1STPAYMENT.NET
GATEWAY INTEGRATION MANUAL

VERSION: 2.55 (3-04-2017)

Table of contents

1 Introduction

1.1 BEFORE YOU START (CHAPTER NOT ONLY FOR IT)

We have created a short checklist that shall be passed before you go live with your Merchant account.

This list will be useful not only for IT, but also for business-departments, as it will help you to prevent problems while processing your transactions. The checklist may be found in chapter 0, we are highly recommending you to read it and check your system against each item listed there before going live.

- Integration samples can be found here (example provided as is, without extra support):

https://github.com/TransactPRO/transactpro-integration-php

- A module for Drupal, developed by 3-rd party company, http://adcisolutions.com/, is available here:

https://www.drupal.org/project/propayment_commerce

Please contact them directly with any questions about module installation, support, additional development, etc.

1.2 ACCOUNT INFORMATION

To create at test or production account, the following data must be provided to the 1stpayments administrators:

1. Table

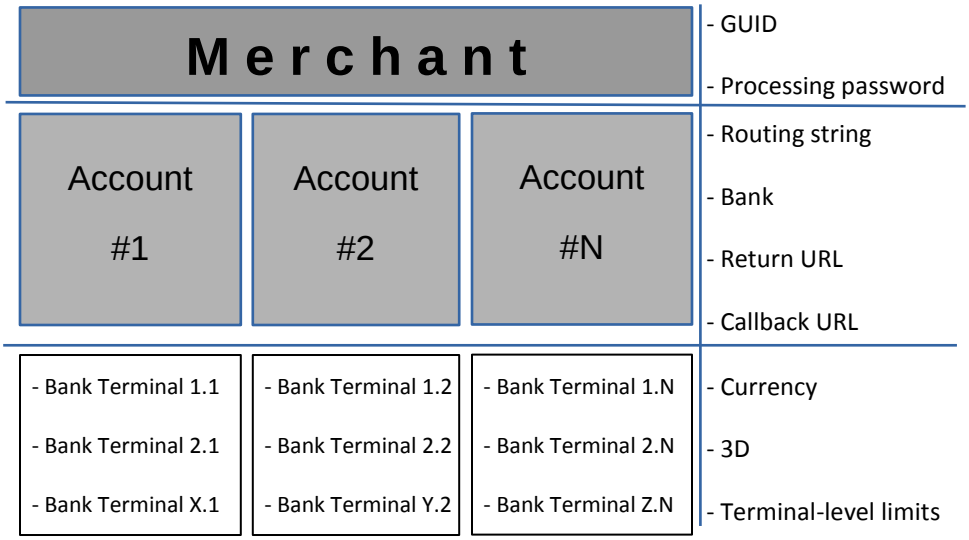
Data item	Description
For test/production environment	
Server IP list	Your server(s) IP(s) that will be added to our firewall. This is a mandatory item.
Default return URL	Gateway will use this URL to return a cardholder. This is mandatory item for 3D accounts and for accounts set to collect card details at gateway side. **Notice:** A URL length can be up to 255 characters.
Contact email	Gateway administrators or account managers will use this email to send notices, updates, and other information, if other contacts won't be accessible. This is a mandatory item.
Bank name	Indicate name of the Bank you have processing agreement with and where you will register terminals for transaction processing (MIDs)
Bank manager name and contact email	Gateway administrators or account managers will use this email to contact your manger in case, if we will need to receive any additional information about your account settings etc.
Your company name	Indicate company that will have processing agreement with the bank.
Your website name	Website that you will integrate with our gateway.
For test environment (Notice: for a test account details, please check chapter 7 of this manual)	
Local IP list	Your local IP(s) that will be used in testing process must be added to our firewall. This is a mandatory item.
Mobile phone number	Gateway administrators or account managers will use this phone number to send you test card details.

After the data from table 1 will be provided, you should get the following data from the 1stpayments support:

2. Table	
Data item	**Description**
Merchant GUID	GUID to identify you, one for all accounts.
Processing password	Password that will be used for the transaction processing, one for all accounts. (Must be passed into gateway API as sha1 hash)
Routing String	In case when more than one account is used, a string that will be used to define which account must be used by the gateway to process your transaction.
Merchant area web login	Login for the gateway front-end.
Merchant area password	Password for the gateway front-end.

Merchant area address is ***https://www2.1stpayments.net/merchantarea.php*** for production account. URL for test account is **https://gw2sandbox.tpro.lv:8443/gw2test/merchantarea.php**

1.3 MERCHANT ACCOUNT STRUCTURE



By default, you will have one account with one 3D terminal, with card details entered at gateway side.

1.4 LAST DOCUMENT UPDATE

The following table provides the document update description:

3. Table

Date	Description
27/06/2016	Chapters 3.4, 3.5 created, chapters 2.1.7, 2.2, 6.2 updated.
28/06/2016	Cosmetic update: style and documentation format.
01/07/2016	Chapter 3.5: update a description of a dynamic descriptor.
12/07/2016	Chapter 2.4.1: added a description for new f_extended values.
19/07/2016	Chapter 2.5.2: added format description for transaction dump response.
20/07/2016	Chapters 2.1.10 and 2.1.11: fixed format for client_birth_date field.
29/07/2016	Chapter 10: added description of Hosted Fields functionality.
04/08/2016	Chapters 2.1.7 and 6.2: a description improved for card storing functionality.
04/08/2016	Chapter 2.5.4: a description of automatic reports message securing for data integrity is added
04/08/2016	Chapter 0: output formats description is added.
22/08/2016	Added format description for API fields.
23/08/2016	Chapter 3.3: changed direct terminal selection description.
29/08/2016	Chapter 2.4.1: fixed Gateway response description.
06/09/2016	Chapter 2.4.1: added StatusID to transaction status format if f_extended is equal to 100.
13/09/2016	Chapter 2.3: added description of “cancel_request”.
22/09/2016	Fixed format description for API fields.
10/10/2016	Chapter 3.4: description for card verification is appended.
31/10/2016	Chapter 10.1: appended description of Hosted Fields. Chapter 10.6: added description of *destroy* method for Hosted Fields. Chapter 2.1.11: appended description of field name_on_card for P2P transactions.
01/11/2016	Chapter 2.4.1: added fields CardIssuerCountry, NameOnCard, CardMasked and ResultCodeStr to transaction status format if f_extended is equal to 100.
09/11/2016	Chapters 2, 3.4 and 6.2: improved description of Gateway supported operations.
02/12/2016	Chapter 4.1: added API method for loading terminal limits.
22/12/2016	Chapter 10.3: changed version of Hosted Fields library.
06/01/2017	Chapter 2.4.1: added ProcessorError to transaction status format if f_extended is equal to 100; appended transaction statuses list.
18/01/2017	Chapter 7: added description for MOTO transactions with previously saved card.
21/02/2017	Chapter 2.4.1: added field CardSaveStatus to transaction status format if f_extended is equal to 100.
23/02/2017	Chapter 2.4.1: fixed transaction statuses. Chapter 2.5.2: fixed transaction and card data save statuses
16/03/2017	Chapter 2.5.2: fixed response fields list; response have new field user_ip.
03/04/2017	Chapter 2.4: added information about account and terminal limits warnings.

Notice: if you have any suggestions or notices about this document, feel free to write us to support email gwsupport@transactpro.lv. Your feedback will help us to improve our documentation.

1.5 FORMAT DESCRIPTION

Fields' format is described in the following format: "<*type*>(<*length*>)".

<*type*> shows the set of allowed characters:

a: alphabetic characters are the upper case letters A through Z; the lower case letters a through z, and the blank (space) character.
h: hexadecimal number.
n: numeric characters are the numbers zero (0) through nine (9).
s: special printable characters are any printable characters that are neither alphabetic nor numeric, have an ASCII hexadecimal value greater than 20, or an EBCDIC hexadecimal value greater than 40. Occurrences of values ASCII 00 – 1F and EBCDIC 00 – 3F are not valid. **Not all special characters are usually enabled. See fields' description for details.**
u: Unicode alphabetic characters.

For string values, a length is shown in the parentheses:

- 1..255: variable length field (in this case, field's length must be from 1 to 255 symbols)
- 15: constant length field (in this case, field's length must be **exactly** 15 symbols)
- If length is not defined, the field accepts only specified values, defined in the field's description.

2 Integration Process

2.1 PROCESS DESCRIPTION

Technically, gateway provides API to make credit card transactions. Credit card information can be collected on merchant side or on gateway side, transaction can be done in *one step (SMS)* or in *two steps (DMS)*, *with 3D or without*. Minimal amount for 3D transaction is 100 minor units (i.e. USD/EUR/GBP/... 1.00) Gateway can do recurrent transactions and MOTO transactions.

From **business side**, you need to ask your account manager what types of transactions are allowed and available for your account. Below you will find schemes for typical transaction flows. Once again, you need to find out which scheme you need to implement – on test server, you can request any type of account, on production – you need to finalize it with your account manager first.

Transact Pro provides test environment not connected to real Visa / MasterCard networks, so you can begin integration before paperwork with the bank will be finished. Test environment details described into corresponding manual chapter.

In this chapter you can find a brief descriptions for a typical transaction methods, details can be founded in corresponding manual chapters.

Notice: If your scheme includes return URL, you can read about return URL customization at chapters **2.4.3** and **2.4.4**.

2.1.1 COMMON PAYMENT PROCESS

Common payment process without 3D Secure is shown on Figure 1. This scheme is common for all financial payments (SMS, DMS, CRD etc.) and is performed in two steps: an initialization step and a payment step.

Real financial operation is performed only when a request is sent to a processor and Gateway has returned a successful result. Any financial manipulations with an original transaction like refunds or DMS cancellation are impossible until the processor has received the original payment.

From the other hand, not paid but initialized request can be cancelled only if a payment was not sent to a processor.

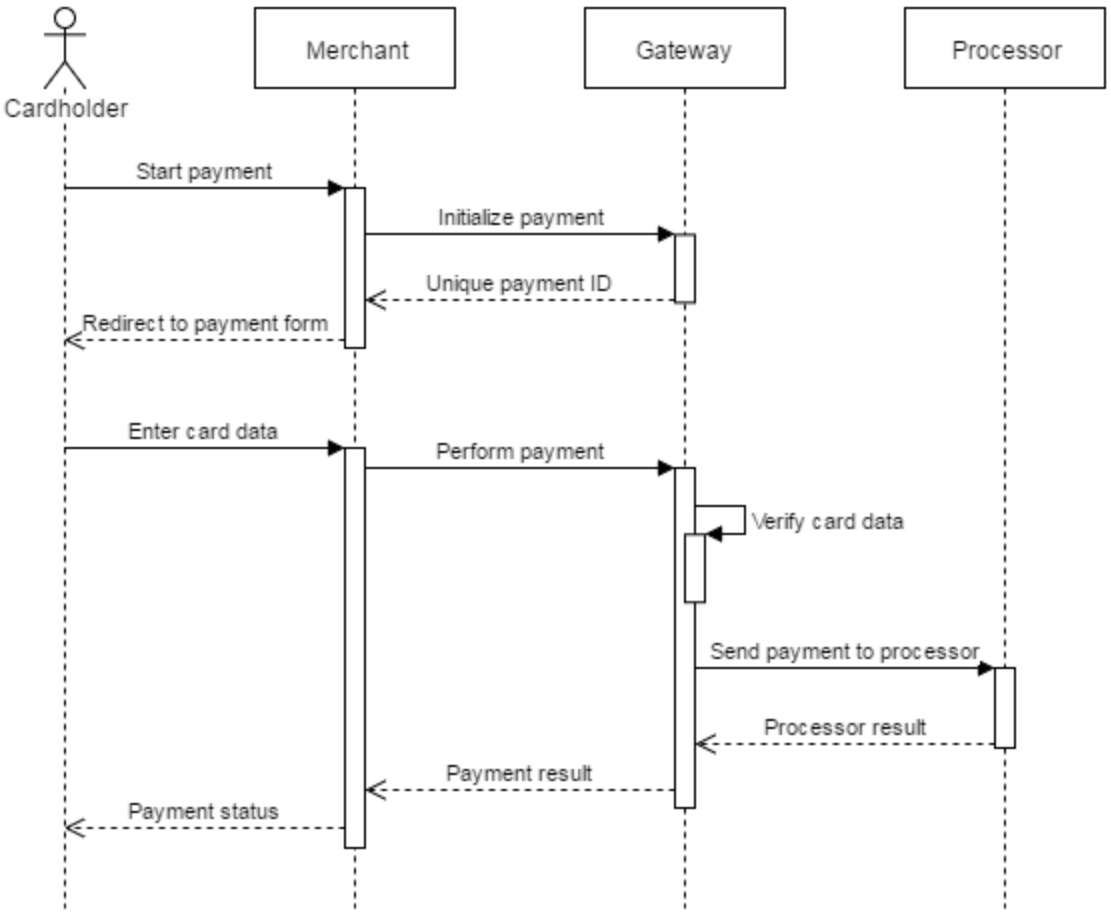


Figure 1 Common payment process without 3D Secure

When card details is entered at Gateway side, merchant should not show his own payment form. A URL to Gateway payment form will be returned. When payment is finished, cardholder is redirected to merchant's page.

A payment with 3D Secure requires a redirect of cardholder browser to the issuer bank ACS (Access Control Server) page. The payment will be successful and will be sent to processor only after cardholder will pass 3D Secure validation and will be return to Gateway.

For a payment scenario with 3D Secure see Figure 2. This payment scenario is common for SMS and DMS payments. If a card is not enrolled in 3D Secure, after an MPI (Merchant Plug-In) call, a payment process will be the same as for a payment without 3D Secure.

