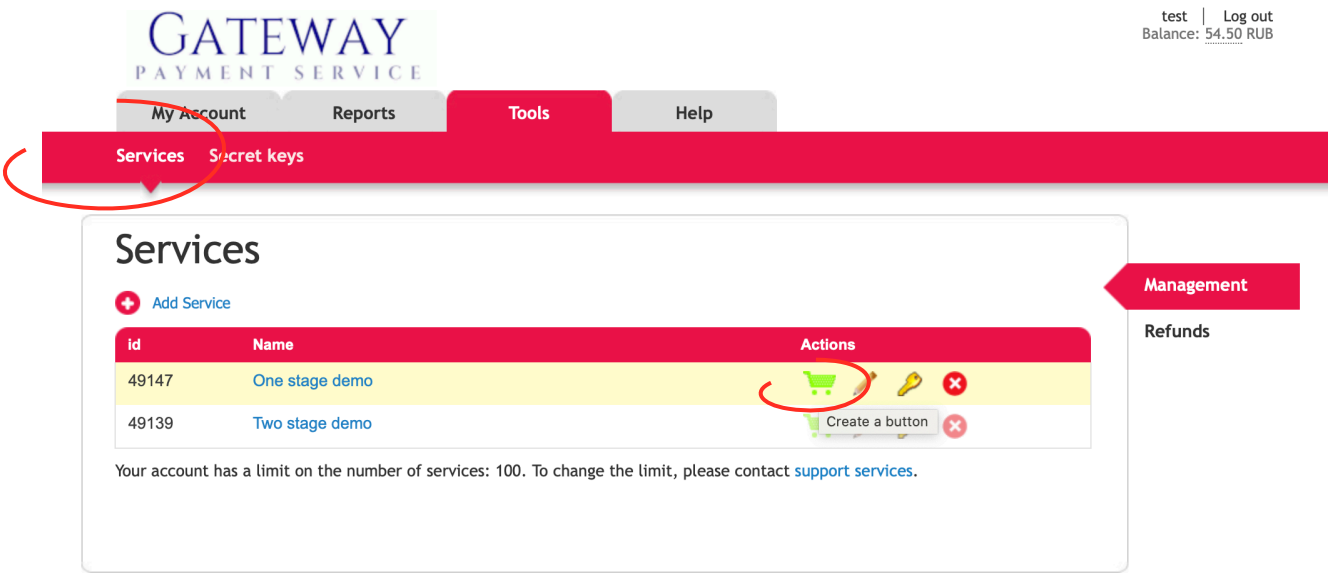
Value	Description
spg	credit cards
spg_test	credit cards, test connection

Example of the credit card payment

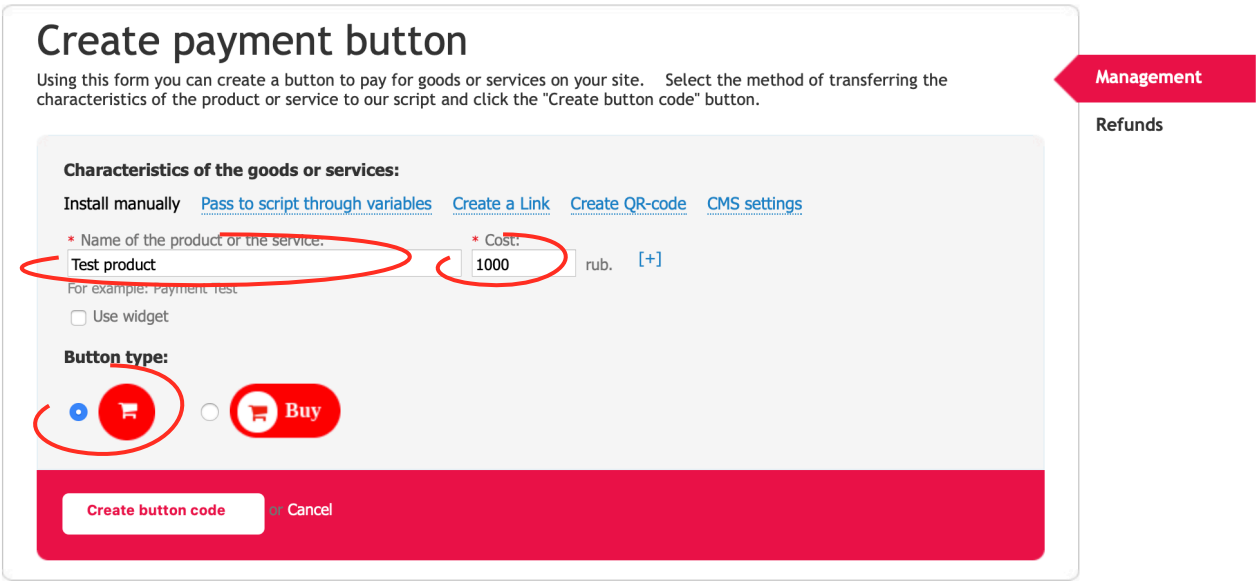
```
1 <form method="POST" class="application" accept-charset="UTF-8" action=«https://partner.gps.com/alba/input/">
2 <input type="hidden" name="key" value="b5/uqup/i/ueWBrRyp9V0n97zyHty5YtV5u/NW27nlk=” />
3 <input type="hidden" name="cost" value="1" />
4 <input type="hidden" name="name" value=«Name of the service"/>
5 <input type="hidden" name="phone_number" value="74951234567"/>
6 <input type="hidden" name="email" value="customer@site.ru" />
7 <input type="hidden" name="payment_type" value="spg" />
8 <input type="hidden" name="verbose" value="0" />
9 <input type="hidden" name="order_id" value="0" />
10 <input type="image" style="border:0;" src=«https://partner.gps.com/gui/images/a1lite_buttons/button_large.png" value="Pay" />
11 </form>
```

3.3. Payment button

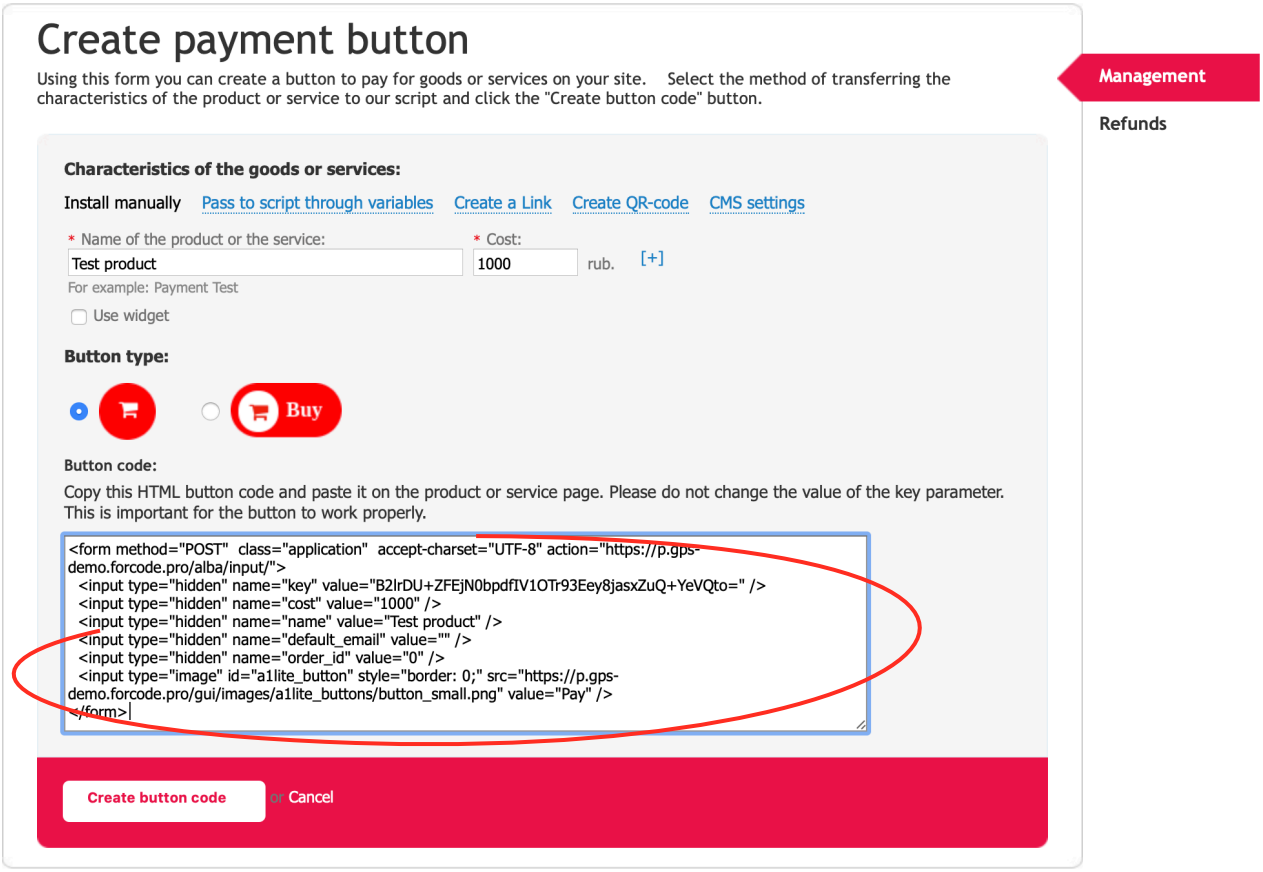
To set up the Button manually you should open the **Service** menu of the **Tools** bar and press the **Basket** pic as shown below:



The Create payment button page will be opened. On this page you should input the **Name of the product or the Service** to the correspondent field of the page and the product price, Button type as shown below:



After the Create button code pressing - the code will be shown in the correspondent field of the page.



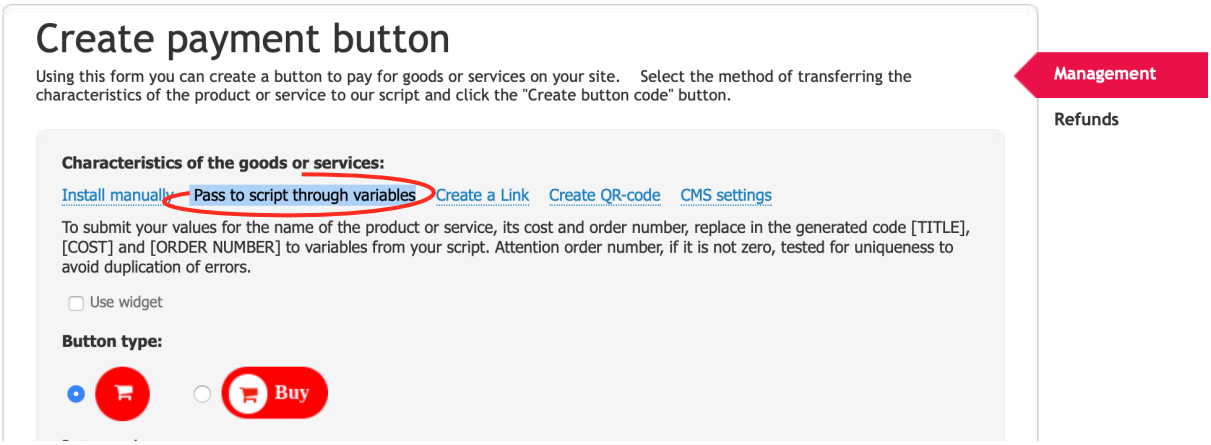
Copy this HTML button code and paste it on the product or service page. Please do not change the value of the key parameter. This is important for the button to work properly.

3.4. Payment button with variables

A variable payment button means that you can create a payment button with your own payment options.

To create a payment button not for a single product, but for different products (for example, if you have an online store), you need the script on your website to substitute the name of the product and its price when generating the button. To do this, when creating a button, select the parameter “Pass to the script through variables” and instead of the parameter values in brackets ([COST], [NAME]) insert the variables of your script.

To do it press link Pass to script through variables on the page of Create payment button:

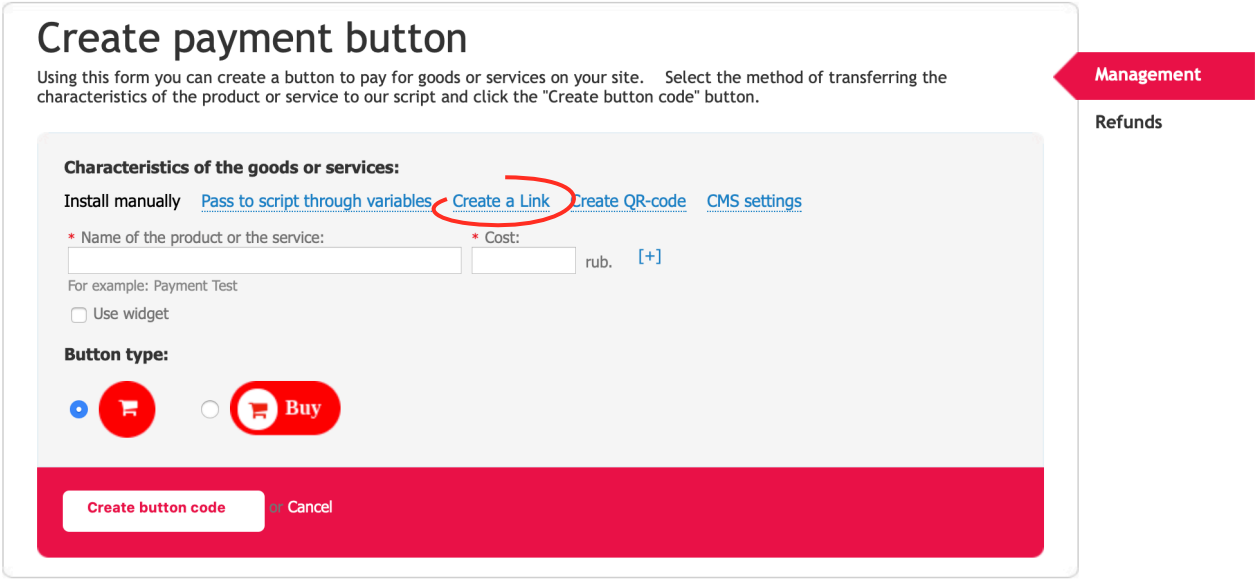


and continue by the same process that is describe in the previos section.

