Silentpay mode

- Transfer of payment parameters
- Additional parameters for System of Fraud Intelligence (SOFI)
- 3D-Secure authorization
  - 3D-Secure authorization using the protocol version 1
  - 3D-Secure authorization using the protocol version 2

A hidden payment mode, when all data about the order, about the customer, about the payment method and payment means are transmitted directly by the merchant, can be performed using a card or a token.

Transfer of payment parameters

To use this mode, the *silentpay* web service is provided.

URL for payment via *silentpay* :

https://<SERVER-NAME>/pay/silentpay.cfm

Silentpay service parameters:

Name	Mandatory	Accepted values	Default value	Description
merchant_Id	Yes	Number		Merchant identification number in IPS Assist
login	Yes	String		Service Login
password	Yes	String		Password
orderNumber	Yes/No	128 symbols		Order number (order identification on the merchant side)
orderAmount	Yes	Number, 15 digits, two digits after the delimiter (delimiter '.')		Payment amount (ex.: 10.34)
orderCurrency	No	3 symbols	Default currency of enterprise or merchant	Order currency code (for *orderAmount* value) Ex.: RUB, USD, EUR and so on.
orderComment	No	4000 symbols		Order comment
delay	No	0 - one stage payment, 1- two stage payment	0	Flag for selection of one or two payment stages ⚠ If you want to use the double-stage payment mechanism, you should first consult with the support service staff (support@belassist.by) .
language	No	RU - Russian EN - English	Enterprise or merchant language	Language of payment pages
clientIP	No/Yes			IP-address of customer. IP-address of customer. The parameter is mandatory for the 3D-Secure protocol version 2.

cardType	No	1 - VISA 2 - EC/MC 3 - DCL 4 - JCB 5 - AMEX 6 - MIR		Card type
cardNumber	Yes			Card number
cardHolder	No*	70 letters (no digits). Space as delimiter		Card-holder
expireMonth	Yes	1-12		Card expiration month
expireYear	Yes	Year in YYYY format		Card expiration year
cvc2	Yes			CVC2 code
lastName	No*	70 letters (no digits)		Customer second name
firstName	No*	70 letters (no digits)		Customer first name
middleName	No	70 letters (no digits)		Customer middle name
email	No*	128 symbols		E-mail of customer
address	No	256 symbols		Customer address
mobilePhone	No	20 symbols		Customer mobile phone
country	No	3 symbols		Customer's country code
state	No	3 symbols		Customer's region code
city	No	70 symbols		Customer's city
zip	No	25 symbols		Customer's postal index
isConvert	No	0 - don't convert to the base currency 1 - don't convert to the base currency if possible 2 - always convert to the base currency	1	Currency conversion indicator
format	No	1 - CSV 3 - XML 4 - SOAP 5 - JSON	1	Results format. If the request is sent in SOAP or in JSON format, then the response will also be in SOAP or in JSON respectively , in other cases, in accordance with the passed format value.
signature	No	String		The string is joined according to determined rules. Then the MD5 hash prepared from this string. Hash is signed by private RSA key of the merchant. Key length - 1024. Received bit sequence is a signature. Signature is t ransferred BASE64 coded string. **Attention!** The parameter is necessary in order to protect the transmitted data from the possibility of their substitution by intruders. You should also enable the setting for check value or signature in the Personal account.
recurringIndicator	No	1 - recurring payment 0 - standard payment	0	Recurring payment indicator This parameter must not be transmitted simultaneously with the *customer Number* parameter.

recurringMinAmount	No/Yes	Number, 15 digits, two digits after the delimiter (delimiter '.')		Minimum payment amount for recurring payments. Mandatory when *recurringIndicator = 1*. ⚠ This parameter must not be transmitted simultaneously with the *customerNumber* parameter.
recurringMaxAmount	No/Yes	Number, 15 digits, two digits after the delimiter (delimiter '.')		Maximum payment amount for recurring payments. Mandatory when *recurringIndicator = 1.* ⚠ This parameter must not be transmitted simultaneously with the *customerNumber* parameter.
recurringPeriod	No/Yes	3 digits number		Recurring interval in days. Mandatory when *recurringIndicator = 1*. ⚠ This parameter must not be transmitted simultaneously with the *customerNumber* parameter.
recurringMaxDate	No/Yes	Date in string representation DD.MM.YYYY		Finish date of recurring payments. Mandatory when *recurringIndicator = 1*. ⚠ This parameter must not be transmitted simultaneously with the *customerNumber* parameter.
customerNumber	No	32 symbols		Merchant's internal customer identification ⚠ This parameter must not be transmitted simultaneously with the *recurringIndicator, recurringMinAmount, recurringMaxAmount, recurringPeriod, recurringMaxDate* parameters.
saveCard	No	1 - the card is stored to this customer number; 0 - the card is not stored.	0	This parameter permits to store the card to this client number for subsequent payments, if the current payment is successful. If this card for this client number already has been saved before, the parameter is ignored.
disable3DS	No	0 - perform 3D-Secure authorization according to the merchant settings; 1 - fulfill payment without 3D-Secure.	0	Flag of disabling 3D-Secure. The use of this operating mode is possible in agreement with Assist. To configure it, you need to contact the support service (support@belassist.by). When using this parameter, it must also be added to the order signature, which is built according to determined rules.
challengeResponseNotificationUrl	No	255 symbols		The URL to which the 3D-Secure completion result is sent and the customer is redirected after the *Challenge*.

