MEDORO Electronic Payment Gateway

Documentation for Merchants

v1.6.2

1. Table of contents

2. Preface

This document describes integration process and usage options for ***Medoro***, the electronic payment gateway, provided by AS LPB Bank (LPB). ***Medoro*** allows merchants to collect Visa, MasterCard or Maestro credit card payments online. It ensures secure and reliable card data entry on merchants’ websites for cardholders.

3. Confidentiality

This document, its appendices and attachments are intellectual property of ***Medoro*** and are confidential.

4. Document history

Table 1. Document history.

Date	Version	Description of changes	Author of changes
04.09.2013.	v1.0.0	First version of the document.	Anita Murzina
18.09.2013.	v1.1.0	Updates and extensions throughout the whole document. Request data on API via SOAP interface added (see section 8.4.1).	Anita Murzina
19.09.2013.	v1.1.1	Payment processing state 5 added, state 10 removed (see section 6.6).	Anita Murzina
26.09.2013.	v1.2.1	Section on terms (see section 5) and section on merchant's MPI (see section 6.5) added; sequence of sections changed; minor changes included.	Anita Murzina
28.10.2013.	v1.2.2	Payment response data and Token Registration response data edited (see sections 8.4.2.1 and 8.4.2.2).	Anita Murzina
29.10.2013.	v1.2.4	Inaccurate XSD schemas corrected.	Anita Murzina
01.11.2013.	v1.2.5	Specification of elements "Expiry" (see sections 8.4.1.3, 8.4.1.8), "StartDate" and "LastDate" (see section 8.4.2.1) updated.	Anita Murzina
02.12.2013.	v1.2.6	Possible length for name on credit card and for name of token increased.	Anita Murzina
02.01.2014.	v1.3.0	Information about Original credit (see sections 5, 6.5) and Server callbacks (see sections 5, 7, 8.1) included. Specification on CSC updated (see section 8.4.1.3), terms "Additional payment", "Token", "Recurring payment" (see sections 5, 6.6.1) added, appendices removed, minor changes included.	Anita Murzina
10.01.2014.	v1.3.1	Description of Error codes (see sections 8.3.3, Appendix 1: Error codes) and Action codes (see sections 8.4.2.1, Appendix 2: Action codes) added.	Anita Murzina

14.01.2014.	v1.3.2	Specifications for dynamic descriptor changed (see sections 8.4.1.1 and 8.4.1.3).	Anita Murzina
30.01.2014.	v1.3.3	Element <RemoteAddress> in Payment request data through SOAP request is now mandatory (see section 8.4.1.3). XML examples updated. List of error codes extended (see Appendix 1: Error codes).	Anita Murzina
28.02.2014.	v1.3.4	3D Secure Enrollment Statuses and 3D Secure Authentication Statuses updated (see Table 19 and Table 20).	Anita Murzina
07.03.2014.	v1.3.5	3D Secure Enrollment Statuses updated (see Table 19), private action codes edited (see Appendix 2: Action codes).	Anita Murzina
12.05.2014.	v1.3.6	Action code 180 changed to 181, new action code 180 added (see Appendix 2: Action codes).	Anita Murzina
01.07.2014.	v1.3.7	New element “MessageID” included in request and response data (see sections 8.4.1.3 and 8.4.2.1).	Anita Murzina
14.07.2014.	v1.4.0	Element “DeliveryAddress” removed, element “Country” under “BillingAddress” changed in request and response data (see sections 8.4.1.1, 8.4.1.3 and 8.4.2.1).	Anita Murzina
19.08.2014.	v1.4.1	Value “MD” in Table 5 is edited.	Anita Murzina
28.08.2014.	v1.4.2	New error code 305122 added (see Appendix 1: Error codes).	Anita Murzina
10.09.2014.	v1.5.0	Payment processing state “Closed” is removed (see sections 6.6, 8.2, 8.4.1). Time reference changed from GMT+2 to EET – Eastern European Time.	Anita Murzina
25.09.2014.	v1.5.1	Payment processing cycle updated – transition after timeout while being at the state “Approved” is no longer available (see section 6.6).	Anita Murzina

13.10.2014.	v1.6.0	Custom MPI feature removed.	Nikolajs Arhipovs
12.02.2015.	v1.6.1	Descriptor and CSC field lengths corrected (reduced to 20 and 3 accordingly). CSC requirement rules changed (for Original credits, payments with tokens and Maestro cards CSC field is optional). Merchant interface layer removed from system.	Nikolajs Arhipovs
10.11.2016.	v1.6.2	Server callbacks documentation corrected (see section 5).Original credit redesigned (see section 6.5), mode 8 added (see sections 6.4, 7.1.7) and payment request data adjusted (see sections 8.4.1.1, 8.4.1.3). In some cases time for cancelling authorisations reduced (see section 6.6.1).	Grigorijs Golubevs

5. Terms

3D Secure

XML-based protocol that adds additional security measure to online payments with credit cards. In addition to standard payment procedure, cardholder is also required to pass 3D Secure authentication. It ensures that payment is initiated by cardholder and therefore cannot be claimed to be fraudulent. 3D Secure was developed by Visa and is identified by name *Verified by Visa*. MasterCard has adopted it under name *MasterCard SecureCode*.