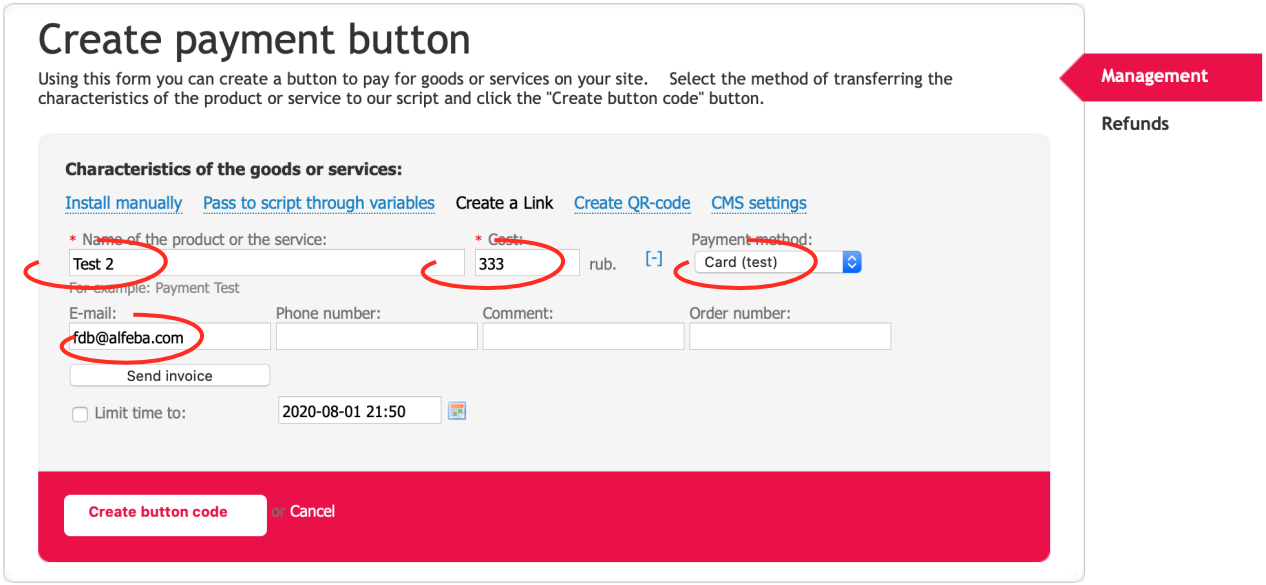
3.5. Short link (order creation)

You can create a custom link that immediately opens the payment window (without having to click a button on the website). This is convenient if you sell products via social networks, issue various invoices for payment, etc.

On the Create payment button page activate **Create a link**:



The correspondent page will be oponed. On the page input **Product name**, **price**, **e-mail** into the correspondent fields and choose a **Payment method** as shown below:



When you click the "Send invoice" button, the link will be sent to your email address.

Attention! A link is a generator. With each click on the link, the payer generates a new transaction. The transaction can be paid within the next 48 hours, but the link is valid indefinitely! If you want to limit the validity of the link, use the appropriate option: check the "**Limit time to**" checkbox and specify the date and time of the expiration.

In the case of "There is already an order with order_id XXXXX. Old number XXXXXXXXX» remember, that it isn't error!

There is a mechanism for checking the order number (order_id) for uniqueness. It's not a mistake. The warning you see in such cases is the result of this check.

A link or button is a generator. Each transition through it is an attempt to create a new transaction with the number that the link is currently transmitting. Transferring the same order number from the store admin (CMS) leads to the same result.

An example of a link containing an order number:

```
https://partner.gps.com/alba/input/?
name=Rent_hall&cost=15200&key=Agl%2FskVOgsU6ZizcKvXjIlWhNJPyYri9x0J%2B
Y9ex6C0%3D&default_email=&order_id=192032
```

An example of an algorithm for receiving a warning:

- The first click created transaction **3950000001** with order number **192032**. Payment failed, the payer closed the browser.
- After a while, the payer clicked on this link again. The bank gateway tried to create transaction **3950000002** with order number **192032**.
- Warning. The existence of two different transactions with the same number is impossible!

If you want to avoid this behavior, pass a new order number for each click on the button from the cart or link. If you continue to receive the warning "**`There is already an order with order_id XXXXX. Old number XXXXXXXXX`**", then your store sends the same message for every click. You can verify the existence of a transaction with the "old" number indicated in the warning by checking the "Reports" section.

3.5.1. Creating a payment link using the API

The service `https://partner.gps.com/alba/build_link/input_short/` should be sent all the parameters that will go to the payment initiation URL (/ alba / input). The response will be in JSON format. If all the data is correct, then there will be status = ”ok”, and the “url” will contain a short link.

Description of available parameters in the Section 4.2. of the Manual.

Example:

Message request format

```
https://partner.gps.com/alba/build_link/input_short/?
name=Тестовая+оплата&cost=10&key=GX4XuBD4myLEaU7k4lSyxvJ0TjVN%2Bhr2uy6OsoAhTm4=&
email=test@test.ru&order_id=0
```

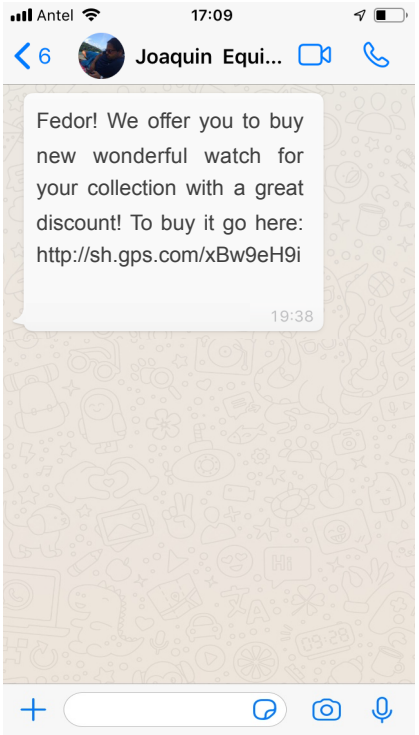
Message response format

```
{“status”:”ok”,”url”:”http:\/\/sh.gps.com\/xxxxxx”}
```

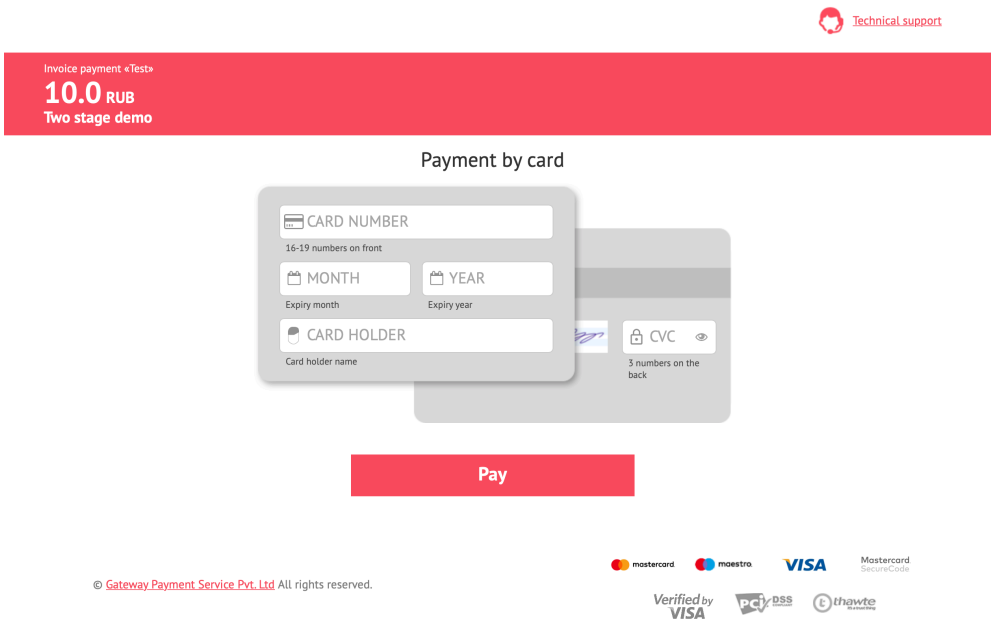
The "key" parameter must not contain the "+" sign.

3.5.2. How does it work?

With the help of a short link, you can send invoices to your customers for payment via your Twitter, Facebook or email newsletter directly. This is how it can look:

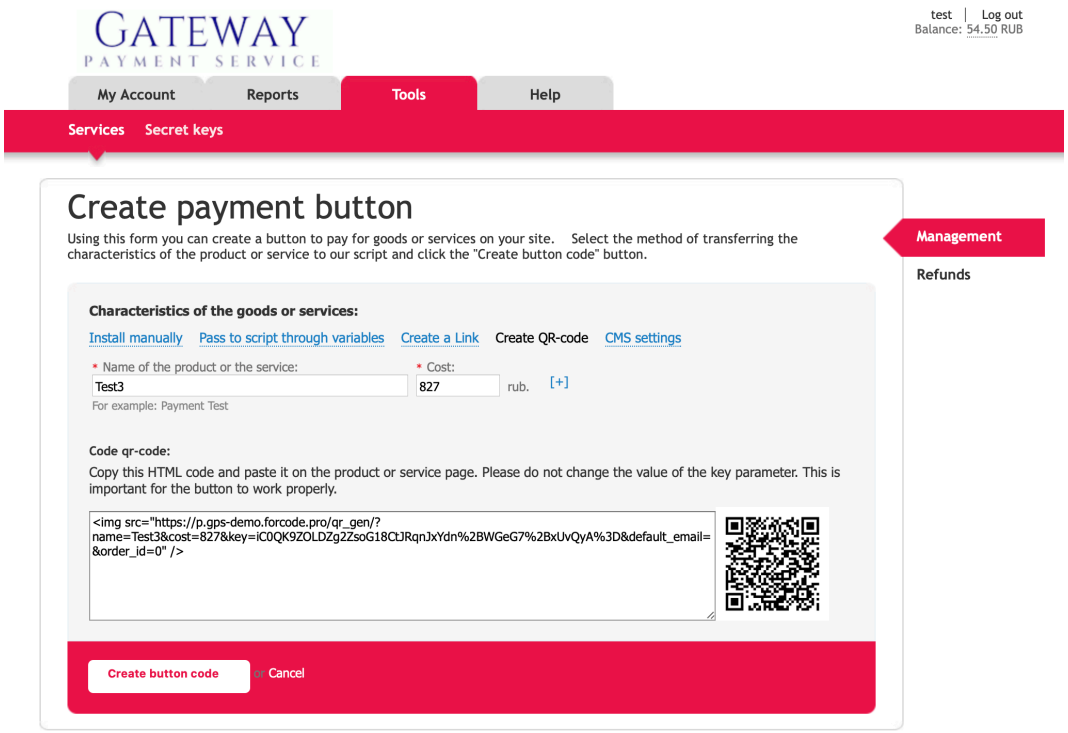


When clicking on the link, the client will see a standard GPS eCommerce payment window:



3.6. QR-code

With this functionality, you can create a special QR code for payment. The main advantage of a QR code is its easy recognition by scanning equipment (including a mobile phone camera), which makes it possible to use it in trade, production, and logistics. Due to the ease of use, QR codes are gaining popularity and they can be seen in everyday life more and more often. Most modern smartphones by default or using special applications can recognize a QR code. You create a QR and can use it in print ads or on a website. The user uses the phone to recognize the code and follows the hard-coded link to the GPS eCommerce payment window adapted for mobile phones. Follow the path **TOOLS** → **SERVICES** in the GPS eCommerce Personal Account merchant profile page. Click on **Create button** next to an existing service. Click on the **Create QR Code** hypertext. Now enter the **product name** and **cost** (additional parameters are hidden by the [**+**] sign) and click on the **CREATE BUTTON CODE**. After that, the HTML code of the picture with the QR code will be generated:



3.6.1. API usage for the QR-code creation

The link for the QR code image can also be obtained through the API: at `https://partner.gps.com/alba/build_link/input_qr/` you should pass all the parameters that will go to the payment initiation URL (/ alba / input). The response will be in JSON format. If all the data is correct, then it will be status = ”ok”, and the “url” will contain the URL of the QR code image.

Description of the available parameters in the Section 4.2. of the Manual.

3.7. Card payments tests

When you create an account, the Visa / MasterCard channel (spg_test) is automatically available for use. Transactions through this channel are a full-fledged transaction without using real money.

You can use both test and any real-life cards issued by Nepal issuers.

The test channel has a 15% commission. A transaction made on such a channel will not change the balance on the payer's card.

You can initiate a test transaction using your personal account. To do this, use the link generator functionality described in the section 3.5. of the Manual.

In the Table 3.7.0.0. we provide test card details.

Table 3.7.0.0. Test cards details

Payment system	Card number (PAN)	Exp. date	CVC
VISA International	4300 0000 0000 0777	11/2022	any
VISA International	4111 1111 1111 1111	12/2024	123
MasterCard International	5555 5555 5555 4444	any > current date	any

Chapter 4. Application programming interfaces

This chapter contains the next sections:

Section	Description	Page
4.1.	General information	33
4.2.	Data sending format by the partner request	34
4.3.	Getting additional parameters about a transaction (webhook notifications)	36
4.4.	Transfer of extended transaction data (custom_fields, airline tickets)	38

This page doesn’t contain any information

4.1. General information

In this chapter we describe how do you can build your own universal solution without resorting to using modules for popular CMS, using the API of the SM3000 IAP.

The API supports POST requests containing JSON arguments. For the successful execution of requests, a number of conditions must be met:

- use correct secret_key
- use correct service_id
- apply Content-Type: application / x-www-form-urlencoded in request header

To read more about Secret keys see the Section 5.2. of the Manual.

To read more about Service ID please see the Section 2.3. of the Manual.