⚠ All request parameters are automatically validated. The validation rules are given in the table " The validation rules for input parameters ".

HTTP POST request example :

```
<FORM ACTION="https://SERVER-NAME/pay/silentpay.cfm "method="POST">
<INPUT TYPE="hidden" NAME="merchant_Id" VALUE="Your_merchant_ID">
<INPUT TYPE="hidden" NAME="login" VALUE="Service login">
<INPUT TYPE="hidden" NAME="password" VALUE="Password">
<INPUT TYPE="hidden" NAME="orderNumber" VALUE="011001-10">
<INPUT TYPE="hidden" NAME="orderAmount" VALUE="22">
<INPUT TYPE="hidden" NAME="orderCurrency" VALUE="RUB">
<INPUT TYPE="hidden" NAME="orderComment" VALUE="Order 011001-10">
<INPUT TYPE="hidden" NAME="delay" VALUE="0">
<INPUT TYPE="hidden" NAME="isConvert" VALUE="1">
<INPUT TYPE="hidden" NAME="language" VALUE="RU">
<INPUT TYPE="hidden" NAME="clientIP" VALUE="Customer IP address">
<INPUT TYPE="hidden" NAME="cardType" VALUE="Card type">
<INPUT TYPE="hidden" NAME="cardNumber" VALUE="Card number">
<INPUT TYPE="hidden" NAME="cardHolder" VALUE="Card-holder">
<INPUT TYPE="hidden" NAME="expireMonth" VALUE="Card expiration- month">
<INPUT TYPE="hidden" NAME="expireYear" VALUE="Card expiration - year">
<INPUT TYPE="hidden" NAME="cvc2" VALUE="CVC2 or CVV code">
<INPUT TYPE="hidden" NAME="lastName" VALUE="Second name">
<INPUT TYPE="hidden" NAME="firstName" VALUE="First name">
<INPUT TYPE="hidden" NAME="middleName" VALUE="Middle name">
<INPUT TYPE="hidden" NAME="email" VALUE="Email">
<INPUT TYPE="hidden" NAME="address" VALUE="Customer address">
<INPUT TYPE="hidden" NAME="mobilePhone" VALUE="Customer mobile phone">
<INPUT TYPE="hidden" NAME="fax" VALUE="Customer fax number">
<INPUT TYPE="hidden" NAME="country" VALUE="Customer's country">
<INPUT TYPE="hidden" NAME="state" VALUE="Customer's region">
<INPUT TYPE="hidden" NAME="city" VALUE="Customer's city">
<INPUT TYPE="hidden" NAME="zip" VALUE="Customer postal index">
<INPUT TYPE="hidden" NAME="testMode" VALUE="Test mode">
<INPUT TYPE="hidden" NAME="format" VALUE="Result format">
<INPUT TYPE="Submit"></FORM>
```

The service description for SOAP format can be found on page:

https:// <SERVER-NAME> /pay/silentpay.wsdl

Returned values:

Name	Description
ordernumber	Order number
billnumber	IPS Assist bill number
testmode	Test mode
ordercomment	Comment
orderamount	Original order amount
ordercurrency	Original order currency
amount	Payment amount
currency	Payment currency
rate	Currency rate
firstname	Customer first name

lastname	Customer second name
middlename	Customer middle name
email	Email
ipaddress	IP-address of customer
meantypename	Payment mean type
*Customer card options**	
meanssubtype	Payment mean sub-type
meannumber	Payment mean number
cardholder	Payment mean holder
cardexpirationdate	Card expired date
issuebank	Issuer-Bank name
bankcountry	Issuer-Bank country
Order parameters	
orderdate	Order date (GMT)
orderstate	Order status
responsecode	Return code
message	Message
customermessage	Message for customer
recommendation	Recomendations
approvalcode	Authorization code
protocoltypename	Protocol
processingname	Processing
operationtype	Operation type
packetdate	Request date (GMT)

signature	Signature. It is created according to the following algorithm: 1) A combined string is created from the parameters (in their string representation, in the format as they are passed in the response): *billnumber, ordernumber, responsecode, orderamount, ordercurrency, meannumber, approvalcode, orderstate, packetdate* (without delimiters). 2) The received string is signed with IPS Assist private key. 3) The resulting byte sequence is encoded in BASE64.
pareq	3D-Secure authorization request packet
ascurl	URL for 3D-Secure authorization redirect
*KROK parameters***	
qrurl	Link with QR code for payment
qrtimeout	Payment link expiration date (date and time)
qrtype	Link type, *qrdynamic* - dynamic

* Parameters are not sent if the request contains the *krokPayment* value for the payment type (payment via KROK).

** Parameters are sent only if the request contains the *krokPayment* value for the payment type (payment via KROK).

There are performance limitations when using the service.

Request result will be provided according the requested format.