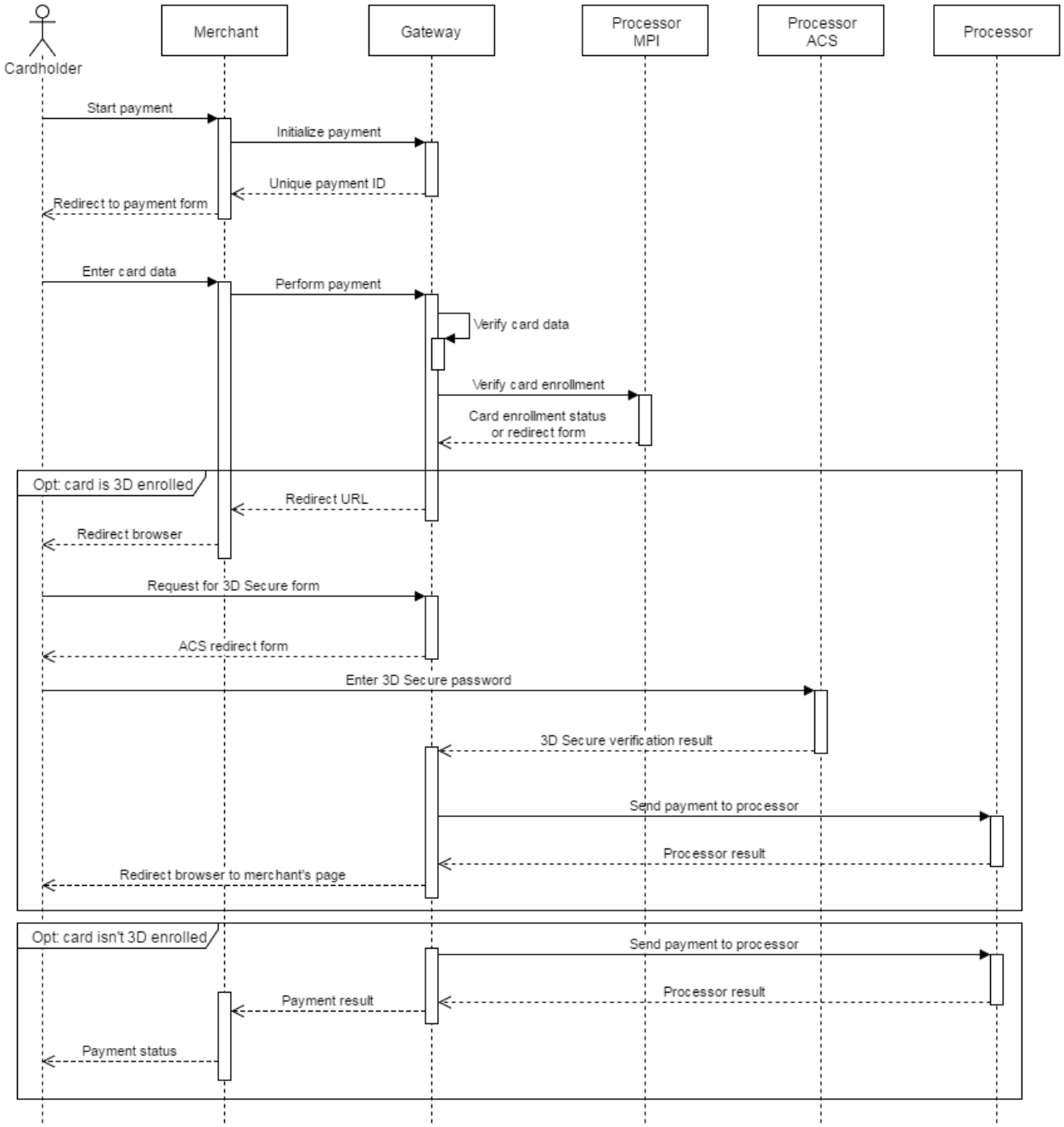


Figure 2 Common payment process with 3D Secure

When card details is entered at Gateway side, merchant should not show his own payment form. A URL to Gateway payment form will be returned. Gateway will perform redirect to processor ACS by itself if card is 3D Secure enrolled. When payment is finished, cardholder is redirected to merchant's page.

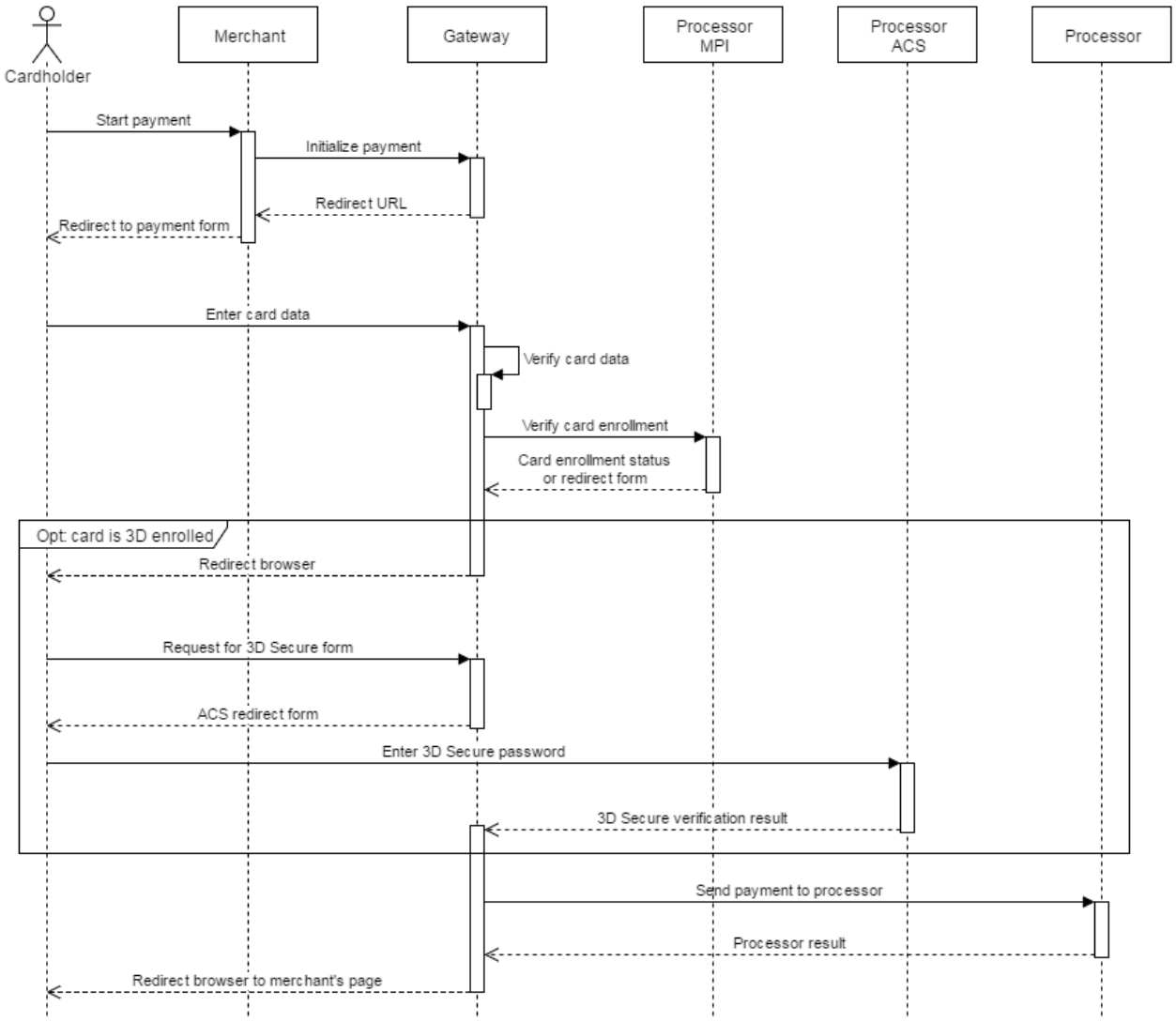


Figure 3 Payment with 3D Secure (card details is entered at Gateway side)

2.1.2 SMS (SINGLE MESSAGE) TRANSACTION, CARD DETAILS ENTERED AT MERCHANT SIDE.

1. Initialize SMS transaction. You will get initial transaction ID, if all is ok.

2. Send card details to make a charge. If your bank terminal is 3D, as far as a card is 3D enrolled – you will get redirect link. If card or terminal are not 3D – you will get result message. (In non-3D case charge is done, steps 3 and 4 are for 3D).

3. If you've got 3D link, you should redirect cardholder to this link. Cardholder will be redirected back to your Return URL (if he won't close browser).

4. You will need to make server to server request to the gateway, to get transaction result.

2.1.3 SMS (SINGLE MESSAGE) TRANSACTION, CARD DETAILS ENTERED AT GATEWAY SIDE

1. Initialize SMS transaction. You will get initial transaction ID and redirect link, if all is ok.

2. Redirect cardholder to the link received at step 1. At this link, he will enter his card details (this page can be customized, if you need).

3. After cardholder fills the data and submits the form, he will be redirected to 3D, if required, and back to your return URL.

4. You will need to make server to server request to the gateway, to get transaction result.

2.1.4 DMS (DOUBLE MESSAGE) TRANSACTION, CARD DETAILS ENTERED AT MERCHANT SIDE

1. Initialize DMS transaction. (See 3.2 for more details). You will get initial transaction ID, if all is ok.

2. Send card details to make a **hold**. If your bank terminal is 3D, as far as a card is 3D enrolled – you will get redirect link. If card or terminal are not 3D – you will get result message.

3. In case of 3D transaction, redirect the cardholder and check transaction status, as it's described for SMS 3D.

4. If hold was successful, compete the transaction in 28 days as described into corresponding manual chapter (see Figure 4).

Notice: before step 4, amount is only held at cardholder's bank account. You need to charge it to complete transaction.

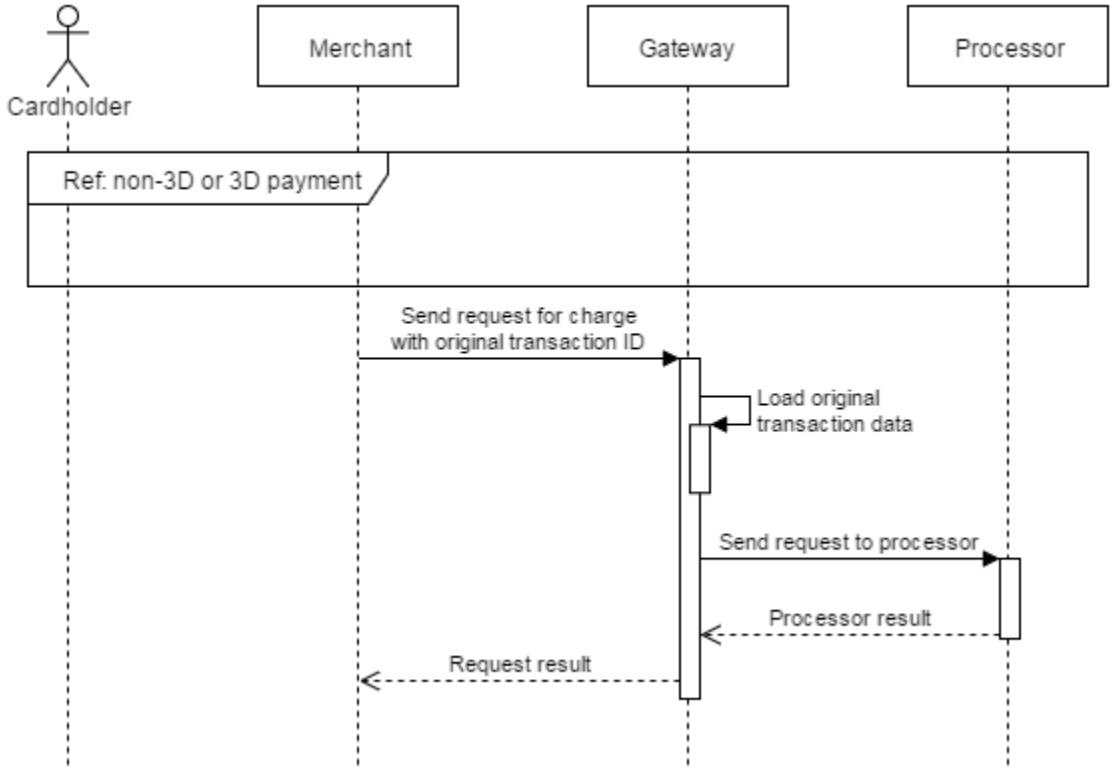


Figure 4 DMS scenario

2.1.5 DMS (DOUBLE MESSAGE) TRANSACTION, CARD DETAILS ENTERED AT GATEWAY SIDE

1. Initialize DMS transaction. (See 3.2 for more details). You will get initial transaction ID and a re-direct link, if all is ok.

2. Redirect cardholder to the link received at step 1. At this link, he will enter his card details (this page can be customized, if you need).

3. After cardholder fills the data and submits the form, he will be redirected to 3D, if required, and back to your return URL.

4. You will need to make server to server request to the gateway, to get transaction hold result.

5. If hold was successful, in 28 days, compete the transaction as described into corresponding manual chapter.

2.1.6 CHANGES FOR MOTO TRANSACTIONS

MOTO transaction (Mail Order / Telephone Order) is a type of transaction, where card can be processed without CVV code. You can do MOTO as SMS (DMS for some acquirers, ask support for the details), with card details entered at your side.

Notice: you need to get approval for MOTO transactions from your account manager. If you use 3D terminal, additional non-3D terminal will be required.

Notice: If account is MOTO, it will ignore CVV sent with transaction details, and proceed transaction as MOTO. You should divide transactions flow at your side, and send MOTO transactions to MOTO ac-count, and non-MOTO transactions to non-MOTO account.

2.1.7 RECURRENT TRANSACTIONS

Recurrent transaction (Subscription/Re-bill Order) is a type of transaction, where transaction can be processed, using card details that have been previously saved on gateway side (see Figure 5). You can do Recurrent as SMS.