4.2. Data sending format by the partner request

The client is redirected to the Bank's system using the following HTML form.

Request format:

```
1 <form method="POST"  class="application"  accept-
charset="UTF-8" action="https://partner.gps.com/alba/input/">
2 <input type="hidden" name="key" value="b5/uqup/i/
ueWBrRyp9V0n97zyHty5YtV5u/NW27nlk=" />
3 <input type="hidden" name="cost" value="1" />
4 <input type="hidden" name="name" value=«Service name"/>
5 <input type="hidden" name="email" value="mail@example.com" />
6 <input type="hidden" name="order_id" value="0" />
7 <input type="image" style="border:0;" src="https://partner.gps.com/
gui/images/alba_buttons/button_large.png" value="Pay" />
8 </form>
```

Parameters required to initialize the payment are provided in the Table 4.2.0.0.

Table 4.2.0.0. The payment initialization parameters description-1

Parameter name	Value	Description/ example
cost	the amount that the client must pay	100 (if the amount is transferred with cents, then use the "dot" separator, eg 100.65)
name	Description of the paid product / service. Displayed on the payment page.	No more than 128 characters. Example: Order No. 212
email	customer email field is required for recurring payments, for other payment options, the need for input is regulated in the service settings	test@test.com
phone_number	Numerical field, required The order number in the partner's system must be unique. Orders with the same order_id cannot be paid twice. If there is no need to define each order, then the order_id value should be set to 0. The maximum length is 64 characters. For recurring payments, length> = 6 characters	00001 The use of the Latin alphabet is allowed. Cyrillic is not supported (an error will be received when creating a transaction).
comment	Payment comment. You can transfer any of your information through it. The information passed in this parameter is not displayed on the payment page and can be used for the internal needs of the store.	Text field, no more than 512 characters

Parameter name	Value	Description/ example
invoice_data	Data in json format for a fiscal receipt (is used in Russian region only)	htmlspecialchars JSON
custom_fields	Optional parameter. Designed to transfer additional information to various payment channels	urlencoded словарь JSON
check*	Version 2.0 signature is the electronic signature of the request. See Section 5.3. for the further information	It is obligatory to pass the parameter version = '2.0' and service_id. The key parameter is not required in this case.
service_id	Required field, Service ID	121233
version	Line. Required to install a version of the API other than 1.0. If not specified, API version 1.0 is used.	2.0.

* Forced signature verification is activated by the GPS administrator in the store service settings.

For recurrent operations, additional fields must be passed, see the description of recurrent payments in the Section 6.9. of the Manual.

4.3. Getting additional parameters about a transaction (webhook notifications)

If you'd like to use recurring payments or to provide a service to the user in part (for example, to charge the internal account depending on the funds that the user has deposited), you could be interested in the opportunity to receive additional payment parameters. To do this, you need to specify the script URL when you create a service to get additional payment parameters. If the URL is specified and the checkbox is not enabled, the script will not be called.

For the Service creation see Section 2.3. of the Manual.

An example of parameters passed by an additional handler is provided in the Table 4.3.0.0.

Table 4.3.0.0. The parameters of the additional handler example

No	Parameter name	Description/ example
1	tid	Transaction ID
2	name	The name of the product or service. Displayed on the payment page.
3	comment	Payment comment passed during the payment initialization process. See Sec.4.2. for the further information
4	partner_ID	Partner ID, that is, your ID
5	service_ID	Service ID
6	order_ID	Order ID
7	type	Type of the payment
8	currency	Currency type
9	cost	The total amount of the order transferred when the payment transaction was initiated
10	income_total	The total amount paid by the buyer may differ from income and system_income only if paid in installments or when the commission is transferred to the payer
11	income	The amount received from the payment instrument under this payment transaction
12	partner_income	The amount, the store's income from this payment transaction
13	system_income	The amount paid by the buyer for this payment transaction
14	command	Current action: • command = process - called for any (including partial) payment for the service • command = cancel - a refusal from the payment channel was received, explanation of the reason in the resultStr field • command = success - called when the service is fully paid • command = recurrent_cancel - called if the cardholder canceled recurrent payments. • command = recurrent_expire - called when the recurrent expired. • command = refund - called as a result of a payment cancellation operation. In the field result = ok or fail. And resultStr contains the reason for the refusal. • command = authorize_payment - called when using double authorization for payment • command = funds_blocked - called when using two stage payment (BLOCK + CHARGE). Description in the section "Two-step payments (pre-authorization)". IMPORTANT: in case of full payment for the service, both success and process will come.
15	result	Only for command = refund, 'ok' or 'fail' values
16	resultSTR	Notification text

No	Parameter name	Description/ example
17	version	Version of the notification protocol. (Currently: 2.0). The default version is 1.0. Switching to other versions is performed on the bank side upon request from the client
18	phone_number	Optionally phone number
19	email	Optionally email
20	date_created	date and time of transaction creation, format 'YYYY-MM-DD HH24.MI.SS'
21	recurrent_order_id	Order ID (order_id), which was sent when the recurrent payment was called for the first time (only for recurring operations)
22	card	Masked card number, always filled in when there is a card field
23	cardholder	Cardholder's name if present in the transaction
24	card_binding_id	a unique token for the stored card data. See Sec. 6.10 for the further information
25	test	Value 1 (only for test payments). See Sec.6.2. for the further information
26	paid_date	date and time of payment for the transaction (confirmation of payment by the payment channel), format 'YYYY-MM-DD HH24.MI.SS'
27	check	Algorithm can be presented by the request to ALFEBA's team
28	refund_ext_id	additional refund id when making multiple refunds within a transaction. See Sec. 6.6. for the further information

All parameters are involved in the formation of the check signature. The signature is formed as md5 from all parameters concatenated into a string without spaces + the service secret key added at the end.

Example:

```
$param['check'] == md5($param['tid'].$param['name'].$param['comment'].
$param['partner_id'].$param['service_id'].$param['order_id'].
$param['type'].$param['cost'].$param['income_total'].$param['income'].
$param['partner_income'].$param['system_income'].$param['command'].
$param['phone_number'].$param['email'].$param['result'].
$param['resultStr'].$param['date_created'].$param['version'].
$secretKey);
```

For recurrent payments, two fields are added (**card** and **recurrent_order_id**) and the signature line is formed as follows:

tid + name + comment + partner_id + service_id + order_id + type + cost + income_total + income + partner_income + system_income + command + phone_number + email + resultStr + date_created + version + card + recurrent_order_id + secret_key

It should be noted that in case of full payment for the service, two calls will be generated - **success** and **process**.

- **resultStr** - notification text. It is standard for command = process and success parameter values. For command = cancel, the output is what the payment gateway will answer.
- **version** - version of the notification protocol. In the future, we plan to support the versioning of the protocol.
- **date_created** - the date the transaction was created

4.4. Transfer of extended transaction data (custom_fields, airline tickets)

“Airticket” - information about the ticket for which payment is made in case of payment by cards (long entry).

Example:

```
1  {
2    "airticket": {
3        "passengers": [
4 {
5                "ticket_number": "3905241025377",
6        "first_name": "KONSTANTIN",
7                "last_name": "IVANOV"
8            }
9        ],
10        "restricted": "0",
11        "triplegs": [
12            {
13                "date": "2014-06-06",
14                "to": "RHO",
15                "carrier": "XY",
16                "from": "ATH"
17            }
18        ]
19    }
20 }
```

Airticket parameters are provided in the Table 4.4.0.0.

Table 4.3.0.0. The air ticket parameters description

No	Parameter name	Format	Description
1	passengers		passenger list (from 1 to 4)
		ticket_number	Ticket number (up to 13 digits, including a check digit)
		first_name	Passenger name (up to 20 latin characters)
		last_name	Last name of the passenger (up to 20 Latin characters)
		passport	passport(optional) Passport series and number
		country	(optional) Nationality of the passenger (ISO 3166-1 alpha-3 code)
2	restricted		Restrictions on ticket refund (0 - no restrictions, 1 - not refundable)
3	system		(optional) Code of the system used for booking and purchasing the ticket (up to 4 Latin characters)
4	agency_code		(optional) Agency code (up to 8 Latin characters / numbers)
5	agency_name		(optional) Agency name (up to 25 Latin characters / numbers)

No	Parameter name	Format	Description
6	triplegs		route. It can contain from 1 to 4 stages. Data format of each stage
		date	Departure date (YYYY-MM-DD format)
		carrier	Carrier (two-character IATA code)
		from	Airport code of departure (three-character IATA code)
		to	Arrival airport code (three-character IATA code)
		class	(optional) Service class (1 character)
		stopover	(optional) flag of the ability to make a stop when transferring (O - stop is allowed, X - stop is not allowed)
		fare_code	(optional) Fare code (up to 6 characters)
		flight_number	(optional) Flight number (3 to 5 characters)

This page doesn’t contain any information

Chapter 5. Request authentication

This chapter contains the next sections:

Section	Description	Page
5.1.	General information	43
5.2.	Keys for the request authentication	44
5.3.	Signature creation algorithm	46
5.4.	System IP addresses	53

This page doesn’t contain any information

5.1. General information

In this chapter we describe how do you can build your own universal solution without resorting to using modules for popular CMS, using the API

To avoid fraudulent actions by third parties, you must look through the check parameter (digital signature) during the request processing from the RFI system. The digital signature provides a high level of security, as it is generated using a **secret_key** known only to your service and the processing system. In case of compromise, **secret_key** must be reported to GPS by email.

If the store does not use a script to process requests from the system, then be sure to specify **Email** when creating the service, since you must manually check the cost of the goods and the amount paid by the client (it will be indicated in the email).

Parameters to be signed must be in urldecoded format.

In addition to the parameters involved in the digital signature, the **phone_number** and **email** parameters, the user's phone number and email are sent to the handler and the successful purchase page (if you made it mandatory to enter these data in the service settings)

If the payment amount is changed and the script on the side of the store does not verify the authenticity of the request, then fraud on the part of unscrupulous users is possible.

To avoid fraudulent actions on the part of third parties, when processing a notification, you must check:

- the correspondence of the parameter **values** and **check** (digital signature) transmitted in the notification
- correspondence of the **system_income** value passed in the notification and the **cost** of the payment specified during the initialization of the payment.
- optionally, you can check other parameters passed when initiating payment, for example, **order_id**, **comment**.

‘

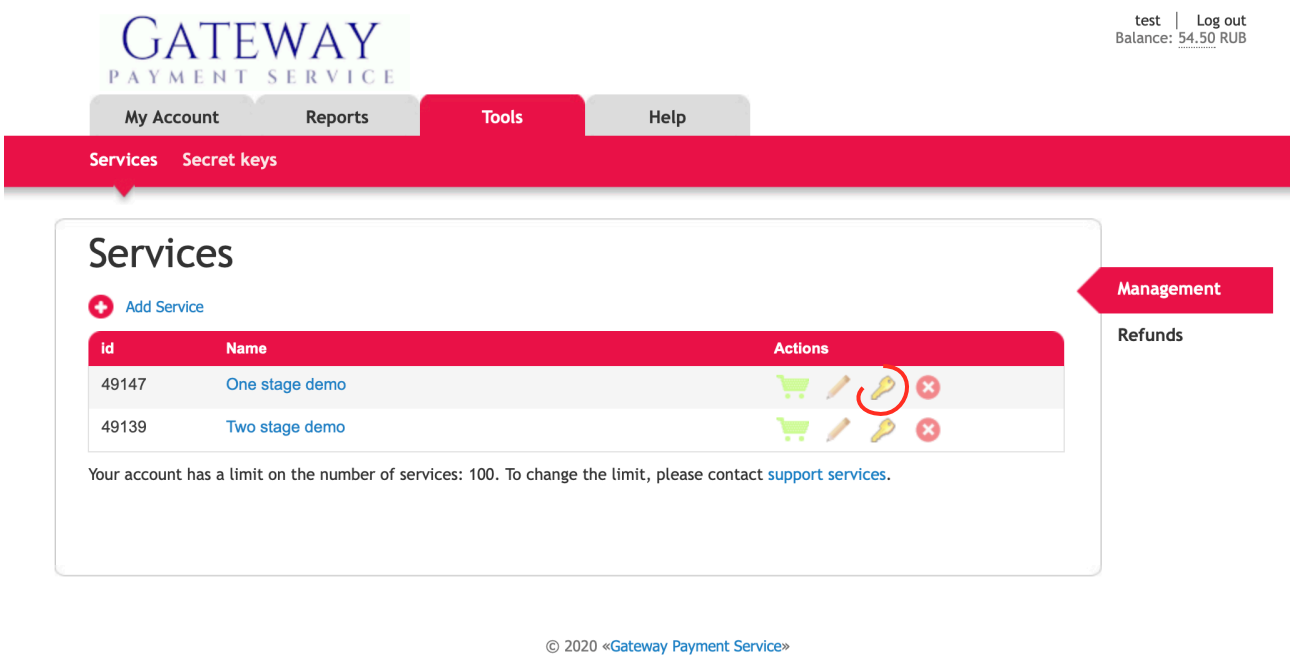
These checks ensure that the cost of the product / service is reconciled with the amount actually paid by the client. If these parameters do not match, the buyer must be denied the provision of the product / service and you should report such cases to GPS by email.