For CSV format:

```
Field name:field value Field name:field value....Field name:field value
```

For XML format:

```
<?xml version='1.0' encoding='UTF-8' standalone='yes'?>
<!DOCTYPE result [
<result firstcode='first code' secondcode='second code' count='objects count'>
<orders><order>
<ordernumber>Order number</ordernumber>
<responsecode>Return code</responsecode>
<recommendation>Recommendations</recommendation>
<message>Message</message>
<ordercomment>Comment</ordercomment>
<orderdate>Payment date/time</orderdate>
<amount>Payment amount</amount>
<currency>Currency code</currency>
<meantypename>Card type</meantype>
<meannumber>Card Number</meannumber>
<lastname>Second name</lastname>
<firstname>First name</firstname>
<middlename>Middle name</middlename>
<issuebank>Issuer-Bank name</ issuebank >
<email>E-mail</email>
<bankcountry>Issuer-Bank country code</bankcountry>
<rate>Currency rate</rate>
<approvalcode>Authorization code</approvalcode>
<meansubtype>Card sub-type</meansubtype>
<cardholder>Card-holder</cardholder>
<cardexpirationdate>Card expired date</cardexpirationdate>
<ipaddress>IP-address</ipaddress>
<protocoltypename>Protocol type</protocoltypename>
<testmode>Test mode payment indicator</ testmode >
<customermassage>Message for customer</customermassage >
<orderstate>Order status</orderstate>
<processingname>Processing name</ processingname>
<operationtype>Operation type</operationtype>
<billnumber>Bill number</billnumber>
<orderamount>Original payment amount</orderamount>
<ordercurrency>Original currency </ordercurrency>
<paketdate> Packet data</paketdate>
<signature> </signature>
<pareq>pareq value</pareq>
<ascurl>URL of Issuer-Bank </ascurl>
</order></orders></result>
```

For SOAP format:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://www.paysecure.ru/ws
/">
<soapenv:Header/>
<soapenv:Body>
<ws:SilentPayResponse>
<return>
<ordernumber xsi:type="xsd:string">Order number</ordernumber>
<responsecode xsi:type="xsd:string">Return code</responsecode>
<recommendation xsi:type="xsd:string">Recommendations</recommendation>
<message xsi:type="xsd:string">Message</message>
<ordercomment xsi:type="xsd:string">Comment</ordercomment>
<orderdate xsi:type="xsd:string">Payment date/time</orderdate>
<amount xsi:type="xsd:string">Payment amount</amount>
<currency xsi:type="xsd:string">Currency code</currency>
<meantypename xsi:type="xsd:string">Card type</meantype>
<meannumber xsi:type="xsd:string">Card number</meannumber>
<lastname xsi:type="xsd:string">Second name</lastname>
<firstname xsi:type="xsd:string">First name</firstname>
<middlename xsi:type="xsd:string">Middle name</middlename>
<issuebank xsi:type="xsd:string">Issuer-Bank name</ issuebank >
<email xsi:type="xsd:string">E-mail</email>
<bankcountry xsi:type="xsd:string">Issuer-Bank country code</bankcountry>
<rate xsi:type="xsd:string">Currency rate</rate>
<approvalcode xsi:type="xsd:string">Authorization code</approvalcode>
<meansubtype xsi:type="xsd:string">Card sub-type</meansubtype>
<cardholder xsi:type="xsd:string">Card-holder name</cardholder>
<cardexpirationdate xsi:type='xsd:string'>Card expired date</cardexpirationdate>
<ipaddress xsi:type="xsd:string">customer IP-address</ipaddress>
<protocoltypename xsi:type="xsd:string">Protocol type</protocoltypename>
<testmode xsi:type="xsd:string">Test mode payment flag</ testmode >
<customermassage xsi:type="xsd:string">Message fro customer</customermassage >
<orderstate xsi:type="xsd:string">Order status</orderstate>
<processingname xsi:type="xsd:string">Processing name</processingname>
<operationtype xsi:type="xsd:string">Operation type</operationtype>
<billnumber xsi:type="xsd:string">Bill number</billnumber>
<orderamount xsi:type="xsd:string">Original payment amount</orderamount>
<ordercurrency xsi:type="xsd:string">Original payment currency</ordercurrency>
<paketdate xsi:type="xsd:string">packet date</paketdate>
<signature xsi:type="xsd:string"> </signature>
<pareq xsi:type="xsd:string">pareq value </pareq>
<ascurl xsi:type="xsd:string">Issuer-Bank URL </ascurl>
</return>
</ws:SilentPayResponse>
</soapenv:Body>
</soapenv:Envelope>
```

Result of request in JSON format for a payment made via KROK.

```
{
    "silentpay": {
        "meantypename": "KROK",
        "customermessage": " ",
        "message": " ",
        "qrtype": "qrdynamic",
        "testmode": 0,
        "protocoltypename": "NET",
        "qrurl": "https://fake.krok.by/l3aula6l6rmz6blv0ax89nzp",
        "operationtype": 100,
        "customer": {
            "lastname": "Test",
            "firstname": "Test",
            "middlename": "Test",
            "ipaddress": "127.0.0.1",
            "email": "null@assist.by"
        },
        "orderdate": "23.09.2026 13:59:09",
        "packetdate": "23.09.2026 13:59:10",
        "orderamount": 100,
        "ordercomment": "",
        "ordercurrency": "BYN",
        "recommendation": "",
        "processingname": "KROK",
        "qrtimeout": "23.09.2026 14:19:09",
        "orderstate": "In Process",
        "rate": 1,
        "billnumber": "5489874275265301.1",
        "currency": "BYN",
        "amount": 100,
        "ordernumber": "14d5s2f78sdsaf9s5",
        "responsecode": "AS300"
    }
}
```


In case of successful payment the return code *responsecode* is equal to AS000 value.

In case of unsuccessful payment the *responsecode* is one of AS100-AS998 values (except AS110 code that is returned when 3-D Secure authorization required - see here , for details).

If the request can't be processed the *firstcode* and *secondcode* parameters have non-zero values.

If *responsecode* AS300 is received, and order status (orderstate) and operation status (*operationstate*) are *'In Process* ', the current status of the payment can be obtained later via the request to the web-service *orderresult* .

If the payment result is not received (for example, due to network problems), then you can get it later via the request to the web-service *orderresult* .

Result example in XML format for error return (not correct password):