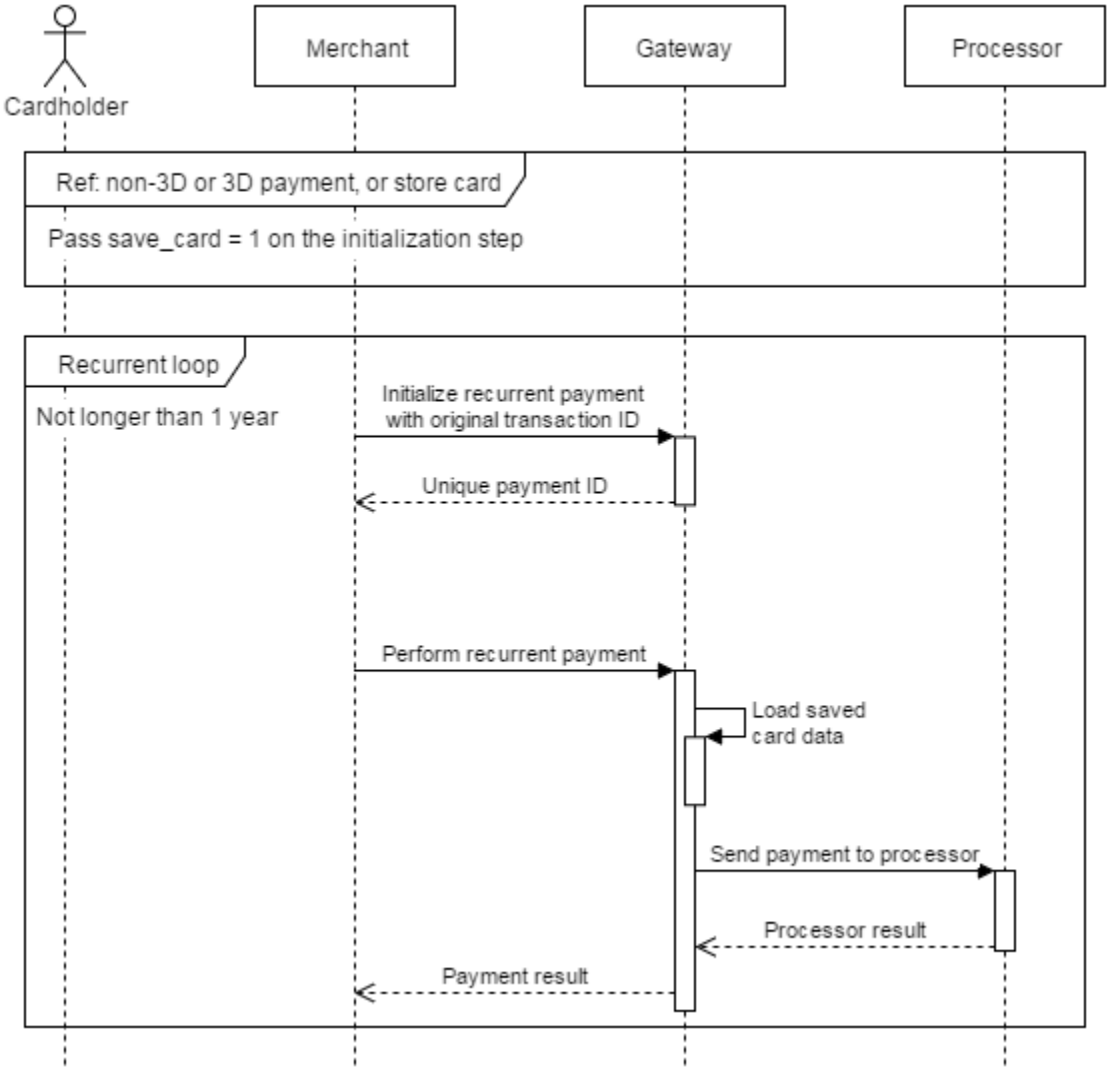


Figure 5 Recurrent payments scenario

Notice: you need to get approval for recurrent transactions from your account manager. If you use 3D terminal, additional non-3D terminal will be required.

You can create recurrent transaction in two ways.

Usual method:

1. Initialize “plain” transaction with recurrent flag (*see 6.1 manual chapter*).
2. Complete “plain” transaction.
3. Do recurrent transaction sending required requests described in corresponding manual chapter.

Additional method:

1. Initialize store card request for a required operation (*see 6.2 manual chapter*).
2. Complete request with "**store_card**" method.
3. Do recurrent transaction sending required requests described in corresponding manual chapter.

Do not forget about different routing strings for steps 1 and 3. Once again, ONLY transactions which are recurrents (i.e. you've done transaction thru USUAL account with passed flag to save card details, and now you need to make recurrent transaction for this transaction).

2.1.8 CREDIT TRANSACTIONS

Credit transaction is a type of transaction, where funds are transferred to cardholder's account, using card details as a token. This operation is available for both MC and Visa. Credit transaction differs from usual SMS non-3D MOTO transaction only by its initial step: you have to **init credit transaction** with request to:

https://www2.1stpayments.net/gwprocessor2.php?a=init_credit

All other fields follow same policy as plain "**init**" step (see 2.2 for the details) Second step (where the card details sent) should be sent to:

https://www2.1stpayments.net/gwprocessor2.php?a=do_credit

Fields are the same as for a "**charge**" step for a MOTO transaction (see 2.4 for the details), except parameter "**expire**" - this parameter is not mandatory. Reply is also the same as for the "**charge**".

Notice: This operation cannot be refunded. Money will be available for a cardholder with delay which depends on issuer bank clearing settings.

2.1.9 RECURRENT CREDIT TRANSACTIONS

Recurrent credit transaction is credit transaction using card details from other succeed credit transaction. Work flow is the same as for recurrent transactions with difference that parent transaction should be credit.

2.1.10 P2P TRANSACTIONS

P2P function in our system is to transfer funds to cardholder's account.

Notice: In current version two-cards solution (when first card is debited first for transaction amount) is not sup-ported.

P2P transaction differs from CRD transaction by its initial step and some additional details:

4. Table

Field	Format	Description
cardname	ans(1..255)	Recipient cardholder name, as printed on a card
recipient_name	an(1..45)	Recipient name
client_birth_date	n(8)	Sender birth date, MMDDYYYY format (for example: 06161981). In case of a legal person – company registration date.

Also, some fields meaning is changed: you have to pass sender address into address-related fields, and, sender name into name on card field.

You have to init P2P transaction with request to:

https://www2.1stpayments.net/gwprocessor2.php?a=init_p2p

All other fields follow same policy as plain "**init**" step (see 2.2 for the details)

Second step (where the card details sent) should be sent to

https://www2.1stpayments.net/gwprocessor2.php?a=do_p2p

On a second step, you have to send only card number field, using **cc_2 field name**. Rename is done as its planned to support two-cards transactions in future.

5. Table		
Field	**Format**	**Description**
cc_2	n(13..19)	Recipient card number
init_transaction_id	h(40)	init_transaction_id received for this recurrent transaction
f_extended	n	Return extended charge details (optional)
expire	ns(5)	Card expiration date, in format MM/YY (optional)

Gateway reply is the same as for the "**charge**" method.

Notice: This operation cannot be refunded. Money will be available for a cardholder with delay which depends on issuer bank clearing settings.

2.1.11 P2P AND CRD TRANSACTIONS REQUIREMENTS ON INITIALIZATION REQUEST URL

6. Table		
Environment	**CRD**	**P2P**
Production	*https://www2.1stpayments.net/gwprocessor2.php?a=**init_credit***	*https://www2.1stpayments.net/gwprocessor2.php?a=**init_p2p***
Test	*https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=**init_credit***	*https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=**init_p2p***

POST values on initialization request

A Server to server POST request must contain values listed in the table below.

7. Table				
Filed	**Format**	**Description**	**CRD (init_credit)**	**P2P (init_p2p)**
guid	ans(19)	Your merchant GUID.	mandatory	mandatory
pwd	h(40)	SHA1 hash of your processing password.	mandatory	mandatory
cardname	ans(1..255)	Recipient cardholder name, as printed on a card	not used	mandatory
recipient_name	ans(1..45)	Recipient name. Allowed symbols: [' . -].	not used	mandatory
client_birth_date	n(8)	Sender's birth date, MMDDYYYY format (for example: 06161981). In case	not used	mandatory

		of a legal person – company registration date.		
rs	an(1..12)	Your routing string.	mandatory	mandatory
merchant_transaction_id	ans(5..50)	Your transaction ID, must be unique for every transaction you submit to the gateway. The transaction ID must be from 5 to 50 characters.	mandatory	mandatory
user_ip	ns(7..15)	Sender's IP, as string (AA.BB.CC.DD).	optional (ask account manager to set mandatory or not)	mandatory
description	uns(5..255)	Order items description, from 5 to 255 characters (Example: SDHC Memory card x 2, AAA battery pack x 1).	optional (ask account manager to set mandatory or not)	optional (ask account manager to set mandatory or not)
amount	n	Transaction amount, in **MINOR** units (i.e. 2150 for $21.50 transaction).	mandatory	mandatory
currency	a(3)	Transaction currency, ISO 4217 3-character code (USD, EUR, CHF, and other).	mandatory	mandatory
name_on_card	as(2..100)	Sender's name. In case of a legal person – registered company name. Allowed symbols: [' . -]. A space is used as a delimiter for name and surname. **Name may not be shorter than 2 English alphabet letters.**	optional (ask account manager to set mandatory or not)	mandatory Maximum length for **VISA** cards: **25 symbols**. Maximum length for **MasterCard** cards: **30 symbols**.
street	ans(2..50)	Sender's address – street. Allowed symbols: [' . -) (:].	optional (ask account manager to set mandatory or not)	mandatory Maximum length for **VISA** cards: **30 symbols**. Maximum length for **MasterCard** cards: **35 symbols**.
zip	ans(2..15)	Sender's address – ZIP. Allowed symbols: [' . -].	optional (ask account manager to set mandatory or not)	mandatory Maximum length: **10 symbols**.
city	as(2..25)	Sender's address – City. Allowed symbols: [' . -].	optional (ask account manager to set mandatory or not)	mandatory
country	a(2)	Sender's address – country, 2-letter ISO 3166-1-Alpha 2 code.	optional (ask account manager to set mandatory or not)	mandatory
state	ans(2..20)	Sender's address – state	optional (ask	optional (ask

		(send NA if you don't have state information).	account manager to set mandatory or not)	account manager to set mandatory or not)
email	ans(1..100)	Sender's address – email	optional (ask account manager to set mandatory or not)	optional (ask account manager to set mandatory or not)
phone	ns(5..25)	Sender's phone number.	optional (ask account manager to set mandatory or not)	optional (ask account manager to set mandatory or not)
card_bin	n(6)	Cardholder card BIN (first 6 characters of CC number). - not required if card data collected at gateway side.	optional (ask account manager to set mandatory or not)	optional (ask account manager to set mandatory or not)
bin_name	uns(3..50)	Cardholder bank name (non-mandatory).	optional (ask account manager to set mandatory or not)	optional (ask account manager to set mandatory or not)
bin_phone	ns(3..25)	Cardholder bank phone given on a back side of used card (non-mandatory).	optional (ask account manager to set mandatory or not)	optional (ask account manager to set mandatory or not)
merchant_site_url	ans(1..255)	Purchase site URL.	optional (ask account manager to set mandatory or not)	optional (ask account manager to set mandatory or not)

2.1.12 P2P AND CRD TRANSACTIONS REQUIREMENTS ON FINAL REQUEST URL

8. Table

Environment	CRD	P2P
Production	*https://www2.1stpayments.net/qwprocessor2.php?a=**do_credit***	*https://www2.1stpayments.net/qwprocessor2.php?a=**do_p2p***
Test	*https://qw2sandbox.tpro.lv:8443/qw2test/qwprocessor2.php?a=**do_credit***	*https://qw2sandbox.tpro.lv:8443/qw2test/qwprocessor2.php?a=**do_p2p***

POST values on final step

9. Table

Filed	Format	Description	CRD (do_credit)	P2P (do_p2p)
guid	ans(19)	Your merchant GUID.	mandatory	mandatory
pwd	h(40)	SHA1 hash of your processing password.	mandatory	mandatory
init_transaction_id	h(40)	init_transaction_id received for this recurrent transaction	mandatory	mandatory
cc	n(13..19)	Card number. Must be defined as a string without spaces or other dividers.	mandatory	not used
cc_2	n(13..19)	Recipient card number	not used	mandatory
expire	ns(5)	Card expiration date, in format MM/YY.	optional	not used
expire2	ns(5)	Card expiration date, in format MM/YY.	not used	optional
f_extended	n	Return extended charge details	optional	optional

2.2 INITIALIZING A TRANSACTION

URL for production: ***https://www2.1stpayments.net/gwprocessor2.php?a=init*** *(init_dms, for DMS transactions)*

URL for test: ***https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=init*** *(init_dms, for DMS transactions)*

A Server to server POST request must contain values listed in the table below. All fields are mandatory, if other isn't stated.

10. Table

Field	Format	Description
guid	ans(19)	Your merchant GUID.
pwd	h(40)	SHA1 hash of your processing password.
rs	an(1..12)	Your routing string.
merchant_transaction_id	ans(5..50)	Your transaction ID, must be unique for every transaction you submit to the gateway. The transaction ID must be from 5 to 50 characters.
user_ip	ns(7..15)	Cardholder's IP, as string (AA.BB.CC.DD).
description	uns(5..255)	Order items description, from 5 to 255 characters (Example: SDHC Memory card x 2, AAA battery pack x 1).
amount	n	Transaction amount, in **MINOR** units (i.e. 2150 for $21.50 transaction). **Notice:** check JPY exception notice below!
currency	a(3)	Transaction currency, ISO 4217 3-character code (USD, EUR, CHF, and other). **Notice:** check JPY exception notice below!
name_on_card	ans(2..100)	Cardholder name, as printed on a card (pass client name if card data collected at gateway side)
street	ans(2..50)	Cardholder address – street. (min 2 symbols)
zip	ans(2..15)	Cardholder address – ZIP. (min 2 symbols)
city	as(2..25)	Cardholder address – City. (min 2 symbols)
country	a(2)	Cardholder address – country, 2-letter ISO 3166-1-Alpha 2 code.
state	ans(2..20)	Cardholder address – state (send NA if you don't have state information).
email	ans(1..100)	Cardholder address – email
phone	ns(5..25)	Cardholder phone number (min. 5 symbols).
card_bin	n(6)	Cardholder card BIN (first 6 characters of CC number). - not required if card data collected at gateway side.
bin_name	uns(3..50)	Cardholder bank name (non-mandatory).
bin_phone	ns(3..25)	Cardholder bank phone given on a back side of used card (non-mandatory).
merchant_site_url	ans(1..255)	Purchase site URL.
merchant_referring_name	ans(1..21)	Must not be send by default. See chapter 3.5 for description if you need to use it.

Additionally, your account manager can ask you to send the shipping information.
For example, it can be required for the additional anti-fraud module. If it is required, pass the shipping address into the following additional fields:

- *shipping_name*
- *shipping_phone*
- *shipping_email*
- *shipping_state*
- *shipping_country*
- *shipping_city*
- *shipping_zip*
- *shipping_street*

Fields length and the data type is the same as for the cardholder address fields. For more information, see the table number 4.

JPY notice: JPY currency don't have minor units, but, for some acquirers you have to send it into minor units anyway. i.e., to charge 105 JPY, you have to send the amount as “10500” anyway. If you'll try to charge minor JPY units (i.e. you'll send “90” or “12510” or “1114” as amount, system will cast this amount to integer, and in case of amount less than 1 JPY, this amount will be equal to zero!). Please also notice that this exception (ab-sense of minor units) may affect some custom-generated reports.

Please consult our technical support before sending JPY transactions.

Pay attention to the following information:

11. Table

Nr	Description
1	Before sending data to the gateway, it must be checked on your side.
2	Some fields can be non-mandatory, due to the anti-fraud module settings for your account. For more information, contact your account manager.
3	To get the faster response from the gateway administrators about the error reasons, please send the whole error output you get as is.
4	No HTML tags are allowed in the filed values, all the HTML code will be removed.
5	The received data will be url-decoded before the processing.
6	If you'll send a transaction with a merchant transaction ID, which already exists for your account, the gateway will reject it. It means that you should avoid multiple submits of the init data.