5.2. Keys for the request authentication

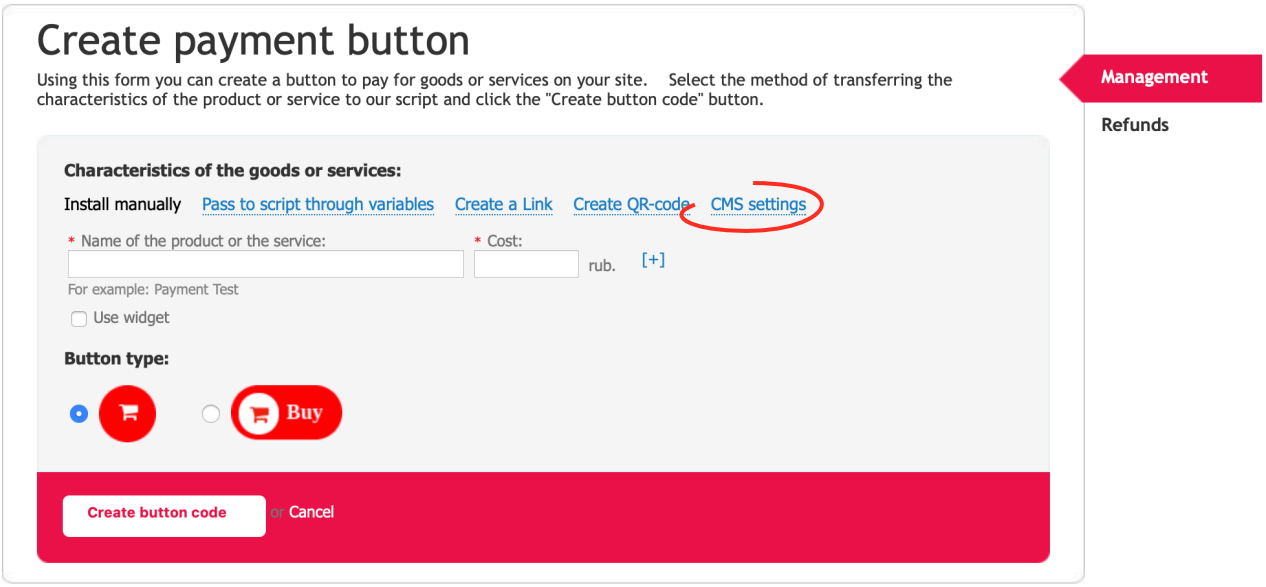
The keys to authenticate the request can be obtained by clicking on "**CMS Settings**" when creating the payment button.

Payment buttons are generated along the path **TOOLS** → **SERVICES** → **Create button**.

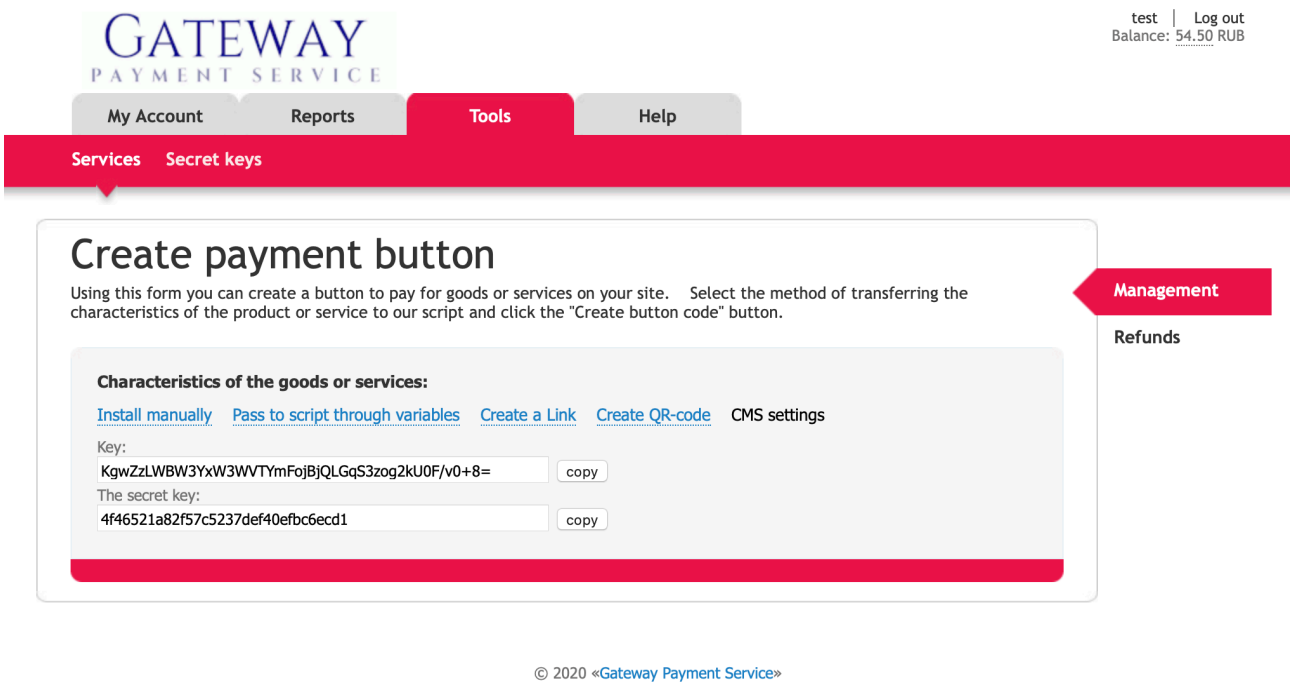
The "**secret key**" corresponds to the second version of the API, that GPS uses.



Or using **CMS settings** activating from the **Create button** page:



The page will be opened:




```
StringToSign = HTTPVerb + "\ n" +
                        ValueOfHostHeaderInLowercase + "\ n" +
                        HTTPRequestURI + "\ n" +
                        CanonicalizedQueryString

     # HTTPVerb is a POST, GET, PUT or DELETE method.
     # ValueOfHostHeaderInLowercase - host parameter from the HTTP request header.
     # HTTPRequestURI - URI component, absolute path to, but not including GET parameters.
     # A '/' is expected for an empty path.
     # CanonicalizedQueryString is the string from the previous step.
```

5.3. Signature creation algorithm

Create a normalized query string for use in the following stages:

A. Sort parameters by name in utf8, comparing byte-wise. Parameters are taken from the GET URI or from the body of the POST request (when ContentType is application / xwwwformurlencoded)

B. URL-encode parameter names and values according to the following rules:

C. Do not encode non-reserved characters defined in RFC 3986. These characters are Az, az, 09, minus (-), underscore (_), period (.), And tilde (~).

D. All other characters must be encoded as% XY, where X and Y are hexadecimal characters 0 through 9 and A through F (all caps). Extended utf8 characters are encoded as% XY% ZA ...

E. Space is encoded as% 20 (and not as +, as is usually done in URLs)

F. The coded parameter names are separated from the coded values by an equal sign (=, ASCII code 61), even if the parameter is empty.

G. Name-value pairs are separated by an ampersand (&, ASCII code 38).

H. Create a signature string according to the following pseudo - grammar (where "\ n" is ASCII line feed characters):

I. Calculate RFC 2104 compliant HMAC from the newly generated StringToSign, using the peer's secret key as the algorithm key, and SHA256 as the hashing method.

J. Convert the received signature to base64.

K. Use the result as the value of the check parameter.

Example of the creation using PHP:

```
function http_build_query_rfc_3986($queryData, $argSeparator='&')
    {
        $r = '';
```

```
4             $queryData = (array) $queryData;
5             if(!empty($queryData))
6             {
7                 foreach($queryData as $k=>$queryVar)
8                 {
9                     $r .= $argSeparator.$k.'='.rawurlencode($queryVar);
10                }
11            }
12            return trim($r,$argSeparator);
13        }
14
15        function sign($method, $url, $params, $secretKey, $skipPort=False)
16        {
17            ksort($params, SORT_LOCALE_STRING);
18
19            $urlParsed = parse_url($url);
20            $path = $urlParsed['path'];
21            $host = isset($urlParsed['host'])? $urlParsed['host']: "";
22            if (isset($urlParsed['port']) && $urlParsed['port'] != 80) {
23                if (!$skipPort) {
24                    $host .= ":{$urlParsed['port']}";
25                }
26            }
27
28            $method = strtoupper($method) == 'POST'? 'POST': 'GET';
29
30            $data = implode("\n",
31                array(
32                    $method,
33                    $host,
34                    $path,
35                    http_build_query_rfc_3986($params)
```

```
36                  )
37              );
38
39              $signature = base64_encode(
40                  hash_hmac("sha256",
41                      "{$data}",
42                      "{$secretKey}",
43                      TRUE
44                  )
45              );
46
47              return $signature;
48          }
49
50      // An example of a call, where $ req is an array with transaction parameters:
51      sign("GET", "https://partner.rficb.ru/alba/input/", $req, 'secret');
```

Example of the creation using python:

```
from urllib.parse import urlparse,quote
import hashlib
import urllib
import base64
import hmac


def sign(method, url, params, secret_key, exclude=['check', 'mac']):
    """
    Типовой метод для подписи HTTP запросов
    """
    url_parsed = urlparse(url)
    keys = [param for param in params if param not in exclude]
    keys.sort()

    result = []
    for key in keys:
        value = quote(
            (params.get(key) or '').encode('utf-8'),
            safe='~'
        )
        result.append('{}={}'.format(key, value))

    data = "\n".join([
        method,
        url_parsed.hostname,
        url_parsed.path,
        "&".join(result)
    ])
```

```
30        secrkey = secret_key.encode('utf-8')
31        mesg=data.encode('utf-8')
32        print(secrkey,mesg)
33        digest = hmac.new(
34            secrkey,
35            mesg,
36            hashlib.sha256
37        ).digest()
38        signature = base64.b64encode(digest)
39        print(signature.decode('utf-8'))
40        return signature
41
42    # Example of the call
43    sign("GET", "https://partner.rficb.ru/alba/input/", {'login': 'newlogin~_-.'},
      '165165165sd');
```

Example of the creation using java:

```
package ru.rficb.alba;

import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.nio.charset.Charset;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Map;
import java.net.URI;
import java.net.URISyntaxException;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DatatypeConverter;

/**
* Signature of the version 2.0+
*/
public class AlbaSigner {

    public static String sign(String method, String url, Map<String, String>
params, String secretKey)
            throws URISyntaxException, UnsupportedEncodingException,
NoSuchAlgorithmException, InvalidKeyException {

        URI uri = new URI(url);

        List keys = new ArrayList<>(params.keySet());
```

```
29            Collections.sort(keys);

30

31            StringBuilder sb = new StringBuilder();

32            for (String key: keys) {

33                if (sb.length() > 0) {

34                    sb.append("&");

35                }

36

37                sb.append(String.format("%s=%s", key,
    URLEncoder.encode(params.get(key), "UTF-8")));

38            }

39            String urlParameters = sb.toString();

40            String data = method.toUpperCase() + "\n" +

41                    uri.getHost() + "\n" +

42                    uri.getPath() + "\n" +

43                    urlParameters;

44

45            Mac hmacInstance = Mac.getInstance("HmacSHA256");

46            Charset charSet = Charset.forName("UTF-8");

47            SecretKeySpec keySpec = new
    javax.crypto.spec.SecretKeySpec(charSet.encode(secretKey).array(),
    "HmacSHA256");

48            hmacInstance.init(keySpec);

49

50            return
    DatatypeConverter.printBase64Binary(hmacInstance.doFinal(data.getBytes("UTF-8"))
    );

51        }

52    }
```

5.4. System IP addresses

From the IP addresses specified in the Table 5.4.0.0., requests will be sent to your handler.

Table 5.4.0.0. IP addresses details for the requests

IP address

This page doesn’t contain any information

Chapter 6. Additional options

This chapter contains the next sections:

Section	Description	Page
6.1.	General information	57
6.2.	Background initiation of payment	57
6.3.	Requesting information about a transaction	66
6.4.	Pre-authorization (additional request about the possibility of making a payment)	69
6.5.	Two-stage payments (holding funds on a card)	70
6.6.	Refund of payments (refund)	71
6.7.	API for sending and receiving payment / refund checks	77
6.8.	Payment cancellation (reversal)	78
6.9.	Description of the recurring payment scheme (RP)	79
6.10.	Saving data of the payer's card	83
6.11.	Using tokens (card tokens) with libraries for mobile applications	84

This page doesn’t contain any information

6.1. General information

In this chapter we describe options for deeper integration with the Bank's services. Most of the features are included in the SDK (php, python, java).

6.2. Background initiation of payment

The difference from the usual scheme of work is that the payment is initiated by calls to API methods without displaying additional WEB pages. For this, an additional background parameter is passed with the value “1”. At the moment, according to the described scheme, it is possible to pay with card tokens, through QIWI and mobile payment for Beeline, MTS, Megafon and TELE2 operators for example, in the Russian Federation.

6.2.1. Getting a list of payment methods available for the service

To get a list of available methods that can be initiated in the background, you need to send a GET request to: https://partner.gps.com/alba/pay_types with the following parameters:

- **service_id** - the id of the service by which you need to get a list of available payment methods
- **version** - "2.0"
- **check** - Electronic signature of the request (See Section 5.3. for the further information).

The service response comes in the form of a JSON string.

- Error response: {‘status’: ‘error’, ‘code’ = ’auth’, ‘message’: ‘Invalid signature’}
- Positive answer: {‘status’: ‘success’, ‘types’: [‘mc’, ‘qiwi’, ‘mtsmoney’, ‘mtsmoney_test ']}

where **types** are payment methods available for this service.

Example of the creation using php:

```
$service = new AlbaService(SERVICE_ID, 'SERVICE_SECRET');

    try {
        $payTypes = $service->payTypes();
        // $payTypes - array of accepted payment methods
    } catch (AlbaException $e) {
        echo $e->getMessage();
    }
```

Example of the creation using python:

```
from alba_client import AlbaService, AlbaException

    service = AlbaService(SERVICE_ID, 'SERVICE_SECRET')
    try:
        pay_types = service.pay_types()
        # pay_types - list of accepted payment methods
    except AlbaException, e:
        print("An error has occurred: {}".format(e))
```

Example of the creation using Java:

```
AlbaService service = new AlbaService (SERVICE_ID, "SERVICE_SECRET");

  Set <String> paymentTypes = service.paymentTypes ();
```

6.2.2. init_payment

Payment is initiated by the method of sending the form from the button code (see Section 4.2.) plus the background = 1 parameter and parameters for authentication. POST request must be sent to: https://partner.gps.com/alba/input/

- cost - the cost of the product / service
- name - payment name
- email - the client's email (for mobile commerce, this parameter is optional)
- order_id - unique order number or 0
- phone_number - customer's phone number
- background - always 1
- type - payment method ('mc', 'qiwi', 'mtsmoney', 'mtsmoney_test'), must be available.
- invoice_data - Data in json format for a fiscal receipt (not used in the current version)
- test - an optional parameter for testing payment from a phone or wallet account; to test payment by cards, use test sets of cards (see Sec. 3.8.). Can be ok or operator_cancel. If the parameter is present, no real calls to providers are made - a successful payment (for ok) or a refusal of the provider (for operator_cancel) is immediately generated. The created transaction is test and no payments are made on it.