```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
  <!DOCTYPE result [...]>
  <result firstcode="7" secondcode="102" count="0"></result>
```


For the decryption of **firstcode** and **secondcode** values, please, refer here.

An enterprise can also initiate the provision of subscription services for payments.

Additional parameters for System of Fraud Intelligence (SOFI)

Merchants, which use *silentpay* service can provide additional parameters about the customer for analyses within the anti-fraud system.

In additional to the main list of the *silentpay* parameters the following optional parameters for SOFI can be provided in the request:

Name	Accepted values	Description

HEADER_HTTP_USER_AGENT	String (255 chars)	Header User Agent http from http request
HEADER_HTTP_ACCEPT	String (255 chars)	Header Accept http from http request
HEADER_HTTP_ACCEPT_LANGUAGE	String (128 chars)	Header Accept Language http from http request
HEADER_HTTP_REFERER	String (255 chars)	Header Referer http from http request
HEADER_REMOTE_HOST	String (16 chars)	Customer IP-address
HEADER_HTTP_FORWARDED	String (16 chars)	Header Forwarded http from http request
HEADER_HTTP_X_FORWARDED_FOR	String (16 chars)	Header Xforwarded-For http from http request
HEADER_HTTP_VIA	String (128 chars)	Header Via http from http request
CLIENT_JS_VER	String (16 chars)	Java script interpreter version
CLIENT_LOCAL_TIME	String (128 chars)	Customer local time
CLIENT_SCREEN_RES	String (16 chars)	Customer screen resolution (<width>x< high>)
CLIENT_SCREEN_COLORS	Decimal (numbers from 1 to 24)	Color depth of customer screen
CLIENT_JS_BROWSER_NAME	String (255 chars)	Customer browser name
CLIENT_TIME_ZONE	Decimal (5)	Customer time zone GMT offset in hours. Conversion formula is (- GMT_H). For example, offset GMT+2 is equal to - 2.
CLIENT_COOKIES	String (16 chars)	Unique browser identity from external system
CLIENT_JAVA	Logical (true, false)	Java script support enabled indicator
CLIENT_STYLESHEETS	Logical (true, false)	css styles support
CLIENT_BROWSER_PLATFORM	String (64 chars)	Browser platform name
CLIENT_SYSTEM_LANGUAGE	String (5 chars)	Language code of customer operational system
CLIENT_BROWSER_LANGUAGE	String (5 chars)	Language code of the browser
CLIENT_USER_LANGUAGE	String (5 chars)	Customer language code
CLIENT_PROCESSOR	String (16 chars)	Processor name of customer computer
CLIENT_CONNECTION	String (16 chars)	HTTP connection type
CLIENT_HOSTADDRESS	String (16 chars)	DNS lookup based on HOST_ADDRESS
CLIENT_HOSTNAME	String (70 chars)	Customer host name

3D-Secure authorization

It is possible to use the card, which requires 3D-Secure authorization for payment via *silentpay* service (when the shop and processing have configured settings for it).

When a payment by a card (which required 3D-Secure authorization) is processed the IPS Assist returns AS110 value in the *responsecode* replay field.

The additional fields are also added in the silentpay response packet. These fields allow the merchant to provide additional payer authentication using 3-D Secure technologies (VISA cards) and Mastercard SecureCode (Mastercard catds).

For cards of international payment systems VISA and Mastercard in most cases version 2 of the 3D-Secure protocol is used by additional authentication of the customer, the same version of the protocol is also used for UPI cards. For cards of the BELKART payment system, authentication is carried out according to version 1 of the protocol.

To start the order payment, the merchant sends an authorization request to the IPS Assist server. The following data about the customer device and browser must be added to the usual request parameters, if this has not been done before for operating with the SOFI . This data is required in the new 3-D Secure protocol version 2.

Name	Accepted values	Description
HEADER_HTTP_ACCEPT	String (255 chars)	Header Accept http from http request
HEADER_HTTP_USER_AGENT	String (255 chars)	Header User Agent http from http request
CLIENT_JAVA	Logical (true, false)	Java script support enabled indicator navigator.javaEnabled()
CLIENT_BROWSER_LANGUAGE	String (5 chars)	Language code of the browser navigator. language
CLIENT_SCREEN_COLORS	Decimal (15)	Color depth of customer screen Screen. pixelDepth
CLIENT_SCREEN_RES	String (16 chars)	Customer screen resolution Screen.width + 'x' + screen.height
ChallengeWindowSize	2 chars (01 - 250x400, 02 - 390x400, 03 - 500x600, 04 - 600x400, 05 - Full screen)	IFrame size for cardholder verification
ClientIP	Maximum of 15 digits, 4 delimiters «.»	IP-address of the customer

3D-Secure authorization using the protocol version 1

When a payment by a card requiring authorization using the protocol version 1 is processed, the IPS Assist returns the response code AS110 and additional fields *pareq* and *acsurl* in response to the authorization request.

The customer has to be forwarded to the issuer-bank site by URL from *acsurl* field.

The following values should be provided in the request form:

AcsUrl	Url of Issuer-Bank: value which is received from IPS Assist in the *acsurl* field
PaReq	The value which is received from IPS Assist in the *pareq* field
TermUrl	Url of the shop for results from issuer-bank receiving
MD	Identification that is used for further reference between order and 3D-Secure authorization

Form example for Issuer-Bank request:

```
<FORM ACTION="acsurl value that is received from IPS Assist" method="POST">
<INPUT TYPE="hidden" NAME="PaReq" VALUE="pareq value that is received from IPS Assist">
<INPUT TYPE="hidden" NAME="TermUrl" VALUE="url of the shop">
<INPUT TYPE="hidden" NAME="MD" VALUE="Any identity (provided by shop)">
<INPUT TYPE="submit" NAME="Submit_3DS" CLASS="button" VALUE="Continue">
</FORM>
```

The issuer-bank returns (by URL which is provided in *TermUrl* parameter of the request) the following values:

PaRes	Result packet
MD	Identity which was provided in the request

After the receiving of the reply from Issuer-Bank the shop has to transfer the 3D-Secure authorization result (*pares* value) to the IPS Assist. Web-service *get3DSec* can be used for it.

Get3DSec - transfer of the 3D-Secure authorization result

Service request URL:

https://<SERVER-NAME>/get3dsec/ws3dsec.cfm.

Request and replay format: SOAP **,** a wsdl-description is available by the following URL:

https://payments.paysecure.ru/get3dsec/get3dsec.wsdl .

The merchant has to transfer the *pares* value to IPS Assist. The following request in SOAP format should be sent to the IPS Assist:

Input parameters:

Method *: send3dsparams*

Parameter	Mandatory	Description
merchant_Id	Yes	Merchant identification number in IPS Assist
login	Yes	Service login
password	Yes	Password
ordernumber	Yes	Order number
pares	Yes	3DS result packet
language	No	Language

Example of the request:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
<s:Body>
<send3dsparams xmlns="urn:assist-processor">
<merchant_id>Merchant ID</merchant_id>
<login>Login</login>
<password>Password</password>
<ordernumber>Order number</ordernumber>
<language>Language</language>
<pares>value pares which is received from issuer-bank</pares>
</send3dsparams>
<s:Body>
<s:Envelope>
```

Return values - the same as in the *silentpay* request result.