The following table presents an error message and successful transaction initialization samples:

12. Table	
Name	**Sample**
Error message	ERROR:1244819489:Init failed: bad access data
Successful transaction initialization	OK:74ca533fbd5bd9ee09057f280a46cf97aa76ebb3
Successful transaction initialization sample with redirection link to enter card data on the gateway side	OK:92c5b5acd6e916cc957931b98adedbfe72497153~RedirectOnsite:https://www2.1stpayments.net/xxdata.php?tid=27fc1f2f063d8416d0fbca3ef8809d15

Notice: Transaction ID and temporary id used into redirect link are completely different. Second one used only for the redirection link, first one is the transaction ID you should keep in your records to refer to this transaction later.

2.3 CANCELLING A TRANSACTION

Just after a transaction was initialized or cardholder was redirected to a card data input form, but the form was not submitted, the transaction can be cancelled (see Figure 6).

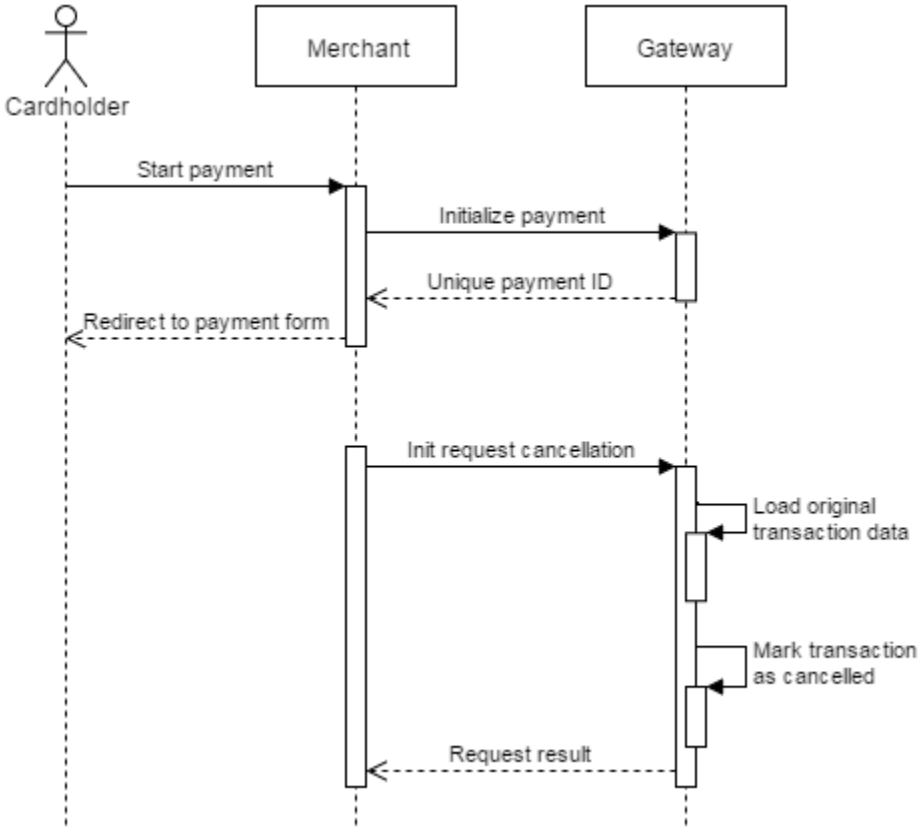


Figure 6 Cancel request scenario

URL for production: ***https://www2.1stpayments.net/gwprocessor2.php?a=cancel_request***

URL for test: ***https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=cancel_request***

A server to server POST request must contain values listed in the table below. All fields are mandatory, if other isn't stated.

13. Table		
Field	**Format**	**Description**
init_transaction_id	h(40)	Gateway transaction ID of a transaction that should be canceled.

The following table presents an output examples:

14. Table	
Name	**Sample**
Error message	ERROR:1244819489:{CODE:0104}Cancel request failed
Successful transaction cancellation	Request cancelled

It is not possible to complete the transaction if it was cancelled. From the other hand, it is not possible to cancel a transaction if any financial operation (charge, make hold etc.) was already performed.

2.4 COMPLETING A TRANSACTION

Notice: You can skip chapter 2.4.1 and 2.4.2 if your account set to collect card details at gateway side. This chapter describes actions you need to implement to collect card details at your side.

URL for production: ***https://www2.1stpayments.net/gwprocessor2.php?a=charge***

URL for test: ***https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=charge***

A server to server POST request (see step 2 from section 2.1 of this manual) must contain values listed in the table below. All fields are mandatory, if other isn't stated.

15. Table		
Field	**Format**	**Description**
f_extended	n	Return extended charge details. This is an optional field.
init_transaction_id	h(40)	Gateway transaction ID.
cc	n(13..19)	Card number. Must be defined as a string without spaces or other dividers.
cvv	n(3..4)	Card CVV number.
expire	ns(5)	Card expiration date, in format MM/YY.
merchant_referring_url	ans(1..255)	Payment referring page URL, where cardholder enters PAN data. Could be optional.

2.4.1 NON-3D CARDS

In case of a non-3D card or non-3D bank terminal, you will get the following output:

`Status:%STATUS%`

Where **%STATUS%** can be the following:

- Success
- Failed
- Pending

You will also receive a transaction status message to your callback URL, if it is set. *(See 2.5.3 section of this manual for the details)* The pending status is very rare, usually it means that there is some problems connected to VISA or MasterCard card issuer bank, and in most cases such transactions have the failed final status.

Warning: if an error occurred on the Gateway side, the **`Status`** will not be present in the response and an error will be returned in the following format:

"`ERROR:%timestamp%:%error description%`"

where `%timestamp%` is a current timestamp and `%error description%` describes an error.

If the **`f_extended`** value is set and there are no errors on the Gateway side, gateway will return the following string:

16. Table

f_extended value	Returned result string
1	ID:%init_transaction_id%~Status:%STATUS%~MerchantID:%merchant_transaction_id%~Terminal:%Terminal Name%~ResultCode:%processor_result_code%
2	ID:%init_transaction_id%~Status:%STATUS%~MerchantID:%merchant_transaction_id%~Terminal:%Terminal Name%~ResultCode:%processor_result_code%~ApprovalCode:%processor_approval_code%
3	ID:%init_transaction_id%~Status:%STATUS%~MerchantID:%merchant_transaction_id%~Terminal:%Terminal Name%~ResultCodeString:%processor_result_code_string%
4	ID:%init_transaction_id%~Status:%STATUS%~MerchantID:%merchant_transaction_id%~Terminal:%Terminal Name%~ResultCode:%processor_result_code%~ApprovalCode:%processor_approval_code%~CardIssuerCountry:%card issuer country%
5	ID:%init_transaction_id%~Status:%STATUS%~MerchantID:%merchant_transaction_id%~Terminal:%Terminal Name%~ResultCode:%processor_result_code%~ApprovalCode:%processor_approval_code%~NameOnCard:%name_on_card%~CardMasked:%card_masked%
6	ID:%init_transaction_id%~Status:%STATUS%~MerchantID:%merchant_transaction_id%~Terminal:%Terminal Name%~ResultCode:%processor_result_code%~ApprovalCode:%processor_approval_code%~dt_created:%dt_created%~3d:%f_3d_status%

7	ID:%init_transaction_id%~Status:%STATUS%~MerchantID:%merchant_transaction_id%~Terminal:%Terminal Name%~ResultCode:%processor_result_code%~ApprovalCode:%processor_approval_code%~CardIssuerCountry:%card issuer country%~NameOnCard:%name_on_card%~CardMasked:%card_masked%~3d:%f_3d_status%
8	ID:%init_transaction_id%~Status:%STATUS%~MerchantID:%merchant_transaction_id%~Terminal:%Terminal Name%~ResultCode:%processor_result_code%~ExtendedErrorCode:%extended_error_code%
9	ID:%init_transaction_id%~Status:%STATUS%~MerchantID:%merchant_transaction_id%~Terminal:%Terminal Name%~ResultCode:%processor_result_code%~ExtendedErrorCode:%extended_error_code%~TerminalWarning:%terminal warning%
100	ID:%init_transaction_id%~Status:%STATUS%~MerchantID:%merchant_transaction_id%~Terminal:%Terminal Name%~ResultCode:%processor_result_code%~ExtendedErrorCode:%extended_error_code%~MerchantReferringName:%merchant_referring_name%~DynamicDescriptor:%dynamic descriptor%~StatusID:%transaction status ID%~CardIssuerCountry:%card issuer country%~NameOnCard:%name_on_card%~CardMasked:%card_masked%~ResultCodeString:%processor_result_code_string%~ProcessorError:%processor error%~CardSaveStatus:%card save status%

Notice: The card issuer bank country can be identified only if the anti-fraud module is enabled for your account. To be sure the anti-fraud module is enabled for your account, please contact your account manager.

Notice: As the BIN ranges are the subject to change without prior notice, in rare cases BIN country can be detected wrongly.

Notice: At this step, the gateway can output the warning messages about the account limits under key "Warning". "Warning" can be outputted for any **f_extended** value, including the default one.

Example: *Status:Success~Warning: account transactions amount limit will be reached soon*

Warning: A format for *f_extended* 100 is generic and will be appended without any additional warning.

The following table describes the return fields:

17. Table

Field	Description
%init_transaction_id%	Gateway transaction ID.
%STATUS%	Transaction status, as described above in this section.
%merchant_transaction_id%	Merchant transaction ID.
%Terminal Name%	Descriptor a cardholder will see in a bank statement.
%processor_result_code%	3-character code of a transaction charge details.
%processor_approval_code%	Approval code a cardholder will see in a bank statement.
%processor_result_code_string%	3-character code of transaction charge details converted into the string with it's description (In this case, return can be longer than 256 bytes).
%card issuer country%	Country of the card issuer bank, two-letter ISO-3166 code. XX or an empty string will be passed if the issuer country was not found.
%name_on_card%	Name, passed as name printed on card
%card_masked%	Card number, in 4111********1111 format
%dt_created%	Transaction creation time as UNIX time stamp (for example 1377614290)

%f_3d_status%	Transaction 3D status (1- not 3D, 2 – 3D, 0 – not defined)
%extended_error_code%	Transaction charge details converted into the string with it's description, given in cases if %processor_result_code% is not set (In this case, return can be longer than 256 bytes).
%terminal warning%	A warning that terminal limits will be reached soon
%merchant_referring_name%	A merchant_referring_name parameter's value from a transaction request, if a dynamic descriptor was used.
%dynamic descriptor%	Full formed dynamic descriptor, if merchant_referring_name parameter was passed into transaction
%transaction status ID%	Numeric representation of transaction status. Available values: • 1: Initialized • 2: Sent to processor • 3: Amount hold OK • 4: Amount hold failed • 5: SMS charge failed • 6: DMS charge failed • 7: Success • 8: Expired • 9: Hold expired • 11: Refund failed • 12: Refund pending • 13: Refund success • 14: Cardholder enters card data • 15: DMS canceled OK • 16: DMS cancel failed • 17: Reversed • 18: Credit failed • 19: Credit success • 20: P2P success • 21: P2P failed • 22: Card data store for SMS success • 23: Card data store for SMS failed • 24: Card data store for CRD success • 25: Card data store for CRD failed • 26: Card data store for P2P success • 27: Card data store for P2P failed • 28: Transaction initialization was cancelled • 29: Transaction was automatically cancelled
%processor error%	Contains an error, obtained from a processor if transaction has unsuccessful status. Will be empty for successful transaction.
%card save status%	Status of card data in the TransactPro internal system for further operations without card data input from a card holder. Possible values: • 0: Not to be saved • 1: Needs to be saved • 2: Successfully saved • 3: Failed to save card • 4: Failed to save card • 5: Saved data will be used • 6: Needs to be saved on a merchant side • 7: Successfully saved on a merchant side • 8: Failed to save card on a merchant side

	• 9: Needs to be saved for MOTO • 10: Successfully saved for MOTO • 11: Saved data will be used for MOTO

Notice: Please, ask support for dynamic description functionality, if it is required.

2.4.2 3D CARDS

In case of a 3D-card and corresponding settings of your account, you will get the following output:

`Redirect: %URL%`

Where **%URL%** is the URL which the cardholder must be redirected to. When cardholder is returned to your Default Return URL, you will be able to request transaction status from the gateway. (*See chapter 2.5 for the details)*.

Notice: At this step, the gateway can output the warning messages about the account and terminal limits, etc. These messages will be outputted after the redirect link / status string under keys "Warning" or "TerminalWarning", divider is **'~'**

Example 1: *`Redirect:%URL%~Warning: account transactions amount limit will be reached soon`*

Example 2: *`Redirect:%URL%~TerminalWarning: terminal transactions amount limit will be reached soon`*

Notice: If the cardholder clicks **Back** in his browser after return to the merchant site after 3D-transaction, gateway will redirect him back to the return URL. If the gateway will be unable to detect client merchant, for example, if the cardholder will click **Back** twice, and then will click **Forward**, the corresponding error message from the gateway will be shown to the client on the gateway page without redirections.

2.4.3 RETURN URL – ADDING MERCHANT TRANSACTION ID

To get information about the order ID of the returned cardholder, you can customize the return URL.

To perform a customization, add the following string to your return URL:
`merchant_transaction_id=ZZZZZZZ` (7 'Z' characters).

This string will be automatically converted to the *merchant_transaction_id=%*merchant transaction ID here% string.

Example:

If your return URL is the following:

http://www.yoursiteurlhere.com/processed.php?merchant_transaction_id=ZZZZZZZ

And the merchant transaction ID for this transaction is 24834933, the cardholder will be returned to the following URL:

http://www.yoursiteurlhere.com/processed.php?merchant_transaction_id=24834933

Notice: No information about a transaction status will be passed with the return URL. To get the transaction status, send a server-to-server request about the transaction status to the gateway. *(Or use a callback URL, see section 2.5.3 for the details)*

Notice: In case when a cardholder enters the card data on the gateway side, you should redirect him to the link received at the step 1, and then get the transaction status – when the cardholder is returned to your site.

2.4.4 RETURN URL AND CALLBACK URL – CUSTOMIZATION

You can set your own return and callback URL instead of default one. To do this, at **init step**, pass additional fields:

18. Table		
Field name	**Format**	**Value**
custom_return_url	ans(1..255)	Custom return URL
custom_callback_url	ans(1..255)	Custom callback URL

Notice: custom return URL domain should be equal to domain into default return URL. For example, if your de-fault return URL is "**www.example.com/return**", you can override it with custom return url "**www.example.com/return2**", but not with "**www.example2.com/return**"

Notice: custom ports are not allowed, due to security reasons.