One of the following parameter sets must be passed for authentication:

- key - a unique key for the service, generated together with the button,

or

- service_id - service id
- version - "2.0"
- check - Electronic signature of the request. (See Sec 5.3.)

6.2.3. init_result

Initialization result means creation of a payment transaction. Payment for the transaction is delayed in time, the result of payment must be received by processing notifications (abbreviated notification, extended notification) from the GPS system or form a request to obtain the status of the transaction.

For operations of repeated write-off by recurrent, the operation is performed synchronously. A positive answer guarantees a successful operation, while it is recommended in any case to process notifications sent from the GPS system.

A positive response

- {‘Status’: ‘success’, ‘tid’: <TID>, 'help': "<text that describes the payment procedure via mobile commerce or terminal>"}
- {‘Status’: ‘success’, ‘tid’: <TID>}

Error response

- {‘Status’: ‘error’, ‘code’: ‘<type | auth | data | common | unique>’, ‘msg’: ”}
- Where message is a textual description of the error, and code is a type:
- type - this type of payment is not available
- auth - invalid signature
- data - data is incorrectly formed (phone format, email)
- common - other errors
- unique - a transaction with this order_id already exists

Positive answer for mc gateway:

```
1 {'status': 'success', 'tid': 12332, 'help': 'text that describes the payment procedure via mobile commerce'}
```

Positive answer for spg gateway (recurring card payment):

```
1 {"status": "success", "tid": "36619984"}
```

An example of a negative answer for spg gateway (recurring card payment):

```
1 {"status": "error", "msg": "Recurrent for service XXXXX / order YYYYYY has status 'canceled'", "code": "common"}
```

Example of the creation using php:

```
1     $ service = new AlbaService (SERVICE_ID, 'SERVICE_SECRET');
2
3         try {
4             $ result = $ service-> initPayment (
5                 'mc', // payment method
6                 10, // amount
7                 'Test', // payment name
8                 'test@example.com', // client email
9                 '71111111111' // customer phone number
10                False, // order_id (optional)
11                'partner', // commission (optional)
12            );
13            if ("success" === $ result-> status) {
14                echo "Successful initiation of transaction id =". $ result-
      > tid;
15            } else {
16                echo $ result-> message;
17            }
18        } catch (AlbaException $ e) {
19            echo $ e-> getMessage ();
20        }
```

Example of the creation using python:

```
from alba_client import AlbaService, AlbaException

    service = AlbaService (SERVICE_ID, 'SERVICE_SECRET')
    try:
        result = service.init_payment (
            pay_type = 'mc',
            cost = 10,
            name = 'Test',
            email='test@example.com ',
            phone = '71111111111',
            commission = 'partner'
        )
        if result ['status'] == "success":
            print ("Successful initiation of transaction {}". format (result ['tid']))
        else:
            print (result ['message'])
    except AlbaException, e:
         print ("An error occured: {}". format (e))
```

Example of the creation using Java:

```
AlbaService service = new AlbaService ("SERVICE_KEY");

    InitPaymentRequest request = new InitPaymentRequest ()
            .builder ()
            .setPaymentType ("mc")
            .setCost (new BigDecimal (10.5))
            .setName ("Test")
            .setEmail ("main@example.com")
            .setPhone ("71111111111")
            .build ();

    InitPaymentResponse response = service.initPayment (request)
```

6.2.4. tr_status

After the transaction is created, the partner has the opportunity to receive information on the transaction by sending a POST request to https://partner.gps.com/alba/details The request must be sent (protocol version 2.0):

- version - "2.0"
- tid - transaction id
- check - Electronic signature of the request.

Alternatively, you can send a GET request to the same address with parameters (DEPRECATED):

- api_key - authorization key (https://home.gps.com/apikeys/)
- tid - transaction id

Error response:

```
1 {'status': 'error', 'code': '<auth | common | method>', 'msg': ''}
```

Where msg is a textual description of the error and code is the type:

- auth - invalid signature
- common - other errors

A positive response:

```
1 {"status": "success", "order_id": "<order_id>", "partner_income": <store income>, "help": "<text that describes the payment procedure via mobile commerce or terminal>", "service ":" <service name> "," transaction_status ": '<open | error | payed | success>'," tid ":" <TID> "," service_id ":" <service id> "," income_total ": <transaction amount>}
```

Example of the creation using php:

```
1     $ service = new AlbaService (SERVICE_ID, SERVICE_SECRET);
2
3         try {
4             $ details = $ service-> transactionDetails (TRANSACTION_ID);
5             echo 'Transaction status:'. $ details-> transaction_status;
6         } catch (AlbaException $ e) {
7             echo $ e-> getMessage ();
8         }
9
```

Example of the creation using python:

```
from alba_client import AlbaService, AlbaException

    service = AlbaService (SERVICE_ID, 'SERVICE_SECRET')
    try:
        details = service.transaction_details (TRANSACTION_ID)
        print ('Transaction status: {}'. format (details ['transaction_status']))
    except AlbaException, e:
        print ("An error occured: {}". format (e))
```

Example of the creation using Java:

```
 // response - response received when the payment was initiated
 TransactionDetails details = alba.transactionDetails(response.getSessionKey());
```

6.3. Requesting information about a transaction

Transaction information

After the transaction is created, the partner has the opportunity to receive information on the transaction by sending a POST request to https://partner.gps.com/alba/details The request must be sent (protocol version 2.0):

Request information on order_id and service_id:

- version - "2.0"
- order_id - order id
- service_id - the service ID can be found in the section of your personal account Tools → Services
- check - Electronic signature of the request.

Request information by transaction_id:

- version - "2.0"
- tid - transaction id
- check - Electronic signature of the request.

Error: {"status": "error", "msg": "Error description", "code": "<auth | common>"}

Transaction not found: {"status": "pending", "tid": "None", "msg": ""}

Transaction found: {"status": "success", "order_id": "123456", "partner_income": 193.45, "help": false, "service": "Service name", "transaction_status": "<open | payed | success | error> "," tid ":" 12312312 "," service_id ":" 12345 "," income_total ": 199.0}

- partner_income - the amount in rubles, the store's income from this payment transaction
- income_total - the amount in rubles paid by the buyer for this payment transaction

transaction statuses:

- open - transaction found, in the process of payment
- error - erroneous transaction, final status
- payed or success - transaction is paid, final status
- pending - the transaction was not found, not created, or is being processed. It is necessary to re-request.

Example 1:

Request

Headers

Content-Type:application/x-www-form-urlencoded

URL

https://partner.rgps.com/alba/details

Body

version:2.0

service_id:67279

order_id:12456

check:7ALsKM72UNEwmnlLBd8IAYUDZV67NnFdUzSIo61KFmo=

Response

{"status": "success", "order_id": "12456", "partner_income": 161.5, "help": false, "service": "\u0422\u043e\u043a\u0435\u043d \u0438 direct_api", "transaction_status": "payed", "tid": "67740380", "service_id": "67279", "income_total": 190.0}

Example 2:

Request

Headers

Content-Type:application/x-www-form-urlencoded

URL

https://partner.gps.com/alba/details

Body

version:2.0

tid:377333230

check:954h3gXzciI8Jnx7ux6dELS6cnurgr6Jtz7f35V9vkM=

Response

{"status": "success", "order_id": "None", "partner_income": 0.85, "help": false, "service":

"\u0442\u0435\u0441\u0442\u043e\u0432\u044b\u0439 \u0441\u0435\u0440\u0432\u0438\u0441", "transaction_status": "payed",

"tid": "377333230", "service_id": "82920", "income_total": 1.0}

6.4. Pre-authorization (additional request about the possibility of making a payment)

Authorize payments

This procedure will allow you to receive an additional request immediately before sending the client to our payment gateway using cards, or to the gateway of other payment systems (e-wallets, mobile payment). To this request, you answer whether you are ready to accept this payment. To use this procedure, specify “**Receive additional payment parameters**” in the service settings, specify the address of your additional parameters handler, specify “**Confirmation of goods reservation**” (if you select “Once”, then the request will come only once)

Then, when paying, you will receive `command = authorize_payment`

The response is received in the format "`status: STATUS \ n \ nMESSAGE`".

- STATUS can take the value "authorized" - payment is allowed. Other values - prohibition of payment;
- MESSAGE - a message that will be shown to the client if the payment is refused;

UTF-8 encoding, request timeout - 10 seconds.

An empty or incorrectly formed response is equated to a payment rejection response.

6.5. Two-stage payments (holding funds on a card)

In some cases, it is required to block funds on the payer's card until the transaction is confirmed by the merchant. In such cases, you need to use the two-step payment functionality.

This functionality is activated for the selected service by separate agreement. According to the rules of payment systems, the transaction must be issued for payment within 7 days, so funds can be in the "For approval" status for a maximum of 7 days - after this period the funds automatically become available to the payer.

During the payment, the funds are blocked on the client's card, and the payment is debited or canceled by the merchant employee.

Upon blocking funds, the system initiates notification `command = funds_blocked`, described in the section 4.4. Payments that require confirmation are reflected in the Merchant profile.

During processing, the merchant manager can choose the following actions:

- cancel blocking
- confirm cancellation of full
- confirm writing off a smaller amount

6.5.1. Reserved Funds Management API

To create an application via the API, you must perform a POST request to the address https://partner.gps.com/alba/process_funds_blocked/

The request must contain (signature version 2.0):

- version - "2.0"
- check - Electronic signature of the request.
- tid - transaction id
- action - type of operation: charge - debiting funds, unblock - unblocking funds;
- amount - the amount to be debited;

Error response: `{'status': 'error', 'msg': ''}`

Positive answer: `{'status': 'success'}`

If, with `action = charge`, you specify an incomplete transaction amount, only the specified amount will be debited. And the rest will be unlocked.

6.6. Refund of payments (refund)

The system assumes the possibility of providing a refund for successful transactions. Refunds can be made from the Merchant profile or via the API.

The functionality is activated for the selected service and payment channel by separate agreement. Refunds may not be available for certain channels.

If the payment channel does not support returns, the message “The payment channel does not support returns” will be displayed

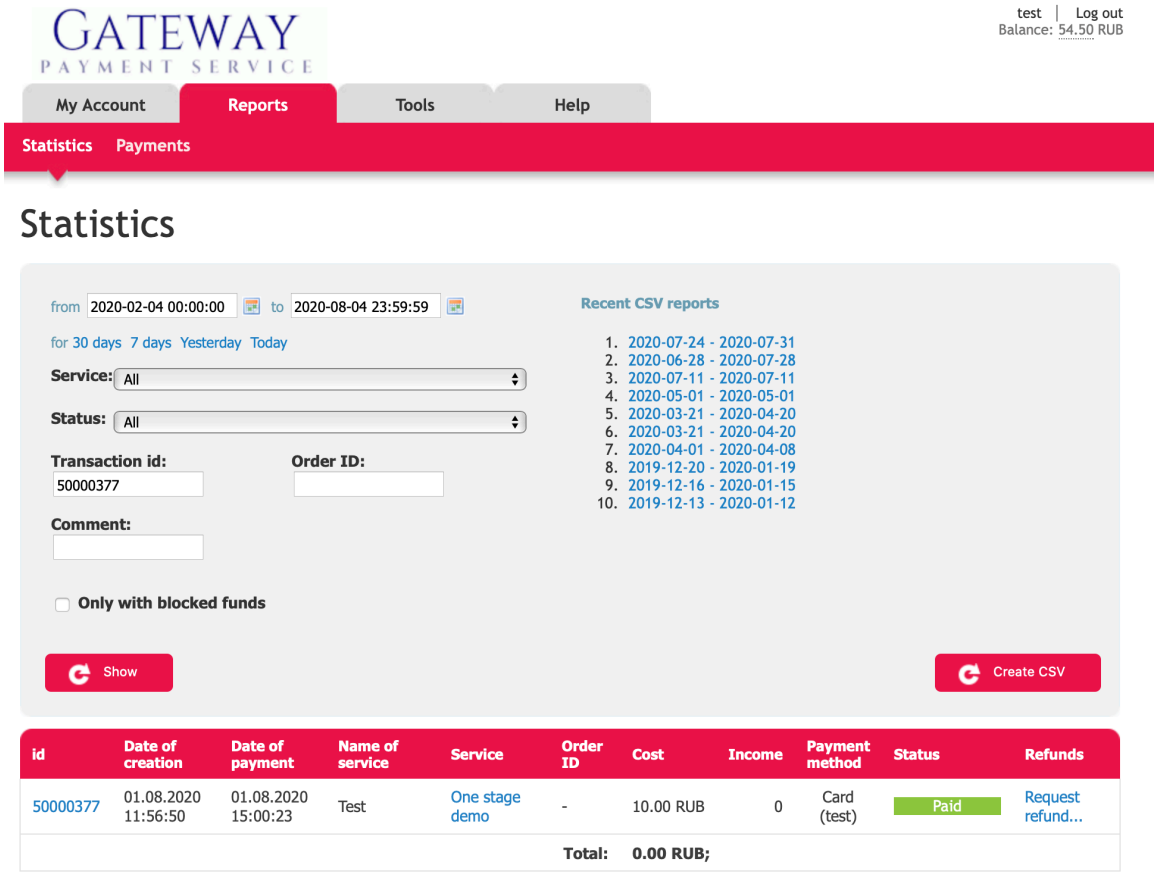
For **Visa / MasterCard** refunds:

- spent day-to-day passes without additional commission and with minimal delay. Funds are credited to the payer's account within 2 hours.
- spent the next day are subject to a commission and the term for crediting funds can be up to 7 working days, depending on the issuer.