In case of request failure:

```
<?xml version="1.0" encoding="windows-1251" standalone="no" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Fault>
<faultcode>First code</faultcode>
<faultstring>Second code</faultstring>
<detail />
</SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3D-Secure authorization using the protocol version 2

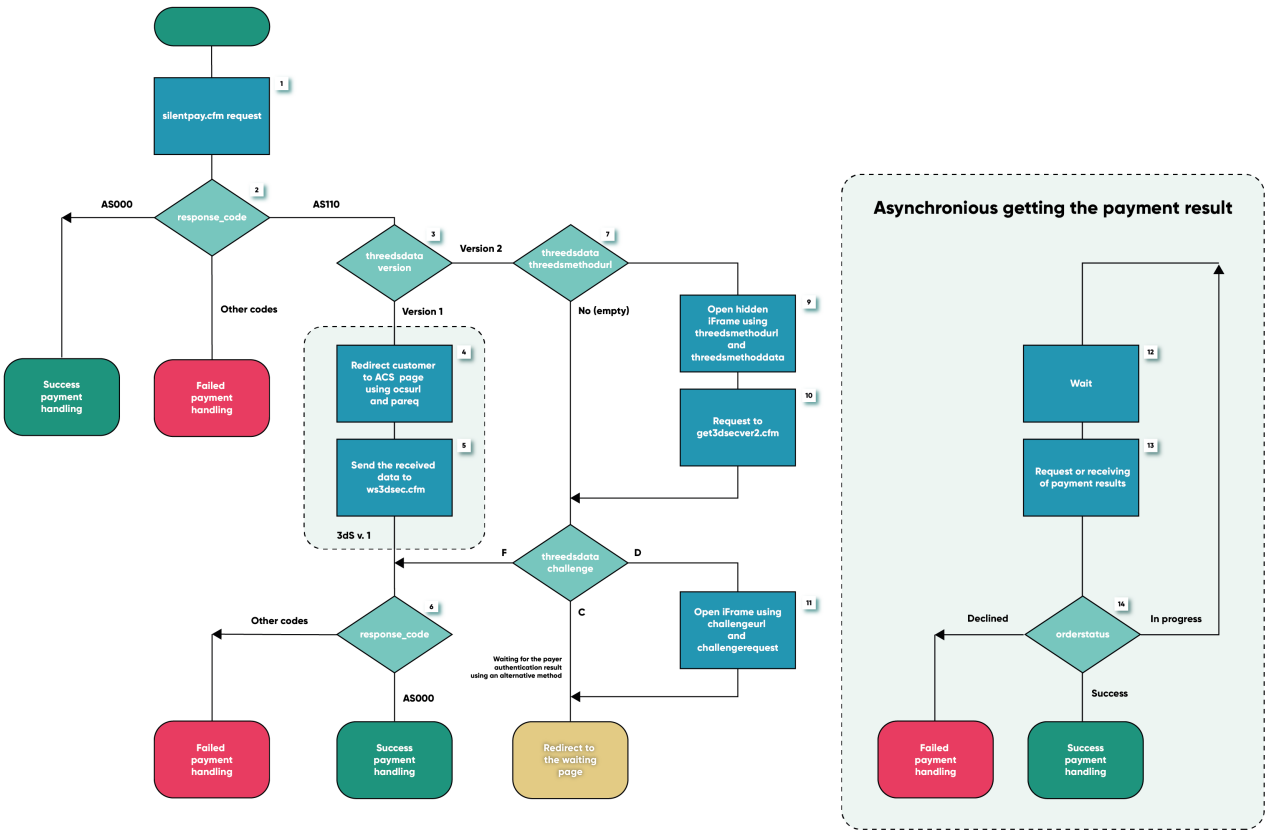
The customer can be authenticated without entering an additional password (based on advanced data) - this is an important feature of the new version of the protocol. The authentication process, in which there is no additional interaction with the cardholder, is called Frictionless Flow. The authentication process, in which the cardholder is required to enter an additional verification code, is called the Challenge Flow process.

Also, before authentication using the new protocol, the client's browser must send an additional request to the ACS of the issuing bank (further we will call it 3DSMethod).

To work with the new protocol, the following changes must be made on the enterprise's side:

1. Check the version of the 3D-Secure protocol in response to an authorization request to the Assist service. For version 1, support the workflow described above.
2. For version 2, create a hidden iFrame on the payment page (for a detailed parameters description see below) and send 3DSMethod request (if available) to the issuing bank ACS.
3. To continue authentication, call the *get3dsecver 2* web service with additional 3D-Secure parameters. If authentication occurs without additional interaction with the customer (Frictionless Flow), then Assist will receive its result and send the authorization transaction to the processing. The enterprise will receive in response a full payment result, which also contains the result of authorization in processing. If additional authentication of the customer is necessary, then the IPS Assist will return additional fields for verification (Challenge Flow) in response to the request.
4. If there are additional fields in the response that indicate the need for additional verification, the enterprise creates an iFrame on the payment page, which implements the display of the ACS page of the issuing bank to enter a one-time password. Customer completes authentication.
5. IPS Assist will receive the result of the verification to the server on its side. In case of successful verification, a payment transaction will be processed. If the verification fails, the operation will fail.
6. To find out the final result of payment for an order, the enterprise must use one of the methods for obtaining an authorization result.

The logic of the new protocol version 2 is displayed as a diagram. The description below contains links to numbered units of the diagram.



To start paying for the order, the enterprise sends an authorization request to the IPS Assist server and the necessary additional parameters (unit 1).

On the payment with a card requiring authorization under the protocol version 2 the IPS Assist will return the AS110 return code (unit 2) and an additional *threedsdata* parameter block in the response. The full list of parameters that may be contained in the *threedsdata* block is presented in the table below:

Name	Accepted values	Description	In the response of the service [1]
version	1 [1] (1.0.0, 1.0.2) 2 [1] (2.0, 2.1.0, 2.2.0)	3-D Secure Protocol version	1,2
threeDSServerTransID	String	3DS Server transaction ID	1,2
threeDSMethodURL	String (255 chars)	The ACS URL that will be used by the 3DS Method	1

threeDSMethodData	String (255 chars)	Request body (Base64 encoded)	1
alphaauthresult	Y - success, N - fail, A - Attempt, U - unable to authenticate, R - rejected, E - error, I - Informational Only	Authentication result will be received in case of Frictionless Flow authentication.	1,2
challenge	F - Frictionless Flow C - Challenge Flow D - Decoupled Authentication	Interaction with the customer (C - required, F - not needed, D - decoupled authentication)	1,2
challengeurl [2]	Full URL like https://acs .... 2048 symbols maximum	The ACS URL for payer verification	1,2
challengerequest [2]	String	Request body (Base64 encoded) for challengeurl request	1,2

1 The parameter may be contained in the response of the service: 1- *silentpay* ; 2 - *get3dserver2* .

2 In case of authorization without additional interaction with the customer (Frictionless Flow), the parameters *challengeurl* and *challengerequest* will not be returned.

Depending on the content of the received *threedsdata* block (units 3,7,8), the authentication proceeds differently.

The main operating scenarios for version 2 are determined by whether a 3DSMethod call is required (generation of a hidden iFrame in the client browser), as well as whether additional client authentication is required and under what scenario it takes place:

- If there is the issuing bank URL *threeDSMethodURL* (unit 7), the merchant generates hidden HTML iFrame on the payment page (unit 9) and sends a POST request with one parameter *threeDSMethodData* to the received address *threeDSMethodURL* , and then calls the get3dsecver2 service (unit 10).
- If there is the issuing bank URL *threeDSMethodURL* (units 7, 9, 10), without the additional interaction with the card holder (unit 8) - *Frictionless Flow (F)* , IPS Assist immediately executes a transaction in processing or completes the operation with an error. In response to a call to the get3dsecver2 service, the merchant will receive the payment status (unit 6).
- If there is the issuing bank URL *threeDSMethodURL* (units 7, 9, 10), and the additional interaction with the card holder is necessary (unit 8), the merchant generates hidden HTML iFrame on the payment page and sends an HTTP POST request for the card holder verification to the specified *challengeurl* URL (unit 11). This iFrame displays the issuer's bank ACS page and the customer enters a one-time password received from the bank. This is the *Challenge Flow (C)* scenario.