2.5 REQUESTING TRANSACTION STATUS

2.5.1 TRANSACTION STATUS REQUEST

URL for production: ***https://www2.1stpayments.net/gwprocessor2.php?a=status_request***

URL for test: ***https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=status_request***

To initiate a transaction status report, post the following POST request to the gateway:

19. Table		
Field	**Format**	**Value**
request_type	as	'transaction_status'
init_transaction_id	h(40)	Gateway Transaction ID
f_extended	n	Return extended charge details, see section 2.4 of this manual for more details (optional)
guid	ans(19)	Your GUID
pwd	h(40)	SHA1 hash of your processing password

If the **f_extended** field is set, the gateway will return the transaction status and details as described in the *Table 16.*

Notice: you can replace "**init_transaction_id**" field with "**merchant_transaction_id**" field, and pass corresponding transaction value. In this case, gateway will return status for transaction with corresponding merchant transaction ID. Please note, that if you use this feature, you must have unique merchant transaction ID's for all GUID space.

2.5.2 TRANSACTIONDUMP REQUEST

This request allows you to get transaction data, as far as transactions related to transaction you requested. Related transactions can be refunds, recurrent transactions, etc. To make a request, you need to send a request identical to transaction status request *(described in chapter 2.5.1 of this manual)*, setting "**request_type**" field to "**transaction_dump**". (**f_extended** field will be ignored)

As a result, (if requested transaction exists) you will get XML with the following structure:

```
<frontoffice>
  <requested>
    <dt_created>DT_CREATED</dt_created>
    <amount>AMOUNT</amount>
    <currency>CURRENCY</currency>
    <merchant_transaction_id>MERCHANT_TRANSACTION_ID</merchant_transaction_id>
    <init_transaction_id>INIT_TRANSACTION_ID</init_transaction_id>
    <processor_approval_code>APPROVAL_CODE</processor_approval_code>
    <user_ip>USER_IP</user_ip>
    <name_on_card>NAME_ON_CARD</name_on_card>
    <processor_result_code>RESULT_CODE</processor_result_code>
    <cc_num>CARD_NUMBER_MASKED</cc_num>
    <extended_error_code>EXTENDED_ERROR_CODE</extended_error_code>
    <f_3d_status>F_3D_STATUS</f_3d_status>
    <terminal_name>TERMINAL_DESCRIPTOR</terminal_name>
    <sli>SLI</sli>
    <eci>ECI</eci>
    <issuer_country>ISSUER_COUNTRY</issuer_country>
    <card_bin>CARD_BIN</card_bin>
    <routing_string>ROUTING_STRING</routing_string>
  </requested>
  <relation_source>
    <dt_created>DT_CREATED</dt_created>
    <amount>AMOUNT</amount>
    <currency>CURRENCY</currency>
    <f_status_str>STATUS AS STRING</f_status_str>
    <f_relation_type_str>RELATION TYPE</f_relation_type_str>
  </relation_source>
  <relation_target>
    <dt_created>DT_CREATED</dt_created>
    <amount>AMOUNT</amount>
    <currency>CURRENCY</currency>
    <f_status_str>STATUS AS STRING</f_status_str>
    <f_relation_type_str>RELATION TYPE</f_relation_type_str>
  </relation_target>
</frontoffice>
```

Response fields are listed in the table 20.

20. Table

Field	Sub-field	Description
requested		Information about transaction requested
	dt_created	Transaction creation timestamp
	amount	Transaction amount
	currency	Transaction currency (currency code by ISO 4217)
	merchant_transaction_id	Parameter merchant_transaction_id from request
	init_transaction_id	Transaction initialization ID, generated by Gateway
	processor_transaction_id	Processor's transaction ID
	processor_approval_code	Transaction approval code, obtained from a processor
	user_ip	Parameter user_ip from request

	name_on_card	Name on card
	card_bin	Parameter card_bin from request
	processor_result_code	Transaction action code, obtained from a processor
	cc_num	Masked PAN (format: NNNN********NNNN, where N is a digit)
	extended_error_code	Decline reason
	f_3d_status	Transaction 3D status: • 0: undefined • 1: not 3D transaction • 2: 3D transaction that is sent to bank • 3: 3D transaction that is returned from bank
	f_hsm_data	Card data save status: • 0: Not to be saved • 1: Needs to be saved • 2: Successfully saved • 3: Failed to save card • 4: Failed to save card • 5: Saved data will be used • 6: Needs to be saved on a merchant side • 7: Successfully saved on a merchant side • 8: Failed to save card on a merchant side • 9: Needs to be saved for MOTO • 10: Successfully saved for MOTO • 11: Saved data will be used for MOTO
	terminal_name	Terminal name
	sli	Security Level Indicator
	eci	Electronic Commerce Indicator
	issuer_country	Card issuer country Alpha-2 code by ISO 3166-1 ('XX' value will be returned, if country was not detected)
	xid	Transaction Identifier for 3D transactions
	routing_string	Routing string
relation_source		Information about transaction(s) related to requested transaction. It can be **NULL** if there is no such transactions.
	dt_created	Transaction creation timestamp
	amount	Transaction amount
	currency	Transaction currency (currency code by ISO 4217)
	merchant_transaction_id	Parameter merchant_transaction_id from request
	init_transaction_id	Transaction initialization ID, generated by Gateway
	processor_transaction_id	Processor's transaction ID
	card_bin	Parameter card_bin from request
	f_status_str	String representation for transaction status
	f_relation_type_str	A string representation of "f_relation_type"
	routing_string	Account routing string
relation_target	%same sub-tags as for "relation_source" tag%	Information about transactions to which requested transaction is related. It can be **NULL** if there is no such transactions.

Notice: not all fields can be outputted for each transaction (for example, issuer country can be outputted only if antifraud was started for requested transaction, approval code exists only for successful transactions, etc.).

Notice: XML structure is fixed (tag names, parents-child's, etc.), but new tags can be added without notice (for example, new sub-tags can be added into <**requested**> tag, or after <**frontoffice**> tag).

2.5.3 AUTOMATIC REPORT

When a transaction is finished, the gateway can send a transaction status to the callback URL, if this feature is enabled for your account, and if you need it (by default, it is disabled). This report is a POST request, containing the same fields as an outputted string when you are completing the transaction.

You can choose what set of data you want to receive in this request from report types, shown in table 16. When sending Callback URL to our Gateway administrators, please, indicate what type of report you want to receive.

Please note, that callback won't be sent if cardholder closes his browser, of for non-final statuses like **HoldOK**. For more information on completing a transaction see the Completing a Transaction section.

2.5.4 AUTOMATIC REPORT MESSAGE SECURING FOR DATA INTEGRITY

A message with transaction status, sent to the callback URL, can be signed with Message Authentication Code (MAC) to ensure Message Integrity.

Notice: this feature is not enabled by default. Please, contact your account manager for details.

MAC value is passed in the *Digest* parameter. MAC calculation algorithm:

1. Concatenate all request parameters, sorted alphabetically by a parameter name, except parameter *Digest*, with a secret key. You will obtain a secret key from TransactPro.
2. Calculate SHA-512 hash from that string

Notice: value of *Digest* is BASE-64 encoded.

Notice: do not compare your calculated MAC with MAC, received in a message, directly with string comparison. This will allow a time-based attack and will compromise a secret key. For example, compare hashes from both MACs.

2.6 REFUNDS

URL: ***https://www2.1stpayments.net/gwprocessor2.php?a=refund***

The following limitations are applied to the refunds:

- Only a successful transaction can be refunded.
- 7 Partial refunds are allowed for each transaction

The following table describes a refund request fields:

21. Table

Field	Format	Value
account_guid	ans(19)	Your merchant account GUID.
pwd	h(40)	SHA1 hash of your processing password.
init_transaction_id	h(40)	Transaction ID received from the gateway.
amount_to_refund	n	Amount you want to refund, in **MINOR** units.
merchant_transaction_id	ans(5..50)	Optional field. Can be used to create merchant's id for transaction. Must be

		unique for every transaction you submit to the gateway (from 5 to 50 characters).
details	a(4)	Optional field, pass “true” to get additional information about payment.

See Figure 7 for scenario description.

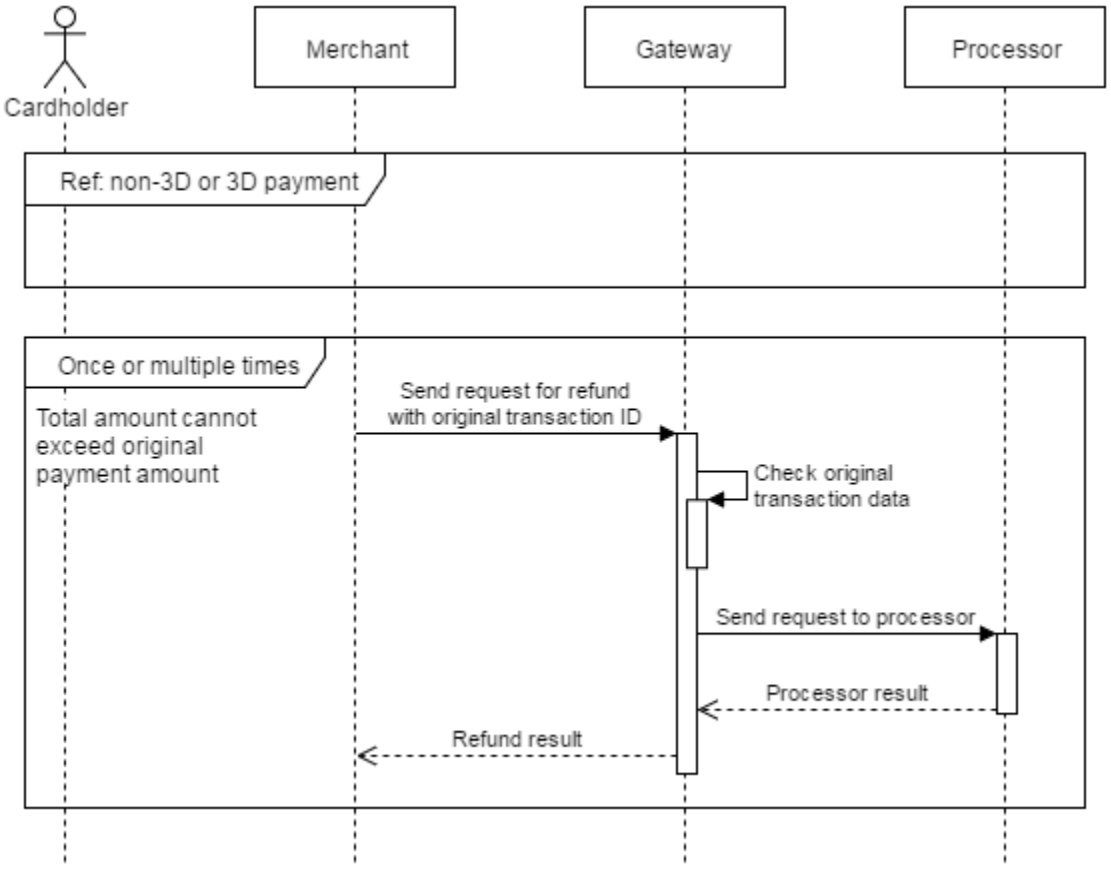


Figure 7 Refund

On a successful refund, you will get the following output:

```
Refund Success
```

If field "**details**" is passed and value is equal "**true**", response output has format:

```
Refund Success~internal_refund_id:c9e54697e4e3bfz2cbbbafccb4bz852dze8d0bc4~me
rchant_refund_id:cd2055c6bz4995dd938z350df86dcbcafd4za1fax22254113
```

If the gateway cannot refund this transaction, the gateway will output: Refund Failed

Notice: If the gateway cannot refund the transaction, please report the transaction ID to the gateway administrators. Do not perform multiple refund attempts. The refunds will be locked after 3 unsuccessful attempts to make a refund. The refunds for the transaction are locked for 10 minutes after each unsuccessful attempt.

Notice: To avoid double-refunds, gateway will block your attempts to send multiple refund requests for one transaction without delay. Now, delay is set to 3 minutes. Please also note, that you won't be able to refund DMS transaction right after the charge - delay is also 3 minutes.

3 Transaction Processing Customization

3.1 ENTERING A CARD INFORMATION ON THE GATEWAY SIDE

You can skip this section if you don't need this method.

Your account can be switched to the mode when a card information is not received from your server, but entered on the gateway side. In this case, on the step 1 you will get a transaction ID and the redirect link for a cardholder. When the cardholder submits his data on the card data page, he is redirected back to your side (i.e. in this case you should provide a default return URL link), then you should get a transaction status using a server to server request described in the Transaction Status Request section. Card collecting form can be customized, see manual chapter 9 for the details.

Notice: The cardholder should be forwarded to the redirect link only once, the redirect link expiration period is 120 seconds and transaction will be declined if the cardholder will click the Forward or Back button in his browser after submitting the form.

3.2 DUAL MESSAGE TRANSACTION

You can skip this section if you don't need this method.

A **Dual Message** (DMS) method is a method which allows you to divide a charge into the following steps:

- Transaction amount blocking on the cardholder's account.
- Charging the blocked amount (you can charge different amount, see notices for this chapter).

DMS transactions available for your account by default, as far as SMS transactions. (Some account types do not support DMS, contact your account manager for the details) To perform the DMS trans-action, submit the data, described in the Initializing a Transaction section, to the following URL:

For production ***https://www2.1stpayments.net/gwprocessor2.php?a=init_dms***

For test: ***https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=init_dms***

For the data described in the Completing a Transaction section (in case of card details collected at merchant side) change URL to the following URL:

For production: ***https://www2.1stpayments.net/gwprocessor2.php?a=make_hold***

For test: ***https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=make_hold***

In case of successful hold, the status message remains the following:

ID:GatewayTransactionID~Status:HoldOk~MerchantID:YourTransactionID~Terminal:YourTerminal~ResultCode:000

For the information on extended codes and system replies, see the Completing a Transaction section. If the returned status is **HoldOK**, the transaction amount is held on the cardholder's account. To finish the transaction and charge this amount, create the following server to server POST request:

22. Table

Field	Format	Description
init_transaction_id	h(40)	Transaction ID received on initialization step.
guid	ans(19)	Your merchant GUID.
pwd	h(40)	SHA1 hash of your processing password.

URL for production: ***https://www2.1stpayments.net/gw2test/gwprocessor2.php?a=charge_hold***

URL for test: ***https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=charge_hold***

Gateway output will be as described in the Completing a Transaction section (case for **'f_extended'**) flag not passed. Also, gateway will send a report to your callback URL, if this setting is enabled.

Notice: If the charging an amount failed, you can retry later, in some cases. If it is possible, the gateway will not set the **Failed** status to the transaction, but will leave the **HoldOk** status.

Gateway reply sample for this case:

ID:GatewayTransactionID~failed to charge hold. Please try again later.

The retry attempts for the failed transaction will be locked for 10 minutes. After 3 unsuccessful charge attempts, the transaction status will be set to Failed.