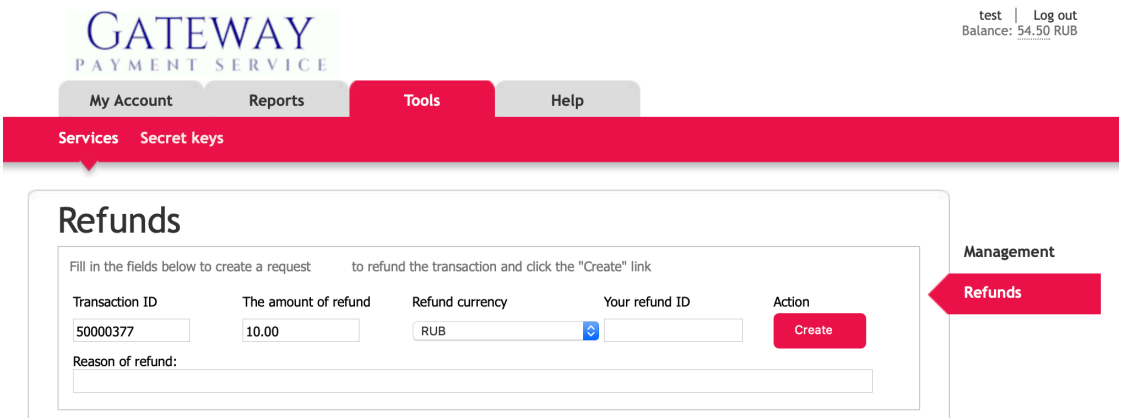
The return operation is irreversible! If you made a mistake with the refund amount or transaction id, contact the technical support of the GPS.

6.6.1. Refund through the Merchant profile

Go to **Merchant profile** → **Reports** → **Statistics**. Find the required transaction and click on "**Request Refund**":



There will be an automatic redirection to the returns management page, where you must specify the amount, reason and id of the return (optional):



If the return id is not specified, the system will assign the value automatically.

Multiple refund

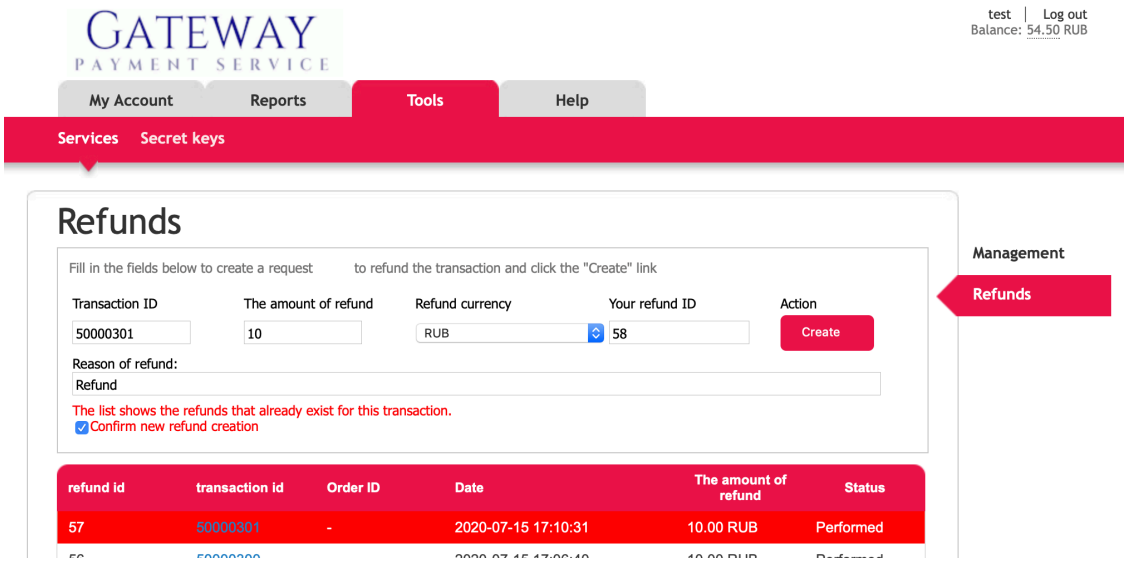
For Visa / MasterCard it is possible to make several refunds within one transaction. This is possible if one refund was made for an amount less than the original one.

Initialization takes place from the page for managing returns **Merchant profile** → **Tools** → **Services** → **Refunds** (to the right of the main window).

Fill in the fields:

- **ID** of the Transaction within which refunds will be initiated.
- **Refund amount** minus previous / previous refunds.
- **Your return ID**. If there is only one return, it is not necessary to fill out.

After entering the data and confirming, an informational message and the previously created return will appear:



Click on the checkbox and click Create.

6.6.2. Statuses

The refund operation can have several statuses:

- In progress - the operation has been created in the system and is in progress;
- Queued - the operation has been created in the system and is in the queue for execution;
- Canceled - the operation is not completed, an error occurred, contact technical support;
- Completed - the operation was completed, the refund was made. The refund check is available in the statistics of transactions by click → transaction ID → show check.

Refunds may exceed the total amount of transactions per day. In this case, the refund is in the queue until the required amount is accumulated (before it is credited to the current account). After each transfer to the current account, the amount will be accumulated again. The GPS/Bank cannot collect funds for a refund from the current account, therefore, only the amount that has accumulated during the day is involved in refunds.

6.6.3. Refund using API

To create an application via the API, you must perform a POST request to the address https://partner.gps.com/alba/refund/

The request must pass (Content-Type: “application / x-www-form-urlencoded”):

- **version** - "2.0"
- **tid** - transaction id
- **amount** - an optional parameter, the amount is specified in the xxx.xx format. If the value is not specified, the entire amount will be returned
- **reason** - optional parameter - reason for return
- **refund_ext_id** - refund id
- **check** - Electronic signature of the request.

JSON is returned in response with the result of processing the request:

- Success: `{‘status’: ‘success’, ‘payback_id’:}`
- Failure: `{‘status’: ‘error’, ‘message’: ”}`

The result determines the creation / initialization of a refund in the system, but not the refund process itself. Information on the progress of the operation should be viewed on the Refunds management page or in the statistics.

Example of the creation using php:

```
$ service = new AlbaService (SERVICE_ID, 'SERVICE_SECRET');

try {
        $ result = $ service-> refund (TRANSACTION_ID);
        echo "Refunds are queued, payback_id:". $ result-> payback_id;
    } catch (AlbaException $ e) {
        echo "An error occured:". $ e-> getMessage ();
    }

```

Example of the creation using python:

```
from alba_client import AlbaService, AlbaException

    service = AlbaService (SERVICE_ID, 'SERVICE_SECRET')
    try:
        result = service.refund (TRANSACTION_ID)
        print ("Return queued, payback_id: {}". format (result
['payback_id']))
    except AlbaException, e:
        print ("An error occured: {}". format (e))

```

Example of the creation using Java:

```
AlbaService = new AlbaService(SERVICE_ID, "SERVICE_SECRET");

    RefundResponse refundResponse = service.refund(
            RefundRequest.builder(response.getTransactionId())
                    .setAmount(new BigDecimal("REFUND_AMOUNT"))
                    .setReason("REFUND_REASON")
                    .build()
    );

```

6.7. API for sending and receiving payment / refund checks

API call to send a check to the email specified during payment:

```
https: //partner.gps.com/alba/send_receipt/? tid = & api_key = ... - for direct check
```

```
https://partner.gps.com/alba/send_receipt/?tid=&api_key=...&receipt_type=payback - for return
```

API call to get a link to a receipt:

```
https: //partner.gps.com/alba/receipt/? tid = & api_key = ...
```

```
https://partner.gps.com/alba/receipt/?tid=&api_key=...&receipt_type=payback
```

6.8. Payment cancellation (reversal)

To cancel a payment, you must complete a POST request to the address `https://partner.gps.com/alba/reversal/<tid>/<payment_type>/`

The url must be passed:

- **tid** - transaction ID
- **payment_type** - payment type, string. Mtsmoney and mtsmoney_ext are available

In the body of the request or in the parameters of the request, you must pass check - electronic signature.

A response containing an HTTP Status equal to 200 means successful processing of the request.

6.9. Description of the recurring payment scheme (RP)

This option is available only for merchants that have agreed to provide recurring payments when connecting to the system. When connecting the RP, you need to prepare your system for processing the messages described in the section 4.4.

6.9.1. Registration of recurring payment

Recurrent payment (RP) consists of two operations:

- payment from RP registrations recurrent_type = first
- RP on demand recurrent_type = next

To register a recurring payment, a regular request is sent to the address `https://partner.gps.com/alba/input/` described in Sections 3.2. and 4.2. plus additional parameters:

Table 6.9.1.0. The additional parameters description for the RP registration

Parameter name	Value	Description/ Examples
order_id	Order ID, length 6-20 characters.	123456 (minimum 6 characters) If the order_ID exists: There is a mechanism for checking the order number (order_id) for uniqueness. It's not a mistake. The warning you see in such cases is the result of this check. A link or button is a generator. Each transition through it is an attempt to create a new transaction with the number that the link is currently transmitting. Transferring the same order number from the store admin (CMS) leads to the same result. An example of a link containing an order number: https://partner.gps.com/alba/input/?name=Rent_hall&cost=15200&key=Agl%2FskVOgsU6ZizcKvXjIlWhNJPyYri9x0J%2BY9ex6C0%3D&default_email=&order_id=192032 An example of an algorithm for receiving a warning: The first click created transaction 3950000001 with order number 192032. Payment failed, the payer closed the browser. After a while, the payer clicked on this link again. The bank gateway tried to create transaction 3950000002 with order number 192032. Warning. The existence of two different transactions with the same number is impossible! If you want to avoid this behavior, pass a new order number for each click on the button from the cart or link. If you continue to receive the warning "There is already an order with order_id XXXXX. Old number XXXXXXXXX", then your store sends the same message for every click. You can verify the existence of a transaction with the "old" number indicated in the warning by checking the "Reports" section.
payment_id	the type of payment to which the payer should be sent	spa spg_test for tests
recurrent_type	An indication that the payment is recurring if the value is: "First" - payment and registration of the RP; "Next" - next RP; all other parameter values - the payment is not an RP.	If value = "first" then fields are required: email, order_id, recurrent_comment, recurrent_url, recurrent_period If value = "next" then the required fields are: background = 1, order_id, recurrent_order_id

Parameter name	Value	Description/ Examples
recurrent_comment	Text description for what the registration of the RP is made	Text field. Transmitted optionally.
recurrent_url	Link to a detailed description of the rules for providing a recurring payment	
recurrent_period	The period after which the next write-off occurs	Currently only valid value is: byrequest Must be passed only if recurrent_type = first
check	Version 2.0 signature - electronic signature of the request	It is obligatory to pass the parameter version = '2.0' and service_id. The key parameter is not required in this case.
service_id	Service ID, required	121233
version	Line. Required to install the API version	2.0

After receiving the correct request, the GPS initiates the first recurring payment.

Based on the results of the payment, the GPS redirects the user to the partner and informs the payer to the specified e-mail with a link to unsubscribe (deactivate) the RP.

The GPS informs the partner through asynchronous notification about the successful activation of the RP and payment of the first transaction. In addition to the extended notification command = success, the following fields are transmitted: recurrent_order_id and card with a masked card number. See Sec.4.4.

If the payer refused to save the card in the notifications, the recurrent_order_id parameter will be absent.

6.9.2. Making the second and subsequent payments to the RP (synchronous operation)

The partner forms a request to `https://partner.gps.com/alba/input/` indicating the main set of parameters plus:

Table 6.9.2.0. The additional parameters description for the 2-more RP payments

Parameter name	Value	Description/ Examples
payment_type	Payment type	spa spg_test for tests
recurrent_type	“Next”, a fixed value for repeated RPs.	“next”
recurrent_order_id	Reference to the order_id of the first RP, it is necessary to transfer the order_id specified during the registration of the first RP. Length 6-20 characters. Specified only for recurrent_type = next	123456 (minimum 6 characters)
background	Parameter indicating that the request is running in the background	1
check	Version 2.0 signature - electronic signature of the request	It is obligatory to pass the parameter version = '2.0' and service_id. The key parameter is not required in this case.
service_id	Service ID, required	121233
version	Line. Required to install the API version	2.0

The payer or the merchant system is also not sent to the card data entry form, but waits for the results of the payment operation to be received through the event handler. The result of the payment is returned to the partner as standard. In addition to the standard notification, the following fields are transmitted: recurrent_order_id and card with a masked card number. See Sec 4.4.

For the description of the result of operations with background = 1 see Sec. 6.2.

6.9.3. Cancellation of recurring payment (synchronous operation)

To terminate a recurring payment, you must send a request with the parameters specified below to the URL: https://partner.gps.com/alba/recurrent_change/

Table 6.9.3.0. The additional parameters description for the RP cancellation

Parameter name	Value	Description/ Examples
operation	The action on the recurrent must be “cancel”	
order_id	order_id passed during RP registration	
service_id	Service ID, required	121233
check	Version 2.0 signature - electronic signature of the request	It is obligatory to pass the parameter version = '2.0' and service_id. The key parameter is not required in this case.
version	Line. Required to install the API version	2.0

6.9.4. Notification of termination of the recurring payment

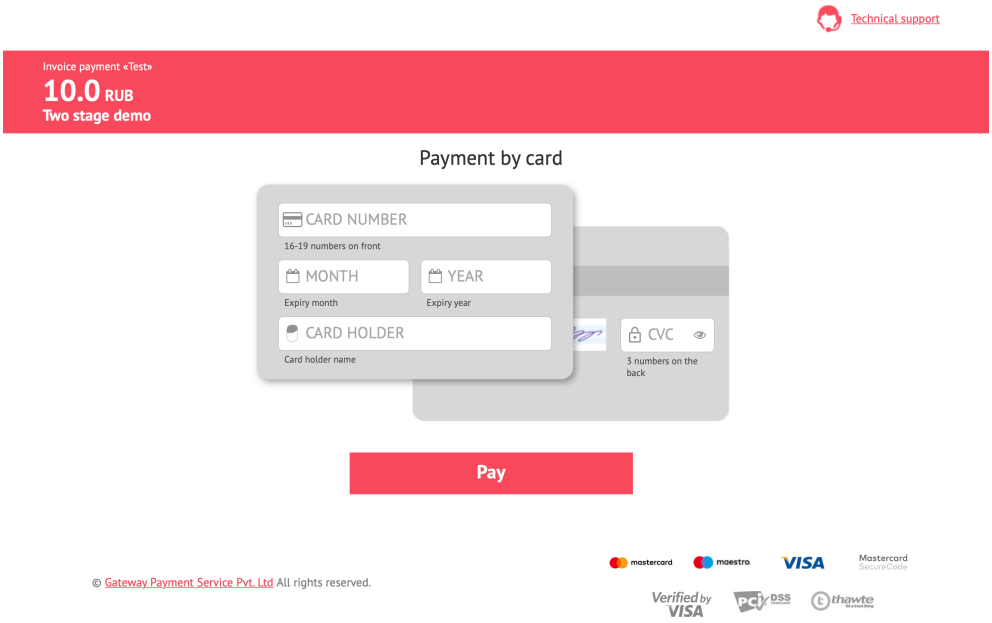
If the cardholder cancels the subscription to recurrent payments, a notification **`command = recurrent_cancel`** is generated.