- If there is the issuing bank URL *threeDSMethodURL* (units 7, 9, 10), the additional interaction with the card holder is necessary (unit 8) and with decoupled authentication, the merchant must keep the order in the *In Process* state and wait for the final payment status (within the lifetime of the order). This is the *Decoupled Authentication (D)* scenario.
- If there is not the issuing bank URL *threeDSMethodURL* >and when the additional interaction with the card holder is not required - *Frictionless Flow (F)* , the transaction will be processed immediately and the payment process will be completed (unit 6). The merchant will receive the authentication result and payment status immediately in response to the call to the *sile ntpay* service.
- If there is not the issuing bank URL *threeDSMethodURL* and the additional interaction with the card holder is necessary (unit 8), but with no issuing bank URL threeDSMethodURL, the enterprise should generate an HTML iFrame object on the payment page and send an HTTP POST request for the card holder verification to the specified *challengeurl*URL (unit 11). This iFrame displays the issuer's bank ACS page and the customer enters a one-time password received from the bank.
- If there is not the issuing bank URL *threeDSMethodURL* and the additional interaction with the card holder is necessary (unit 8) under the *Decoupled Authentication* scenario (*D*), the merchant must keep the order in the *In Process* state and wait for the final payment status (within the lifetime of the order).

In those work scenarios that require the generation of a hidden HTML iFrame, at step 7 in the *threedsdata* unit, the merchant will receive the necessary *threeDSMethodData* and *threeDSMethodURL* parameters for generating a POST request (unit 9). The result of sending the request can be positive (HTTP code 200), negative (any other HTTP code), or the request sending timeout value will be exceeded (set to 10 seconds). After receiving the HTTP code or the timeout has expired, a request must be sent to the *get3dsecver2* service (unit 10) to continue the authentication process.

get3dserver2 - web service of the 3D-Secure authentication continuation

Service request URL:

https://<SERVER-NAME>/get3dsec/get3dsecver2.cfm

Supported formats: SOAP, JSON.

Input parameters:

Name	Accepted values	Description

merchant_Id	Number	Merchant identification number in IPS Assist
login	String	Service Login
password	String	Password
billNumber	15 or 16 digits, extended payment number	Unique number of payment in IPS Assist
threeDSServerTransID	String	3DS Server transaction ID

IPS Assist continues the process of authentication of the customer in the payment system and the issuing bank through 3DS Server.

If additional verification of the customer is not required (Frictionless Flow) (unit 10), IPS Assist executes a transaction in processing or completes the operation with an error (depending on processing and enterprise settings, and authentication result) (unit 12).

The response to the request in this case will contain one of the final response codes (AS000 - operation completed successfully, AS100-AS109 - denial of authorization), all response fields described above , and an additional *threedsdata* data block in which the challenge parameter is F, and the *alphaauthresult* field contains the authentication result (Y, N, U, R, I).

Receiving an AS110 response code in the response of the *get3dsecver2* web service means that an additional card holder verification is required (Challenge Flow). In the *threedsdata* data block, the *challenge* parameter will be equal to C, and the *challengeurl* and *challengerequest* parameters will be filled (block 8). In this case, the enterprise should generate an HTML iFrame object on the payment page and send an HTTP POST request for the card holder verification to the specified URL (block 11) with the *creq* parameter, in which pass the received *challengerequest* value. This iFrame displays the ACS page of the issuing bank and the customer enters a one-time password received from the bank.

In a decoupled authentication scenario, the *challenge* parameter in the *threedsdata* data block will be equal to *D* , and the *challengeurl* and *challengerequest* parameters will be absent (block 8). The merchant must in this case keep the order in the *In Process* state and await the final payment status.

IPS Assist will receive the result of this verification on its server in asynchronous mode. Depending on the authentication result and the settings of the processing and of the enterprise, the IPS Assist will executes a transaction in processing or completes the operation with an error.

In order for the buyer to be able to return back to the merchant's website after passing the additional verification, the IPS Assist support service should be informed of the URL to return the buyer and receive the result of passing the additional verification. Receiving this request will mean for a merchant, that the additional verification is completed, and the merchant can now redirect the buyer's browser to the result page on its side and wait for the completion of the payment transaction in processing.

Receipt of the payment result after additional verification is reflected in the diagram in blocks 12, 13, 14.

> The process of receiving the result of an additional verification on the server of the IPS Assist is asynchronous. Only after receiving this result a payment transaction will be processed (or not conducted) in processing, which will lead to blocking of funds on the customer's card. The merchant should receive the result of the payment transaction from IPS Assist in one of the standard ways. The enterprise can send a request to the receiving of the payment result by order number web service or configure on its side to receive notifications sent by Assist to enterprise server.

An example of the *threedsdata* data block in which additional verification of the card holder is not required (in this case, the responsecode will be different from AS110):