Notice: You can perform the charging of the held sum in 28 days. If you need to finish the DMS after 28 days, this can be done manually, but without charge back (CHB) warranty.

Notice: If the charging of amount failed, and **f_extended** flag was set to show details, you will see valid approval code and result code, as they were set to those values at **Hold** step. You need to check Status value to get operation status.

Notice: At this step, you can charge amount different from original one. (Lower up to 1 minor unit, or higher up to 10% from original). To do so, you need to pass additional field called "**charge_amount**". Value of this field should me amount to charge, in minor units. It's your responsibility if you're passing amount different from original: for example, if you'll pass "10" instead of "1000", transaction will be charged for 10 minor units and closed as successful, second charge is not possible. If you'll pass "**charge_amount**" field empty or with zero value (or will not pass this field at all), original transaction amount will be charged.

3.2.1 HOW TO CANCEL DMS HOLD WITHOUT REFUNDS

For some account types, you can cancel DMS transaction, in 24 hours after hold was done. To do it, send request to URL:

URL for production: ***https://www2.1stpayments.net/gwprocessor2.php?a=cancel_dms***

URL for test: ***https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=cancel_dms***

With same POST parameters as for usual refund. The only difference is that you can cancel entire amount only (no partial cancels).

3.3 DIRECT TERMINAL SELECTION MODE

You can skip this section if you don't need this method.

If your account receives card data from merchant side, and you have more than one terminal into account - you can use direct terminal selection mode. In this mode, you can force gateway ignore terminal balancing, and run transaction thru defined terminal. This is required, if you have a lot of terminals, and need to select pre-defined terminal for each transaction. To avoid creating multiple accounts with one terminal in each, you can (ask gateway support) to add all your terminals to one account, and use direct terminal selection mode.

To use this mode, pass additional field at the appropriate charge step together with card data (**"charge"** for SMS mode, **"charge_recurrent"** for recurrent mode etc.). Supported operations: SMS, DMS, CRD, P2P, recurrent payments. Field name is "**direct_terminal_selection**", field value is terminal external ID (request it from gateway support or your account manager).

In case transaction can be processed by passed terminal, gateway will just run transaction as usually. In case of error, another terminal won't be selected, and you will get declined transaction.

Notice: In direct terminal selection mode, you're responsible for correct terminal selection (terminal should belong to account you're pointing to with routing string, terminal should support transaction currency and card type, you should keep in mind monthly turnover on this terminal) – if you'll pass incorrect terminal, gateway will just reject this transaction with "no active terminals" error. (Extended error code visible into merchant area will be set to "Wrong currency or card type", as gateway interprets such situations as problems with transaction data, not wrong terminal).

3.4 CARD VERIFICATION

This function is needed for clients when is needed to accept only verified by merchant credit cards (see Figure 8).

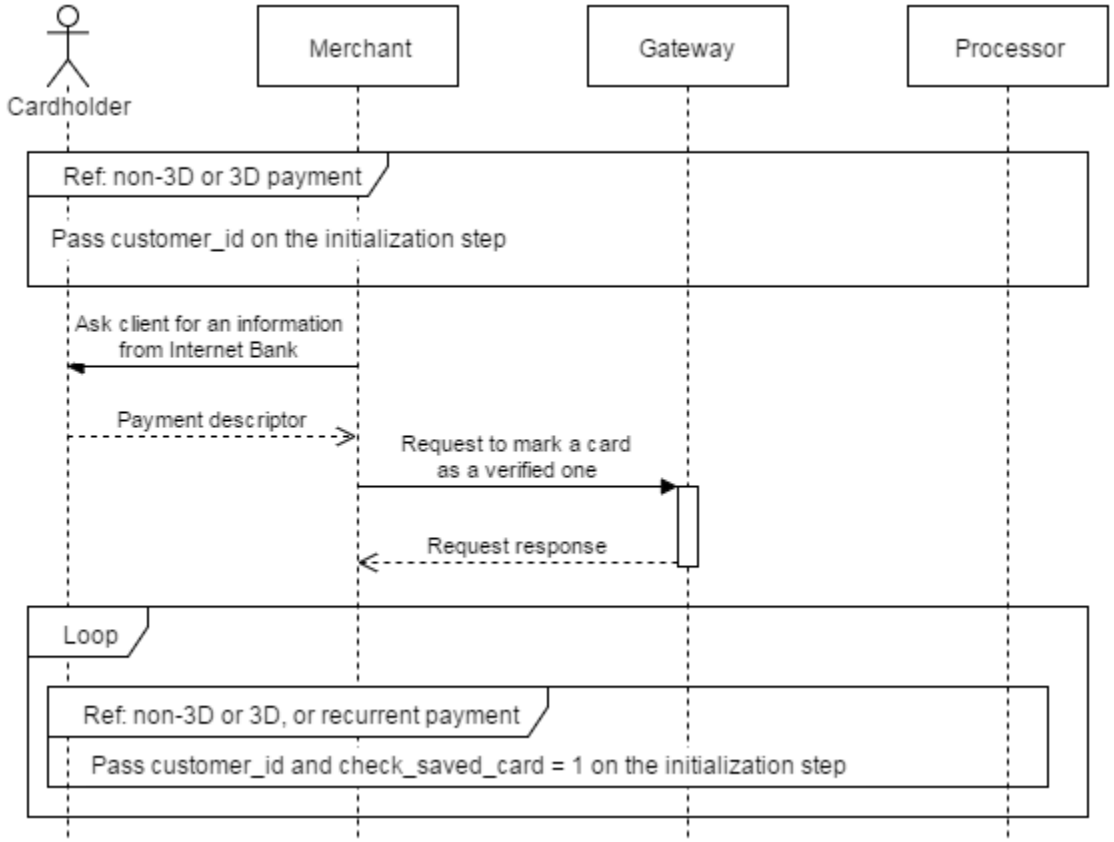


Figure 8 Card verification

1. Using any transaction type on initiation step *(init, init_dms, init_credit, init_p2p)* pass parameter "**customer_id**" for future card verification. After that create next correct request based on transaction type to pass card's data.

2. After getting positive answer from cardholder about transaction status, you must inform gateway about it, sending request to ***https://www2.1stpayments.net/gwprocessor2.php?a=verify_card*** (for production system) or ***https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=verify_card*** (for test system).

23. Table

Field	Format	Description
init_transaction_id	h(40)	Transaction ID received on initialization step
customer_id	ans(1..32)	Customer number

3. When verification process is completed, an initial DMS transaction can be cancelled or transaction of any other type can be refunded.
4. When verification process is completed merchant can create transactions passing **"check_saved_card"** equal 1 on initiation step and **"customer_id"** to use only verified cards for the customer.

Additional fields on any initiation step:

24. Table

Field	Format	Description
check_saved_card	n(1)	If value '1' is passed need to check card
customer_id	ans(1..32)	Customer number

You can use an *init_transaction_id* from the initial request for recurrent transactions if the parameter *save_card* was passed there and recurrent transactions are allowed for you.

You can perform recurrent transactions only of the same type the initial request was: recurrent SMS for initial SMS or DMS, recurrent P2P for initial P2P etc. Please, do not try to perform any other sequences (recurrent P2P after initial DMS etc.). Any unexpected transaction sequences can lead to financial loses and you will be fully responsible for it.

3.5 DYNAMIC DESCRIPTOR

Merchant can pass additional field **"merchant_referring_name"** on initiation request, **"merchant_referring_name"** must contain alphanumeric letters with symbols**. "Terminal name"** + **"merchant_referring_name"** together must not contain no more than 22 symbols. Terminal name is stored at our system.

4 Limits, Settings, and Frequent Problems

- Default transaction amount limit is 1000$ (or its equivalent in other currencies).
- To change the transaction amount limit, contact your account manager.
- By default, the anti-fraud module is enabled for your account, so it can reject some transactions with the high risk of fraudulence.
- We have the twenty-four-hour phone support number +372 67 222 555. The GMT zone for the phone is GMT+2.
- If the gateway rejects your transaction at the first step, please check the IP address the transaction is sent from, verify that the transaction ID is unique (if your previous transaction was declined, you cannot use the same ID to retry it), and that you're passing all mandatory fields.
- If you've tried to refund a transaction more than 3 times, other refund attempts for this transaction will be blocked.
- To solve this problem, contact the support department or your account manager.
- If you have some problems with a transaction processing, initial transaction ID's, send us your problem description and approximate problem time to help us solving your problem faster.
- Technical support email for the gateway is gwsupport@transactpro.lv . If you already have GUID for your account, you can insert it into email subject.

4.1 TERMINAL LIMITS

There are a possibility to get terminal limits for any of terminals, available for your merchant.

URL for production: ***https://www2.1stpayments.net/gwprocessor2.php?a=get_terminal_limits***

URL for test: ***https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=get_terminal_limits***

A Server to server POST request must contain values listed in the table below. All fields are mandatory, if other isn't stated.

25. Table

Field	Sub-field	Description
guid	ans(19)	Your merchant GUID.
pwd	h(40)	SHA1 hash of your processing password.
mid	n(1..21)	Terminal's MID.

A response will be in JSON format. Response for terminals that use common MC and VISA limit:

26. Table

<terminal currency>		Terminal's alpha-3 currency code (for example: EUR)
	month_current_amount	Used amount for current month
	month_stop_amount	Maximum amount allowed per month

	daily_current_amount	Used amount for current day (will not be present if daily limits are not allowed)
	daily_stop_amount	Maximum amount allowed per day (will not be present if daily limits are not allowed)

Response example for common MC and VISA limits

```
{
    "USD": {
        "month_current_amount": "0",
        "month_stop_amount": "35000000"
    }
}
```

Response for terminals that use separate MC and VISA limits will contain all allowed card types:

27. Table			
<card type>			Card type. Will be present only if appropriate card type is available for a terminal. Possible values: MC or VISA.
	<terminal currency>		Terminal's alpha-3 currency code (for example: EUR)
		month_current_amount	Used amount for current month
		month_stop_amount	Maximum amount allowed per month
		daily_current_amount	Used amount for current day (will not be present if daily limits are not allowed)
		daily_stop_amount	Maximum amount allowed per day (will not be present if daily limits are not allowed)

Response example for separated MC and VISA limits

```
{
    "MC": {
        "USD": {
            "month_current_amount": "0",
            "month_stop_amount": "35000000",
            "daily_current_amount": "0",
            "daily_stop_amount": "4000000"
        }
    },
    "VISA": {
        "USD": {
            "month_current_amount": "0",
            "month_stop_amount": "35000000",
            "daily_current_amount": "0",
            "daily_stop_amount": "4000000"
        }
    }
}
```

If any error occurred and it is impossible to load terminal limits, only field "ERROR" will be present in the response JSON.

Error response example

```
{
    "ERROR": "Error on loading data: Authentication failed"
}
```

5 Reconciled Transactions API

URL: ***https://www2.1stpayments.net/rt_api.php***

You can get a list of reconciled transactions using the gateway API. The date range is limited by 1 week (7 days), or 5000 transactions. You will get the corresponding error message if your selection contains more than 5000 records or your date range is wider than 7 days.

Notice: IP checking enabled for this operation, i.e. you should send this request from one of IP's allowed for transactions processing for any of your accounts linked to the passed GUID.

A request for receiving reconciled transactions should include the following POST fields:

28. Table

Field	Format	Description
guid	ans(19)	Your merchant GUID.
pwd	h(40)	SHA1 hash of your processing password.
dt_from	ns(10)	Reconcile date from, format YYYY_MM_DD.
dt_to	ns(10)	Reconcile date to, format YYYY_MM_DD.
f_output_type	n	Output format (optional). Default value: 1

If all parameters passed correctly, you will get the csv-output as return, with transactions reconciled into requested period.

Table below describes the different output formats:

29. Table

Value of *f_output_type*	Description
1	"%init_transaction_id%",%terminal_id%,"%settlement date%","%processed date%","%amount%","%card type%","%transaction type%","%arn%","%merchant_transaction_id%"
2	"%bank transaction ID%",%terminal_id%,"%settlement date%","%processed date%","%amount%","%card type%","%transaction type%"
3	"%init_transaction_id%",%terminal_id%,"%settlement date%","%processed date%","%amount%","%card type%","%transaction type%","%arn%"
4	"%init_transaction_id%",%terminal_id%,"%settlement date%","%processed date%","%amount%","%card type%","%transaction type%","%arn","%merchant_transaction_id%","%Masked PAN%","%E-mail%"

Example:
Settlement report for %Your GUID%
Report date range (SETTLEMENT date), from: %Date From – Date To%
Generation date: %Report generation date and time%

```
"%init_transaction_id%",%terminal_id%,"%settlement date%","%processed
date%","%amount%","%card type%","%transaction
type%","%arn%","%merchant_transaction_id%"
```

Line divider is “\n”.
If no transactions reconciled during the defined period – string “Report is empty” will be outputted.

The following table describes the output fields:

30. Table	
Filed	**Description and format**
Settlement date	Format: YYYY-MM-DD
Processed date	Format: YYYY-MM-DD HH:MM:SS
Amount	In **MINOR** units (i.e. 1000 for 10.00 dollars/euros/other currencies), negative for refunds
Card type	Type: Visa or MC
Transaction type	Type: Transaction or Refund
ARN	Acquirer reference number
merchant_transaction_id	Merchant transaction ID
Bank transaction ID	Transaction ID, received from a bank
Masked PAN	Masked PAN value. Format: NNNN********NNNN, where N is a digit
E-mail	An e-mail from a transaction

6 Recurrent transactions

Gateway allows you to process recurrent transactions – i.e., you can make regular charges from the card without saving card details on your side.

6.1 USING CARD DATA FROM "PLAIN" TRANSACTION FOR RECURRENT TRANSACTIONS

When you initialize transaction (*see section 2.2 of this manual*), pass additional field **'save_card'**, with value '1'. Gateway will save card data, when it will be entered by cardholder, and you will be able to process transactions using this card data for recurrent transactions. (For rare cases when card is saved at acquirer bank side, you have to pass '2' as **'save_card'** value).

6.2 USING NOT VERIFIED ON BANK'S SIDE CARD DATA FOR RECURRENT TRANSACTIONS

You must initialize store card operation for an appropriate type (SMS, CRD or P2P). The additional field **'save_card'** is not required for this operation. Gateway will check card number by internal algorithms and save card data without sending it to bank (see Figure 9).