If the card has expired, the card binding is canceled and a notification is generated with the value **`command = recurrent_expire`**.

See the complete list of extended notifications in the Sec 4.4.

6.10. Saving data of the payer's card

This functionality allows you to save the payer's card data (except for CVC / CVV) and automatically pull them into the card data entry form. The payer will then only need to indicate the CVC / CVV code and pass 3DS authorization to make the payment.



The functionality is connected upon request from the support service.

After enabling this option, the **`card_binding_id`** parameter with a unique token corresponding to the card data will be sent to the additional handler in notifications (see Sec. 4.4.), which is then used, among other parameters, in the request to initiate a payment to substitute card data corresponding to this token.

The fields for filling / changing the card data become unavailable.

6.11. Using tokens (card tokens) with libraries for mobile applications

This functionality is connected upon request to the GPS technical integration service.

Token creation

You need to send a request to the URL:

`https://test.gps.com/cardtoken/create` - test

`https://secure.gps.com/cardtoken/create` - production

Token lifespan is 15 minutes. Each generated token can be used no more than once.

POST request with parameters:

- service_id - service ID
- card - Card number. Numbers without spaces.
- exp_month - The month the card expires. 2 digits with zero
- ahead for months 1 to 9 ('01' ... '09').
- exp_year Card expiration year (2 digits 20XX).
- cvc - Card verification code (CVV2 / CVC2) - 3 digits.
- card_holder (optional) up to 30 characters Only in Latin and capital letters.

Response (JSON-encoded)

Successful token creation: `{“status”: “success”, “token”: “[Token]”}`

Error: `{“status”: “error”, “message”: “[General error description]”}`

In case of an error, there may still be an additional element “errors” - a dictionary, where the keys are the names of the fields, and the values are the description of errors in them.

Payment using a card (/ alba / input)

1. The card token can be used to initiate both regular and recurring payments;

2. Using a token is allowed only in background mode;

```
https://partner.gps.com/alba/
input/?...payment_type=spg_test&background=1&card_token=3dddfa33494c6bfdf4d5230f
2286d44bab8538e13ad18d61cb336c73c5b60f94
```

3. If the payment is not successful, an error will be returned (see the description of responses for background payments);

4. If the card does not require 3DS authorization and the payment is successful, the alba transaction ID will be returned (see the description of responses for background payments);

5. If the card requires 3DS authorization, then the response, along with the transaction ID alba, will return a 3ds dictionary with the data required for 3DS authorization

3DS authorization

If there is a dictionary with the name 3ds in the background payment initiation response, the user should be redirected to the website of the issuing bank to pass 3DS authorization. Part of the data to be sent is in the 3ds dictionary, and the partner must form part of it himself;

1. It should be sent by POST request to the URL specified in the ACSUrl field of the 3ds dictionary;

2. Values for reference:

- PaReq - from the PaReq field of the 3ds dictionary;
- MD - from the MD field of the 3ds dictionary;
- TermUrl - URL of the handler located on the partner site. The user will be returned to it after passing the 3DS authorization on the website of the card issuing bank. This URL must be formed so that information about the transaction is transmitted in it: it is recommended to transmit service_id, tid and order_id (if the partner has formed a transaction with it);

3. Handler of 3DS authorization result (TermUrl)

- This handler should call the API / alba / ack3ds / method to check if the 3DS is authorized by the user;
- In GET parameters, it will receive the previously generated information about the transaction (service_id, tid, order_id);
- In POST parameters, it will receive information from the issuing bank - fields PaRes and MD;

- To check the passage of 3DS authorization, call POST with an API request `https://partner.rficb.ru/alba/ack3ds/`, passing there:
    - service_id;
    - tid or order_id;
    - emitent_response - data received from the issuing bank in the form of a JSON-encoded dictionary;
    - Note: Request authorization: signature version 2.0 or via api_key;
- The result of the API method / alba / ack3ds /:
    - If unsuccessful, a JSON response will be returned:
    - {"status": "error", "message": "ERROR DESCRIPTION"}
    - If the check is successful, a JSON response will be returned:
    - {"status": "success"}

Chapter 7. Reports

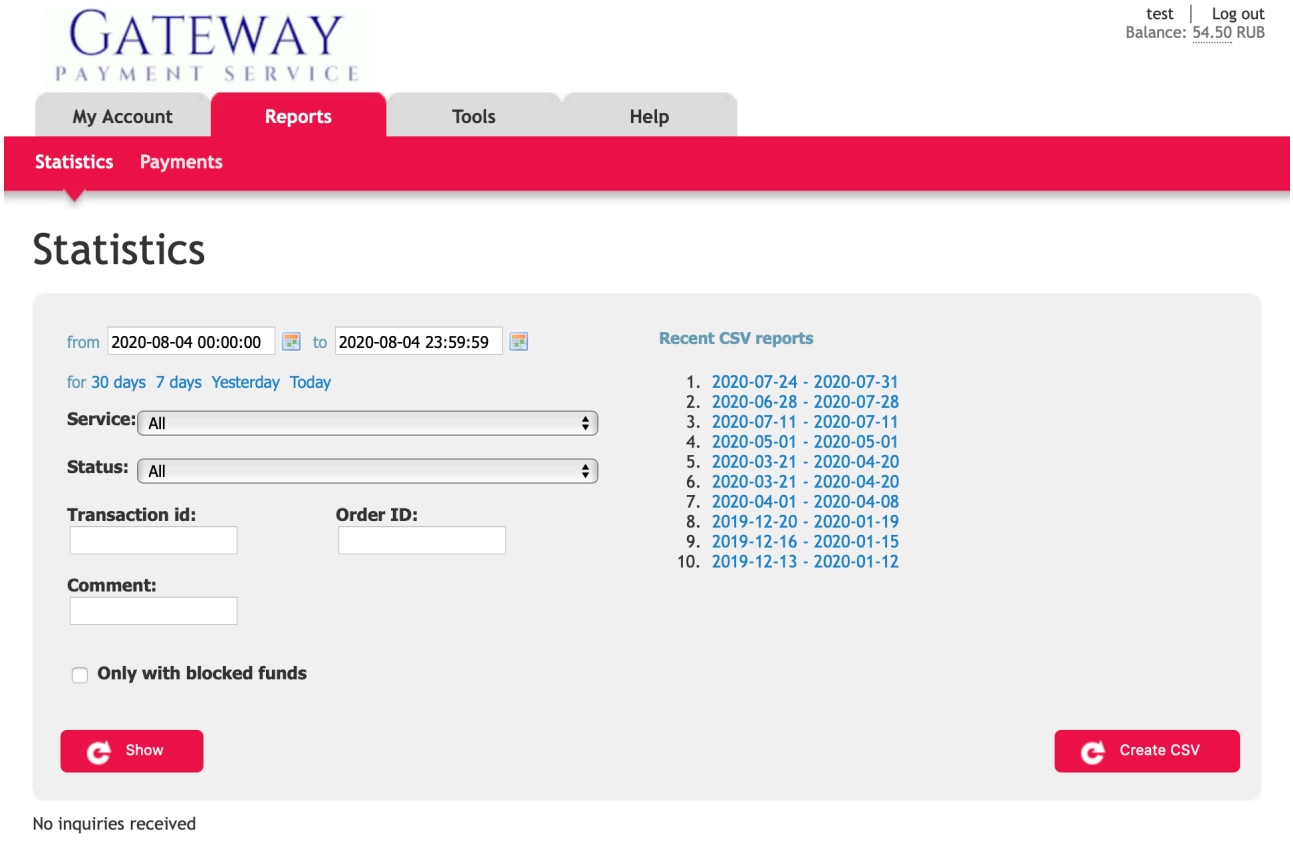
This chapter contains the next sections:

Section	Description	Page
7.1.	General information	89
7.2.	Payments report	90
7.3.	Transactions report	91
7.4.	Configuring notifications about successful transactions	92
7.5.	Sending of acts and details by e-mail	93

This page doesn’t contain any information

7.1. General information

To work with reports you should go to the Reports page of the principal bar:

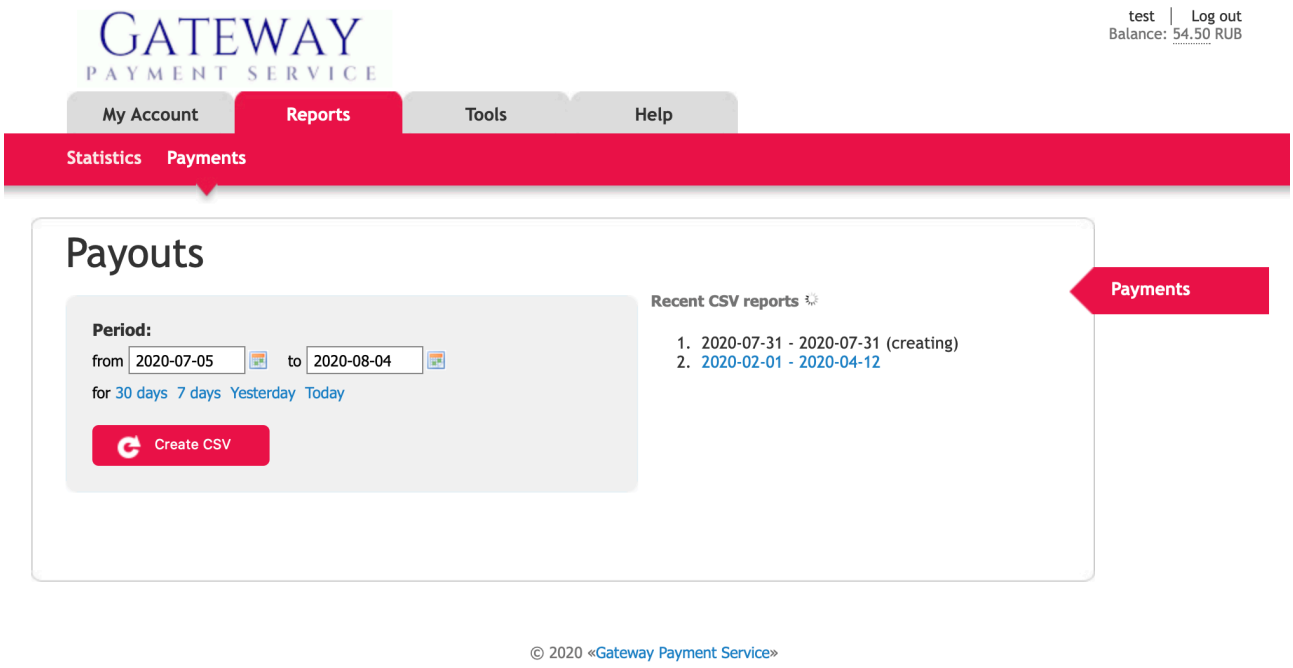


7.2. Payments report

To generate a report, go to the **Reports** → **Payments menu**.

This report contains information about all payment orders generated for the specified period, broken down by transaction numbers and income amounts.

Select the period for which you want to receive information and click "Request csv generation". After a while, the report will be available for download on this page.



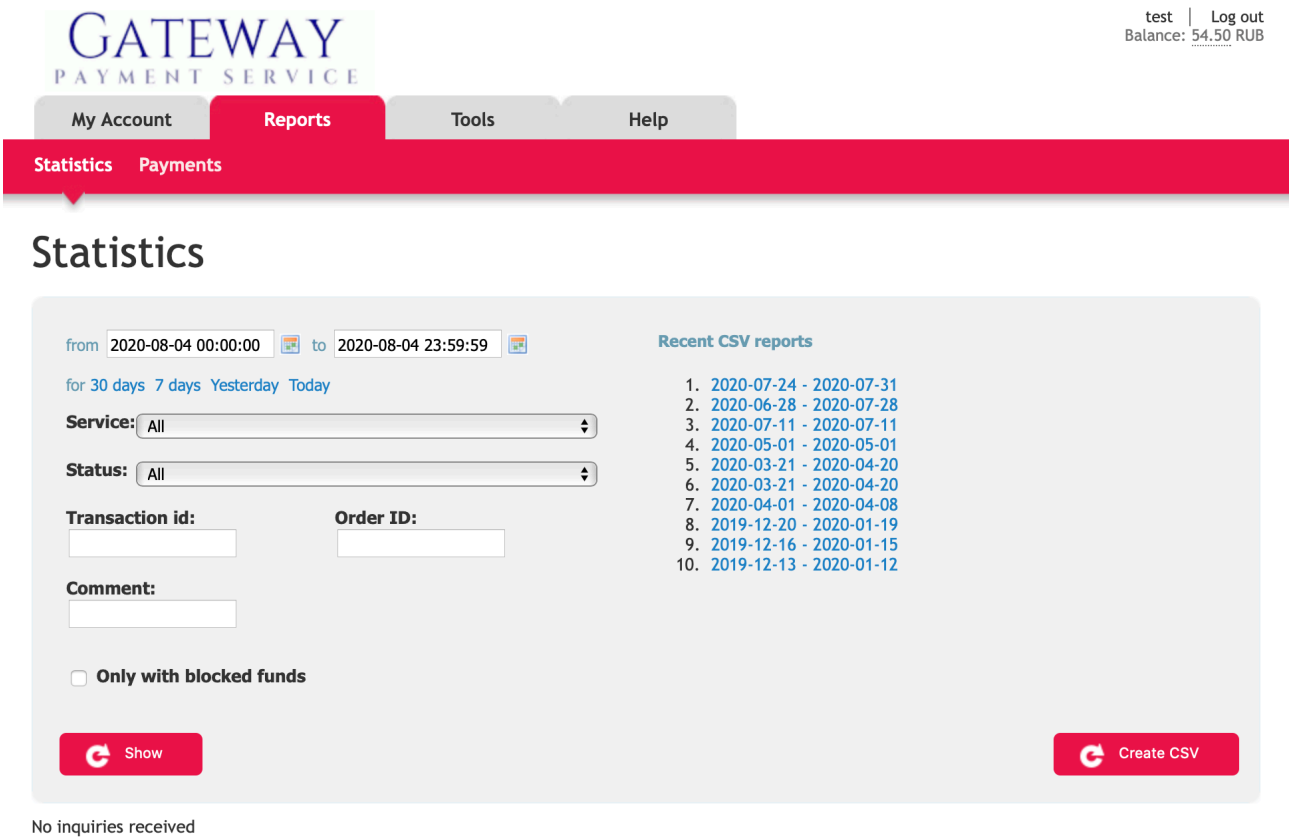
7.3. Transactions report

To receive a report on the statistics of past payments, go to the Reports → Statistics menu.