```
<threedsdata>
<version>2.2</version>
<alphaauthresult>Y</alphaauthresult>
<challenge>F<challenge>
</threedsdata>
```

An example of a *threedsdata* data block in which additional card holder verification is required (responsecode is AS110):

```
<threedsdata>
<version>2.2</version>
<challenge><challenge>
<challengeurl>https://acs.superbank.ru/version20/creq</challengeurl>
<challengerequest>eyJ0aHJlZURTU2VydmVyVHJhbnNJRCI6ImE3ZWJlMDU3LTg2ZjgtNGFmMS05MTJkHGNlYTc5Mzc0OWUxMiIsImFjc1RyYW
5zSUQiOiI5ODhmOWZmYS1kNzYyLTQ0YjktOWI0OS01ZDRkMjU5YmRkZWQiLCJkc1RyYW5zSUQiOiJkMGJmZGQzYy00YzdhLTVmNjktODAwMC0wMD
AwMDAwOGM3NjMiLCJtZXNzYWdlVHlwZSI6IkNSZXEiLCJtZXNzYWdlVmVyc2lvbiI6IjIuMS4wIiwiY2hhbGxlbmdlV2luZG93U2l6ZSI6IjA0In
0</challengerequest>
</threedsdata>
```

To get a web service response in JSON format, you need to pass content-type = application/json or format = 5 in the request. Description of the parameters of all IPS Assist web services for the JSON format is contained in the swagger file at:

https://docs.belassist.by/swagger/

An example JSON response for an operation without additional authentication:


```
{
   "threedsdata": {
      "version": "2.2.0",
      "alphaauthresult": "Y",
      "challenge": "F"
   },
   "MakePaymentResponse": {
      "customermessage": "Success",
      "message": "Success",
      "token": "",
      "testmode": "0",
      "operationtype": "100",
      "orderdate": "23.10.2019 10:45:47",
      "packetdate": "23.10.2019 10:49:48",
      "orderamount": "15.00",
      "ordercomment": "",
      "cardexpirationdate": "12/20",
      "ordercurrency": "BYN",
      "recommendation": "",
      "processingname": "Fake",
      "meannumber": "220000****0001",
      "orderstate": "Approved",
      "rate": "1",
      "amount": "15.00",
      "responsecode": "AS000",
      "meantypename": "VISA",
      "protocoltypename": "NET",
      "bankcountry": "UNKNOWN",
      "customer": {
         "lastname": "Testov",
         "firstname": "",
         "middlename": "Testovich",
         "ipaddress": "10.10.10.10",
         "email": "null@assist.by"
      },
      "cardholder": "TEST",
      "approvalcode": "X38988",
      "billnumber": "5161957242913232.1",
      "issuebank": "UNKNOWN",
      "currency": "RUB",
      "ordernumber": "231020191345849_user2",
      "meansubtype": ""
   }
}
```


A response example for an operation with additional authentication:

```
{
   "threedsdata":    {
      "version": "2.2.0",
      "challengerequest":
"eyJ0aHJlZURTU2VydmVyVHJhbnNJRCI6ImQ3MDlmM2M0LTVkNTItNDNlZS1hMTcxLTQ2NzM0MDRlMzUxMyIsImFjc1RyYW5zSUQiOiDNZWNmOTk
4NC1jZGY2LTRkMWQtYTUzZS1kMzZhOTUyZTgzZDYiLCJkc1RyYW5zSUQiOiJlYWRlMmY4NC1jMDY1LTVmMGItODAwMC0wMDAwMDAwYjU3NWQiLCJ
tZXSKFWdlVHlwZSI6IkNSZXEEJCJtZXNzYWdlVmVyc2lvbiI6IjIuMS4wIiwiY2hhbGxlbmdlV2luZG93U2l6ZSI6IjA0In0",
      "challengeurl": "https://acs.superbank.ru/acs/creq",
      "challenge": "C"
   },
   "MakePaymentResponse":    {
      "customermessage": "New Client authentication by 3DSecure technology.",
      "message": "New Client authentication by 3DSecure technology.",
      "token": "",
      "testmode": "0",
      "operationtype": "100",
      "orderdate": "23.10.2019 10:45:46",
      "packetdate": "23.10.2019 10:46:32",
      "orderamount": "15.00",
      "ordercomment": "",
      "cardexpirationdate": "12/20",
      "ordercurrency": "BYN",
      "recommendation": "",
      "processingname": "Fake",
      "meannumber": "220000****0003",
      "orderstate": "In Process",
      "rate": "1",
      "amount": "15.00",
      "responsecode": "AS110",
      "meantypename": "MasterCard",
      "protocoltypename": "NET",
      "bankcountry": "UNKNOWN",
      "customer":       {
         "lastname": "Testov",
         "firstname": "",
         "middlename": "Testovich",
         "ipaddress": "10.10.10.10",
         "email": "null@assist.by"
      },
      "cardholder": "TEST",
      "approvalcode": "",
      "billnumber": "5161957242913224.1",
      "issuebank": "UNKNOWN",
      "currency": "BYN",
      "ordernumber": "231020191345849_user1",
      "meansubtype": ""
   }
}
```


To implement 3-D Secure authentication methods on merchant pages, you can use a ready-made SDK.

Back on top