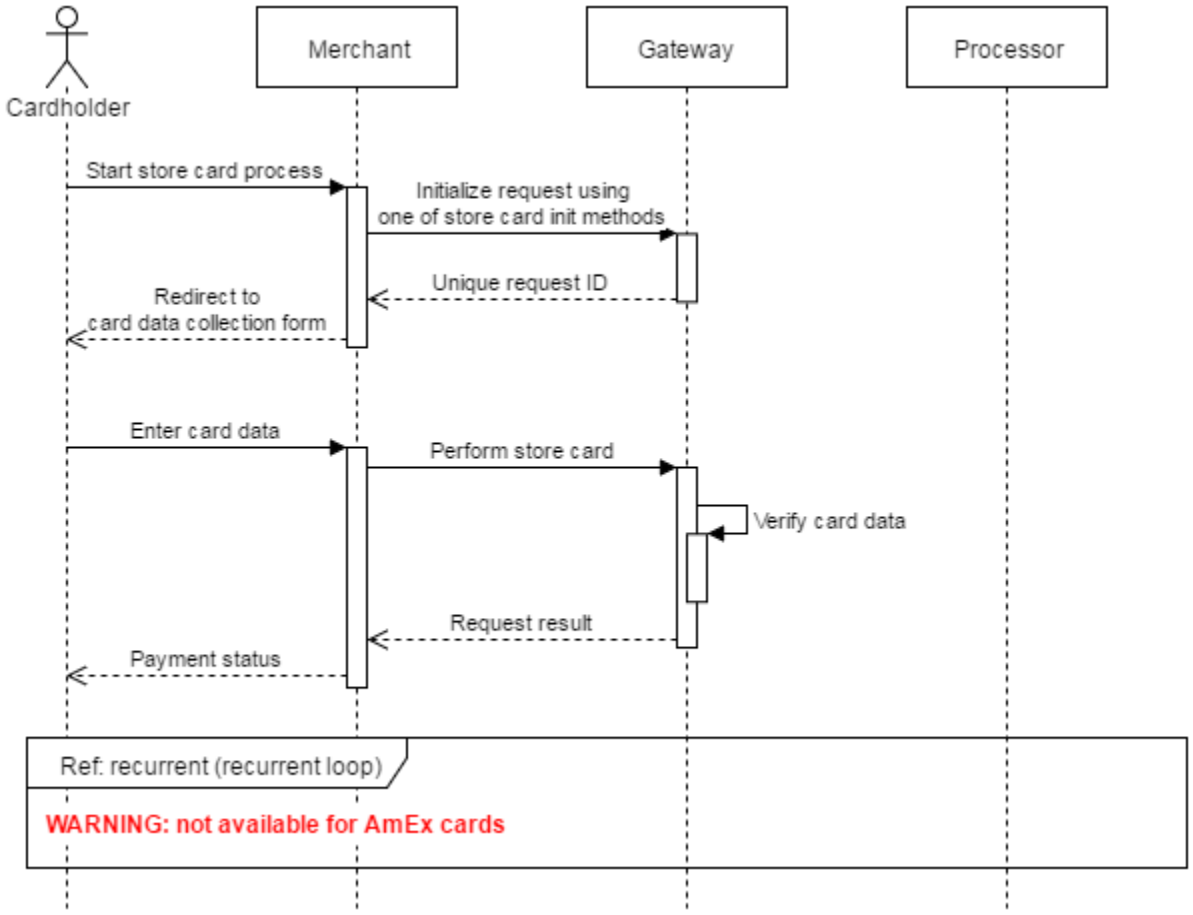


Figure 9 Store card scenario

On the first step, you must use a separate requests to initialize the store card operation for desired recurrent transactions type:

- To use a stored card for recurrent payments, send usual SMS initialization request (see section 2.2) to ***www2.1stpayments.net/gwprocessor2.php?a=init_store_card_sms*** (Test environment: ***https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=init_store_card_sms***)
- To use a stored card for credit operations, send usual CRD initialization request (see section 2.1.11) to ***www2.1stpayments.net/gwprocessor2.php?a=init_store_card_credit*** (Test environment: ***https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=init_store_card_credit***)
- To use a stored card for P2P operations, send usual P2P initialization request (see section 2.1.11) to ***www2.1stpayments.net/gwprocessor2.php?a=init_store_card_p2p*** (Test environment: ***https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=init_store_card_p2p***)

On second step send request to ***www2.1stpayments.net/gwprocessor2.php?a=store_card***

(Test environment***: https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=store_card***)

A server to server POST request must contain values listed in the table below. All fields are mandatory, if other isn't stated.

31. Table

Field	Format	Description
init_transaction_id	h(40)	Gateway transaction ID
cc	n(13..19)	Card number. Must be defined as a string without spaces or other dividers
expire	ns(5)	Card expiration date, in format MM/YY (optional for CRD and P2P)
f_extended	n	Return extended charge details (optional)
merchant_referring_url	ans(1..255)	Payment referring page URL, where cardholder enters PAN data (optional)

Notice: Gateway is saving card data during one year.

Notice: this feature is disabled by default. Before enabling it at production, gateway support should get approval from your account manager to enable it.

Notice: Not-3D terminal required for this feature. If you have 3D terminal and you want to do rebills, you need to order non-3D terminal (as far as get approval from your account managers for MOTO / Rebills).

6.3 SUBSEQUENT RECURRENT TRANSACTIONS

When first transaction finished, you can use **its init_transaction_id**, to make more transactions with the same card. For each transaction, you can change some fields (do not pass them if you want to keep them as into original transaction). Obligate fields must be passed each time you make recurrent transaction.

Notice: Recurrent transactions can be **SMS, CRD, P2P. (i.e. you cannot do recurrent DMS).**

Notice: In most cases, you will have one account for your plain transactions with cardholder presence, and second, for recurring transactions. Please note, that you're NOT allowed to do plain transactions using recurring transactions account. This will be judged as agreement violation.

Obligate fields for each recurrent transaction:

32. Table		
Field	**Format**	**Description**
guid	ans(19)	Your merchant GUID
pwd	h(40)	SHA1 hash of your processing password
rs	an(1..12)	Routing string
original_init_id	h(40)	init_transaction_id of your original transaction
merchant_transaction_id	ans(5..50)	Your transaction ID, **must be unique for every transaction** you submit to the gateway (from 5 to 50 characters)
amount	n	Transaction amount, in **MINOR** units (i.e. 2150 for $21.50 transaction)

Fields that can be changed:

33. Table		
Field	**Format**	**Description**
description	uns(5..255)	Order items description, from 5 to 255 characters (Example: SDHC Memory card x 2, AAA battery pack x 1)

Init recurrent SMS transaction
URL for production environment: ***www2.1stpayments.net/gwprocessor2.php?a=init_recurrent***
URL for test environment:
https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=init_recurrent

Init recurrent credit transaction
URL for production environment: ***www2.1stpayments.net/gwprocessor2.php?a=init_recurrent_credit***
URL for test environment:
https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=init_recurrent_credit

Init recurrent P2P transaction
URL for production environment: ***www2.1stpayments.net/gwprocessor2.php?a=init_recurrent_p2p***
URL for test environment:
https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=init_recurrent_p2p

You will get the same reply as **on 'init' step.**

To complete recurrent transaction, server to server POST requests should contain the following values:

34. Table		
Field	**Format**	**Description**
init_transaction_id	h(40)	init_transaction_id received for this recurrent transaction
f_extended	n	Return extended charge details (optional)

Complete recurrent SMS transaction
URL for production environment: ***www2.1stpayments.net/gwprocessor2.php?a=charge_recurrent***
URL for test environment:
https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=charge_recurrent

Complete recurrent credit transaction
URL for production environment: ***www2.1stpayments.net/gwprocessor2.php?a=do_recurrent_credit***
URL for test environment:
https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=do_recurrent_credit

Complete recurrent P2P transaction
URL for production environment: ***www2.1stpayments.net/gwprocessor2.php?a=do_recurrent_p2p***
URL for test environment:
https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=do_recurrent_p2p

You will get the same reply as on 'Complete the transaction' *(See section 2.4 of this manual)* step.

7 Using saved card for MOTO transactions

Gateway allows you to process MOTO transactions using saved cards.

7.1 SAVING CARD

First of all you must save card for SMS transaction.

When you initialize SMS transaction (see section 2.2 of this manual), pass additional **"save_card"** field with value **"4"** and after that make charge (see section 2.4). Gateway will save card data for future MOTO operations.

7.2 USING SAVED CARDS FOR MOTO TRANSACTIONS

7.2.1 INIT REQUEST

URL for production: ***https://www2.1stpayments.net/gwprocessor2.php?a=init***

URL for test: ***https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=init***

Make POST request operation for *init* SMS to use saved card from original transaction sending fields, listened bellow:

35. Table		
Field	**Format**	**Description**
guid	ans(19)	Your merchant GUID.
pwd	h(40)	SHA1 hash of your processing password.
rs	an(1..12)	Your routing string.
original_init_id	h(40)	init_transaction_id of your original transaction
merchant_transaction_id	ans(5..50)	Your transaction ID, must be unique for every transaction you submit to the gateway. The transaction ID must be from 5 to 50 characters.
description	uns(5..255)	Order items description, from 5 to 255 characters (Example: SDHC Memory card x 2, AAA battery pack x 1).
use_saved_card	n	Pass '1' for using saved card

7.2.2 CHARGE REQUEST

URL for production: ***https://www2.1stpayments.net/gwprocessor2.php?a=charge***

URL for test: ***https://gw2sandbox.tpro.lv:8443/gw2test/gwprocessor2.php?a=charge***

A server to server POST request must contain values listed in the table below. Make POST request operation for charge SMS transaction:

36. Table		
Field	**Format**	**Description**
f_extended	n	Return extended charge details. This is an optional field.
init_transaction_id	h(40)	Gateway transaction ID

8 Test environment

Transact Pro provides test environment not connected to real Visa / MasterCard networks, so you can begin integration before paperwork with the bank will be finished. Test environment details described into corresponding manual chapter.

If you want to use test account - we need to send you test card. We can do it via sms to cell phone. Test currency is USD (even if you will have another currency on production). Minimal transaction for 3D - USD 1.00

To access to test server, please use:

https://gw2sandbox.tpro.lv:8443/gw2test/* (ending depends on request type).

Instead of ***https://www2.1stpayments.net/**** from manual. By default, account will be set to collect card details on gateway side (Card form can be customized).

Login to merchant area on a test server:

https://gw2sandbox.tpro.lv:8443/gw2test/merchantarea.php (after first working week of year 2015)

As test server closed by firewall, please send all IP's involved in tests (including IP's you will use to access merchant area and use to fill card details at our side / redirect for 3D).

9 Card details form customization

9.1 CUSTOM DESIGN RESTRICTIONS

You can customize default card details form used by gateway. You can customize forms for input card data for SMS/DMS and credit transactions. So if you use both methods (Plain and credit), you should create two forms. Form for input card data for credit transaction must not have inputs for expire and CVV. To create new forms, you need to send to support your html to replace default one for your ac-count. (i.e. if you have more than one account – you need to indicate which one you're asking to modify).

General rules for custom design:

- No pop-ups, banners, links to external sites, etc.
- All images / scripts should be loaded from local directory, not from remote sites.
- You need to show transaction amount, currency and description on your form.
- Input field names (“<input type='text' name='field_name'”) should be same as on original form.
- You can edit or rewrite JavaScript to check input fields, but such script should present at your form.
- If you don't rewrite JavaScript to check input fields, elements with following IDs must exist in your form to show information: “lbl_name_on_cc“, “bl_cc_expire“, “lbl_cc“, “lbl_cvv“.
- You should test your design for cross-browser compatibility.
- You should name file as index.tpl.

9.2 CUSTOM FORM TAGS

{start_form} : <form onSubmit="return checkCCData();" method="POST" action="....">;
{submit_form} : <input type="submit" class="button" value="Process">;
{end_form} : </form>;
{default_js} : insert default JavaScript to check data entered
{description}: outputs “description” field passed at init step
{input_name}: outputs <input type="text" id="name_on_cc" name="name_on_cc" value="{name_on_card}" class="name_input_style" AUTOCOMPLETE="off">;
{amount} : outputs transaction amount (formatted)
{currency} : outputs transaction currency
{input_cc} : credit card number, output is <input type="text" id="cc" name="cc" value="" class="cc_input_style" AUTOCOMPLETE="off" maxlength="16">
{CC_EXPIRE}: expiration date, generates date range for card expire
{input_cvv}: CVV, output is <input type="text" id="cvv" name="cvv" value="" class="cvv_input_style" AUTOCOMPLETE="off" maxlength="3">

Path to your template location is set by tag {path_to_img}, i.e. this tag should be included in your links to images, e.t.c.

For example:
Path to your scripts:
<script type="text/javascript" src="{path_to_img}javacript_directory/script.js"></script>;

Path to your images:
 .

9.3 CUSTOM FORM – SAMPLE

```
<style type="text/css" media="all">
    body {
        background: #F5F5F5;
        font-family: Verdana, Geneva, sans-serif;
    }
    .tmain {
        border: 1px solid #CCCCCC;
        border-collapse: collapse;
    }
    .tmain tr td {
        font-size: 10pt;
        padding: 5px 10px 5px 10px;
        color: #696969;
    }
    .footer p {
        font-size: 7pt;
        font-family: Geneva, Arial, Helvetica, sans-serif;
        color: #888888;
        padding: 5px 5px 5px 5px;
    }
    .font input, .font select {
        font-size: 10pt;
        color: #00008B;
    }
```

```
    .button {
        color: #333;
        background: #A9A9A9;
        border: medium none;
        font-family: Verdana, Geneva, sans-serif;
        font-weight: bold;
    }
    .cc_input_style { width: 180; }
    .name_input_style { width: 180; }
    .cvv_input_style { width: 60; }
</style>
<html>
    <body>
    {start_form}
    <table width="100%" height="100%" border="0" cellpadding="0" cellspacing="0">
    <tr><td align="center">
    <table class="tmain" align="center" border=1 bgcolor="#ECECEC" width="600">
        <tr bgcolor="#ffffff">
            <td colspan=2 align="center"><img src="{path_to_img}form/cards.jpg"
vspace=20></td>
        </tr>
        <tr bgcolor="#A9A9A9">
            <td colspan=2><font color=#333> <b>Card details</b></font></td>
        </tr>
        <tr>
            <td bgcolor="#DCDCDC">Amount:</td>
            <td><font color="#B22222"><b>{amount} {currency}</b></font></td>
        </tr>
        <tr>
            <td bgcolor="#DCDCDC">Description:</td>
            <td>{description}</td>
        </tr>
        <tr>
            <td bgcolor="#DCDCDC" id="lbl_name_on_cc">Name on card:</td>
            <td class="font">{input_name}</td>
        </tr>
        <tr>
            <td bgcolor="#DCDCDC" id="lbl_cc">CC Num:</td>
            <td class="font">{input_cc}</td>
        </tr>
        <tr>
            <td bgcolor="#DCDCDC" id="lbl_cc_expire">CC Expire:</td>
            <td class="font">{CC_EXPIRE}</td>
        </tr>
        <tr>
            <td bgcolor="#DCDCDC" id="lbl_cvv">CVV: </td>
            <td class="font">{input_cvv}</td>
        </tr>
        <tr align="center">
            <td colspan="2" bgcolor="#DCDCDC">{submit_form}</td>
        </tr>
    </table>
    </td></tr>
    </table>
    {end_form}
    </body>
</html>
{default_js}
```

10 Hosted fields

10.1 BRIEF DESCRIPTION

Hosted Fields is a hassle-free solution to merchants that combine integration flexibility and compliance with data security requirements. Hosted Fields ensure PCI Council SAQ A eligibility. SAQ A is the most straightforward form of PCI, requiring no scans or auditors. Simply fill out a form, and send it to acquirer.