This report contains statistics of payments for the specified period without reference to the numbers of payment orders.

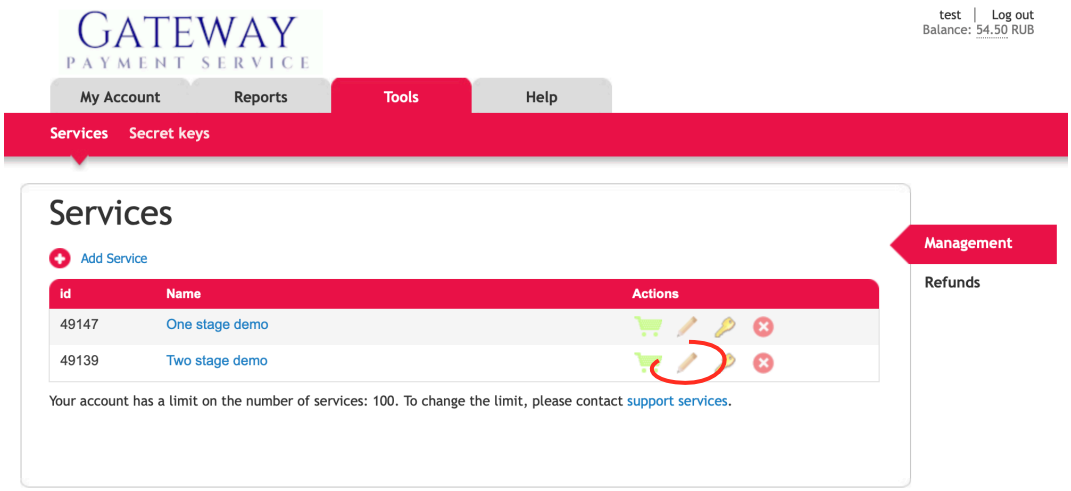
Select the period, transaction status (optional) and the service for which you want to get statistics (optional). You can also select additional filters, such as by payment method.

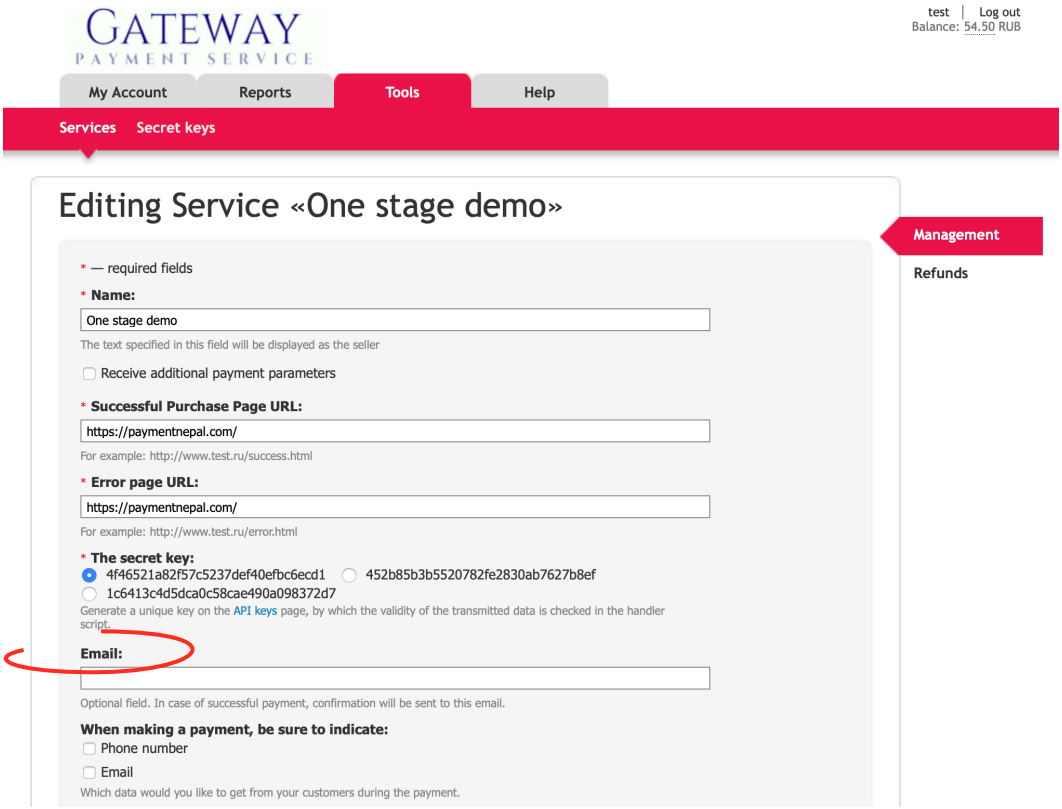
After that, click "Show" to display statistics in the LC interface, or "Upload to csv" to upload the selected report to a csv file.



7.4. Configuring notifications about successful transactions

To receive letters after each successful operation, specify the email address to which you want to receive notifications in the **Tools** settings in the merchant profile:





7.5. Sending of acts and details by e-mail

To set up notifications go to the menu **Merchant profile → PROFILE**

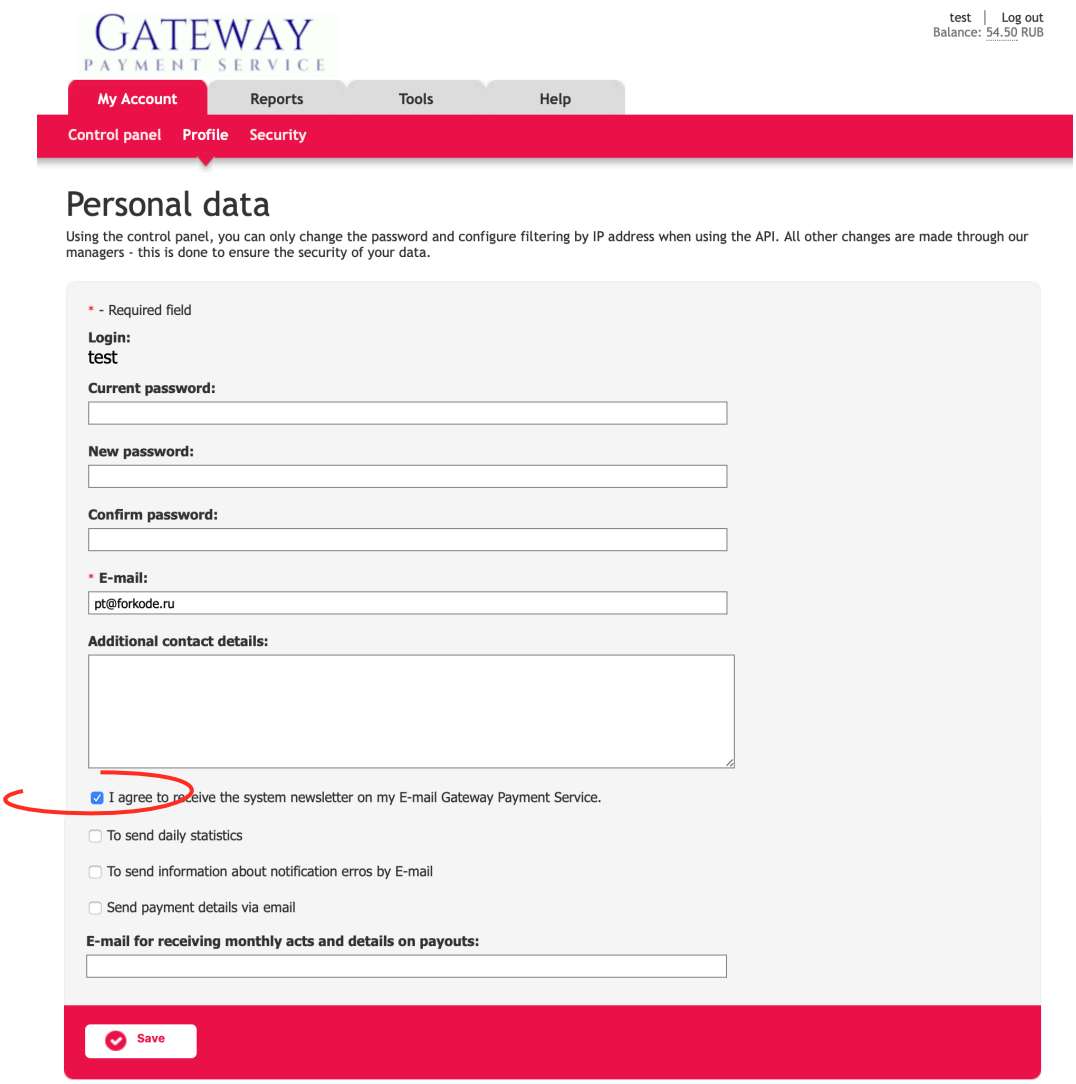
Check the boxes below:

I agree to receive the newsletter of the GPS on my E-mail - if you want to receive the newsletter from GPS about the ongoing work, new functionality, etc.

Send daily statistics - if you want to receive daily statistics on past transactions for the previous day.

Send details on payments to E-mail - if you want to receive a daily report on payments to the current account with a breakdown by transactions

Enter your email address in the box at the bottom of the page and click "Save".



If necessary, check the box «To send information about notification error by E-mail".

If notifications were not delivered to your processor for any transactions, a letter containing the error code and the full text of the notification will be sent to your email address. You can read more about the handler and notifications in the Sec. 4.4.

Chapter 8. Attachments

This chapter contains the next sections:

Section	Description	Page
8.1.	Terms and abbreviations	97
8.2.	External documents references	99

This page doesn’t contain any information

3.1. Terms and abbreviations

3

3D-Secure Is an XML-based protocol designed to be an additional security layer for online credit and debit card transactions.

A

API Application programming interface

Authorization Is an approval from a card issuer, usually through a credit card processor, that the customer has sufficient funds to cover the cost of the transaction.

B

BO Back-office, of the SM3000 IAP, where the Operator’s employers work to maintain the Platform jobs, as Merchants, Transactions, Agents, Reports and file exchange with a main Processing system.

C

Cardholder A person who owns a card, such as a cardholder of a credit card or debit card

ChargeBack Is a return of money to a payer. Most commonly the payer is a consumer. The chargeback reverses a money transfer from the consumer's credit card. The chargeback is ordered by the bank that issued the consumer's payment card.

F

FE Front-end, of the SM3000 IAP, where the cards authorizations are processed in on-line mode

I

IAP Internet acquiring platform. The Platform created as a separate application for the Payment operators and Payment facilitators.

ID Identification number (f.e. transaction ID or Merchant ID)

Incoming-File The data file, that Platform receives from the Bank’s processor

L

Light API The interface to connect the Merchant’s own platform to the SM3000 IAP

M

MasterCard MasterCard International payment system

Merchant	A legal entity carrying out trading activities on the Internet using the software provided by the system
MPI	Merchant Plug-in

O

Operator	Payment operator or Payment facilitator, that uses SM3000 IAP
Outgoing-File	The data file, that the Platform sends to the Bank's processor

P

PAN	Primary account number, or simply a card number, is the card identifier found on payment cards, such as credit cards and debit cards, as well as stored-value cards, gift cards and other similar cards.
Payment Gateway	A hardware-software complex developed and supported by a payment system that automates the acceptance of payments on the Internet.
Payment System	Payment system between users, financial organizations and business organizations. Allows you to pay, bills and purchases, transfer money.

R

Refund	A process in which a customer returns a product to the original retailer in exchange for money previously paid
Reversal	The operation of crediting funds to the payer's account as compensation for the cancellation of the provision of the service or the poorly rendered service.

S

Service	Merchant's service entry, registered for each MCC. It has its own parameters, fees etc.
SM3000	Sequoia Mosaic 3000. The processing platform of the cards issuing and acquiring processing, ATMs, POSs, e-commerce and m-commerce processing
System	A payment system that allows you to transfer money, accept payment for goods and services through various payment gateways.

T

Transaction	Within the framework of this service, a completely completed data exchange operation with a payment system, including debiting / crediting funds to an end user account.

V

VISA	VISA International payment system

3.2. External documents references

This manual doesn’t have any external link to the other documentation of SM3000 IAP.

ALFEBA

Agraciada 2770 Montevideo 11823 Uruguay

Phone: + 598 2 2083142

E-mail: docs@alfeba.com

www.alfeba.com