This functionality allows you to use your own payment form and keep data input only on Gateway side in the same time. So, you can customize your payment form as you wish without sending any templates to Transact Pro. However, you will not have any access to card data, because technically it is collected on the Gateway side.

Notice: this functionality is not available by default. Please, contact your account manager for details.

Supported operations:

1. SMS transaction
2. DMS transaction
3. CRD transaction
4. P2P transaction
5. Store card

10.1.1 WORKING SCENARIOS

The following steps can be performed for non-3D transaction:

1. Initialize transaction as usual. A field “TID” will be available in the response. This value must be used for Hosted Fields authorization.
2. Redirect cardholder to a payment page with Hosted Fields. The page must contain:
    a. A *form* element
    b. Containers for required fields (inside those container Gateway’s input fields will be loaded)
3. After cardholder enters card data, submit the payment form and an AJAX request will be performed to the Gateway.
4. The payment page will receive a response from the Gateway: payment is success or failed. The response will contain the same data as if the request would be sent to the Gateway API.

The following steps can be performed for 3D transactions:

1. Initialize transaction as usual. A field “TID” will be available in the response. This value must be used for Hosted Fields authorization.
5. Redirect cardholder to a payment page with Hosted Fields. The page must contain:
    a. A *form* element

b. Containers for required fields (inside those container Gateway's input fields will be loaded)

2. After cardholder enters card data, submit the payment form and an AJAX request will be performed to the Gateway.
3. The payment page will receive a response from the Gateway with link, where a cardholder must be redirected to complete a 3D authentication.
4. After cardholder completes the 3D authentication, he (or she) will be redirected to the page, configured for the account.

10.2 REQUIREMENTS

Code dependencies: jQuery 2.0.3 or higher

Browser support: before launching an application in a production, please, test a solution in all bowsers, which you plan to support.

Notice: old Internet Explorer versions are not supported.

10.3 CONFIGURATION

To use Hosted Fields functionality, you must plug in our JavaScript library:

```
<script src="https://www2.1stpayments.net/hostedfields/hosted-fields-1.0.5.js">
</script>
```

Or use *https://gw2sandbox.tpro.lv:8443/gw2test/hostedfields/hosted-fields-1.0.5.js* for test.

Supported fields for a payment form are listed in the table below:

37. Table

Field name	Description	SMS	DMS	CRD	P2P	Store card
name_on_card	Name on card for SMS, DMS and CRD transaction. Recipient name for P2P transaction.	M	M	M	M	M
pan	Card number	M	M	M	-	M
pan2	Card number for P2P transactions	-	-	-	M	-
cvv	Card Verification Value	M	M	-	-	-
expiry	Card expiration date in format "MM/YY"	M	M	O	O	M*

M – mandatory field
O – optional field
"-" – not used
* – mandatory for store card for SMS recurrent transactions and optional for store card for CRD and P2P transactions

Following events are available after payment form is initialized and any JavaScript function can be used there:

38. Table

Event name	Description
On load	Is triggered after all payment form's inputs are loaded. There are no input parameters for this callback.
On focus	Is triggered after input receives focus. Callback parameters: • *element:* jQuery element

On field validation	Is triggered after field is changed. Callback parameters: • *element:* jQuery element • *errorNo:* error number • *errorStr:* error string representation
Before submit	Is triggered when form is submitted, but before entered data is sent to the Gateway. Does not affect on submission process. There are no input parameters for this callback.
On submit	Is triggered after a response is received from the Gateway. Callback parameters: • *data:* JavaScript object that contains the Gateway response
On error	Is triggered when any unexpected error occurs. For example, when a form cannot be loaded or data submit failed. Callback parameters: • *error:* error description

Possible field validation error codes:

39. Table

Error number	Description
0	Success
1	Invalid PAN value
2	Invalid name on card
3	Invalid CVV
4	Invalid expiry

The following CSS classes are available for input's containers and will be set automatically on inputs change or form submission:

40. Table

Name	Description
tpro-focused	Input has received focus
tpro-valid	Input contains valid data
tpro-error	Input contains invalid data

10.4 INITIALIZATION WITH JAVASCRIPT

To load a payment form's fields, the *form* element must be present on the page. This element must have *https://www2.1stpayments.net* as *action* attribute's value (or use another link for test server: *https://gw2sandbox.tpro.lv:8443/gw2test*).

After page is fully loaded, call function *loadPaymentForm* for your form element. Function receives a JavaScript object with the following parameters:

41. Table

Name	Description	Options
tid	TID value, received on the initialization step	M
extended	The same value as *f_extended* for transactions. Determines a set of data that will be returned in the response. Default value: 0.	O
name_on_card	Configuration for field *name_on_card*.	O
pan	Configuration for field *pan*.	O
pan2	Configuration for field *pan2*.	O
cvv	Configuration for field *cvv*.	O
expiry	Configuration for field *expiry*.	O
onLoad	A callback for "On load" event.	O
onFocus	A callback for "On focus" event.	O

onFieldValidation	A callback for “On field validation” event.	O
beforeSubmit	A callback for “Before submit” event.	O
onSubmit	A callback for “On submit” event.	O
onError	A callback for “On error” event.	O

M – mandatory
O – optional

Two approaches can be used for a field configuration:

42. Table		
Minimal configuration		
Value	**Description**	
%CSS selector%	A CSS selector, which identifies a container for an appropriate input	
Extended configuration		
Option name	**Description**	**Options**
selector	A CSS selector, which identifies a container for an appropriate input	M
placeholder	Value that should be used as a placeholder for an input. Default values: • For field *name_on_card*: Name on card • For fields *pan* and *pan2*: 1111 1111 1111 1111 • For field *cvv*: 111 • For field *expiry*: MM/YY	O
default	A default value of an input.	O
optional	Can be *true* or *false* and indicates, is a field optional or not. Default value: *false*, the field is mandatory.	O

M – mandatory
O – optional

Code example:

```
<style>
    .tpro-focused {
        border-color: #00b3ee;
        box-shadow: 0 0 2px #00b3ee;
    }

    .tpro-error {
        border-color: #cf1e09;
    }

    .tpro-error.tpro-focused {
        border-color: #cf1e09;
        box-shadow: 0 0 2px #cf1e09;
    }

    .tpro-valid {
        border-color: #099f4e;
    }

    .tpro-valid.tpro-focused {
        border-color: #099f4e;
        box-shadow: 0 0 2px #099f4e;
    }
</style>

<form action="www2.1stpayments.net">
    Name on card: <div id="name_on_cc"></div>
```

```
    Card number: <div id="cc"></div>
    CVV: <div id="cvv"></div>
    Expiry: <div id="expiry"></div>
    <button>Confirm</button>
</form>

<script>
    var form = $('form').loadPaymentForm({
        tid: '%TID from initialization response%',
        extended: 8,
        name_on_card: {
            'selector': '#name_on_cc',
            'default': 'Name Surname'
        },
        pan: {
            selector: '#cc',
            placeholder: '1111 2222 3333 4444'
        },
        cvv: '#cvv',
        expiry: '#expiry',
        onLoad: function() {
            // implement your logic here
        },
        onFocus: function(element) {
            // implement your logic here
        },
        onFieldValidation: function(element, errorNo, errorStr) {
            // implement your logic here
        },
        beforeSubmit: function() {
            // implement your logic here
        },
        onSubmit: function(data) {
            // implement your logic here
        },
        onError: function(errors) {
            // implement your logic here
        }
    });
</script>
```

Notice: if you are using any JavaScript templating framework (AngularJS, mustache.js etc.), initialize payment form only after it will be added to the DOM. For instance, use custom directives for AngularJS, call the payment form initialization after template is rendered and added to its container with backbone.js etc.

10.5 DECLARATIVE INITIALIZATION

To load a payment form's fields, the *tpro-form* element must be present on the page. This element must have *https://www2.1stpayments.net* as *action* attribute's value (or use another link for test server: *https://gw2sandbox.tpro.lv:8443/gw2test*) and *tid* attribute with value, received on the initialization phase.

There are separate custom tags that is used as containers for each particular field. Field configuration is performed using tags attributes. A default value for an input is a tag's inner HTML.

Notice: this initialization way is not fully valid from strict HTML5 validation point of view, yet.

The following attributes are available for *tpro-form* element:

43. Table		
Name	**Description**	**Options**
action	https://www2.1stpayments.net for production or https://gw2sandbox.tpro.lv:8443/gw2test for test	M
tid	TID value, received on the initialization step	M
extended	The same value as *f_extended* for transactions. Determines a set of data that will be returned in the response. Default value: 0.	O
load	A function name for “On load” event’s callback.	O
focus	A function name for “On focus” event’s callback.	O
field-validation	A function name for “On field validation” event’s callback.	O
before-submit	A function name for “Before submit” event’s callback.	O
submit	A function name for “On submit” event’s callback.	O
error	A function name for “On error” event’s callback.	O

M – mandatory
O – optional

The following tags are available for appropriate fields:

44. Table	
Tag name	**Target field**
tpro-name-on-card	name_on_card
tpro-pan	pan
tpro-pan2	pan2
tpro-cvv	cvv
tpro-expiry	expiry

The following optional attributes are available for field configuration:

45. Table	
Name	**Description**
placeholder	Value that should be used as a placeholder for an input. Default values are the same as for the JavaScript initialization way.
optional	If present, the field will be optional. All fields by default are mandatory.

Code example:

```
<style>
    .tpro-focused {
        border-color: #00b3ee;
        box-shadow: 0 0 2px #00b3ee;
    }

    .tpro-error {
        border-color: #cf1e09;
    }

    .tpro-error.tpro-focused {
        border-color: #cf1e09;
        box-shadow: 0 0 2px #cf1e09;
    }

    .tpro-valid {
        border-color: #099f4e;
```

```
    }

    .tpro-valid.tpro-focused {
        border-color: #099f4e;
        box-shadow: 0 0 2px #099f4e;
    }
</style>

<tpro-form
    action="http://formtest.dev" tid="%TID from initialization response%"
    extended="8" load="onLoad" focus="onFocus" field-validation="onFieldValidation"
    before-submit="beforeSubmit" submit="onSubmit" error="onError"
>
    Name on card: <tpro-name-on-card>Name Surname</tpro-name-on-card>
    Card number: <tpro-pan placeholder="1111 2222 3333 4444"></tpro-pan>
    CVV: <tpro-cvv></tpro-cvv>
    Expiry: <tpro-expiry optional></tpro-expiry>
    <button>Confirm</button>
</tpro-form>

<script>
    function onLoad() {
        // implement your logic here
    }

    function onFocus(element) {
        // implement your logic here
    }

    function onFieldValidation(element, errorNo, errorStr) {
        // implement your logic here
    }

    function beforeSubmit() {
        // implement your logic here
    }

    function onSubmit(data) {
        // implement your logic here
    }

    function onError(error) {
        // implement your logic here
    }
</script>
```

Notice: if you add HTML code with JavaScript, you should initialize the payment form:

```
var form = $('tpro-form').loadPaymentForm();
```

Notice: if you are using any JavaScript templating framework (AngularJS, mustache.js etc.), initialize payment form only after it will be added to the DOM. For instance, use custom directives for AngularJS, call the payment form initialization after template is rendered and added to its container with backbone.js etc.

10.6 METHODS

To fully destroy an initialized form, use method *destroy* of the object, returned with *loadPaymentForm* function. This is mandatory when a payment form is destroyed by JavaScript without page reloading.

```
form.destroy();
```

11 Integration checklist

11.1 BEFORE YOU START

Here is a short checklist of facts and features you shall know and have implemented in your system:

1. Test account. There is no need to wait for your production (live) terminals or signed agreement to get test account. Simply request us to create test account as soon as you have received this manual**. If you skip this step you will have less time to complete your integration (or you will launch production later than you could).**
2. Account setup. By default, your production account is set to get card details at gateway side, and use 3D. On test server, after request from your side, we can switch test account to another modes (non-3D, get card from your side, add recurring billing or MOTO, set some required fields as non-mandatory etc.). But, for production, we can do it only upon receipt of an acceptance from your account manager. So, even if your test account was switched to some custom mode – your production will not be switched to that mode until this has been be approved. As this approval takes time – you shall get it prior switching to the live mode, not at the moment of the launch. **If this is not done – your launch to production can be postponed to some days, before you'll get approval (Or, if you don't – you'll need to implement scheme with 3D / card details at gateway side).**
3. Card details form customization. If your account is set to collect card details at the gateway side, respective gateway page can (and shall) be customized, so that its artwork would match one of your website**. If this is not done, some cardholders may simply close their browsers, after viewing payment page that looks not like your site.**
4. Card data verification. On our default form (which shall be customized using artwork of your website!) we have a script to verify card details and prevent submitting of the form if the card details are incorrect. If you are customizing this form – you have to use this script (or write your own). Same with card details sent from your side – you have to check them before you send them, as gateway will reject transaction with incorrect card details. **If this is not done, you will have increased level of rejected/ declined transactions.**
5. Customer data verification. As mentioned in point 2 above, by default, you have to pass all the required data according to the manual. You have to check the data on your side before sending to the gateway, as gateway will reject transaction with wrong or missed transaction details. **If this is not done, you will have increased level of rejected / declined transactions.**
6. Production account testing. It's highly recommended to test your production account before going live. For sure, our support department is doing all the tests on our side before sending you production details, but, testing from the merchant side will let you test full cycle – from creating order and sending us initial data, to return to your site and getting transaction details. **If this is not be done, you may experience problems with transactions flow failed / rejected when going live!**
7. Front office data verification. Gateway has an API to report status for a requested transaction. You need to use this feature to get status for each questionable transaction – for example, when cardholder didn't return to your website, etc**. If this is not done, time gap between problem origin and problem fix will be much longer!**
8. Back office data verification. Gateway can export its back office (successful transactions + refunds), the description of this feature may be found in chapter 0. It is strongly recommended to get reconciled transaction list once per business day (for the previous day), and compare with successful transactions and refunds lists that you have in your database. This is much easier to trace any divergence on the early stage. **If this is not done, it will take much longer to solve any problem with payouts and accounting. Also, this may cause problems listed in point 7.**