Abandoned payment

Payment which was initiated, but for which cardholder did not enter card data during predefined period of time and payment was declined by timeout. It will be deleted from system after 30 days, thus releasing order ID, and making payment unavailable both on web portal and by calling GetPayment through SOAP request.

ACS

Access Control Server (ACS) is a software provider that enables cardholder to undergo 3D Secure authentication in his card issuer bank (if his card is 3D Secure enrolled). ACS can be provided by card issuer bank, or outsourced.

Authorization

Process of putting hold on cardholder’s funds. Authorization is successful when cardholder has sufficient funds, rights, etc. for creating the payment.

Batch

In this document batch refers to a set of financial transactions and reversals done in one financial day. All of these transactions and reversals end up in one batch. Financial day (and thus batch) is closed by system at 00:00 EET (Eastern European Time) if it was not closed manually. After batch is closed, a fee of €0.50 is applied to all new reversals on transactions from this batch.

Example:
If batch was closed at 12:00 EET, then merchant would not be able to close the new batch until that day’s 00:00 EET (12 hours later). Also, if not closed manually, this new batch will close at 00:00 EET of the next day (36 hours after previous batch was closed).

Deposit

Process of writing off cardholder’s funds that were put on hold.

Hold

Cardholder’s funds that are frozen on his account. Hold can be either cancelled (funds will be unfrozen) or deposited (see term *deposit*).

MO/TO

Mail order / telephone order (MO/TO) means that merchant receives cardholder’s order and card data through mail, e-mail or telephone.

Original credit

Type of payment where cardholder receives funds from merchant.

PCI DSS

Payment Card Industry (PCI) Data Security Standard (DSS) ensures security of cardholders’ data. It provides a baseline of technical and operational requirements used for protecting cardholders’ data. PCI DSS compliant merchants are allowed to collect, store, process and transmit cardholders’ credit card data, and therefore can create payments or tokens also through SOAP protocol.

Recurring payment

Type of payment where cardholder allows merchant to make regular charges from his credit card where only first charge is initiated by cardholder. Card data is not saved into merchant’s database.

Reversal

Process of returning cardholder’s funds back to his account.

- **Online reversal:** hold on funds is cancelled or funds are reversed on the same financial day as they were authorized or deposited.
- **Offline reversal:** funds are reversed after the day they were deposited.

Server callbacks

As there is no guarantee that cardholder will return to merchant’s website after conducting a payment, there is a possibility merchant will not receive status on payment through LPB’s response by HTTP POST request. It can be avoided by using Server callbacks. They work in the following way: when payment status changes (to any other payment processing state except for 0, 1 and 7), LPB automatically makes POST request with payment status to merchant’s System callback URL.

In order to activate Server callbacks service, send URL to your account manager.

Important:

- It can be used only in modes 3 and 5.
- It cannot be used for token registration.
- LPB has to register merchant's Server callback URL for merchant to use this service.
- LPB makes POST request during 2 minutes time after payment status changes.
- Server callback URL should respond to LPB's POST request with HTTP status **202 Accepted** (without content). It should be done during 2 seconds time. In case LPB does not receive exactly the specified response during this time, it is assumed that LPB could not deliver payment status to merchant and LPB attempts to send it again during the next 2 minutes time (with maximum number of attempts being 5).

SMS & DMS transactions

During successful payment creation, each payment has to be authorized and deposited.

- **Single Message transaction (SMS):** payment is authorized and deposited simultaneously.
- **Dual Message transaction (DMS):** at first payment undergoes authorization, and only then during predefined period of time merchant can accept the payment by depositing it.

Token

Merchant can allow cardholder to conduct payments without entering his credit card data, simultaneously not saving card data into merchant's database. It is done by registering credit card as a token.

Transaction descriptor

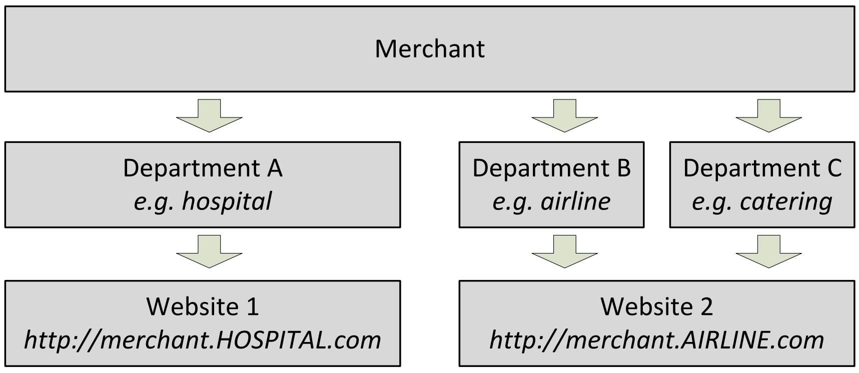
Value provided to card issuing bank during authorization. It consists of merchant's name, terminal's city and terminal's country, and should be shown in cardholder's bank statement.

6. Introduction

6.1. Merchant's account

When registering to *Medoro* payment gateway, **merchant** is given account on *Medoro* web portal. Account can be linked to one or several **departments** that differ by type of activity and are identified by MCC (merchant category code – number assigned by Visa or MasterCard to classify merchant's business by type of goods or services provided). Each department represents a particular **website**.

Merchant can have several websites. However, for each of them different department has to be registered. For each employee merchant can request an account linked with same departments. See Schema 1.



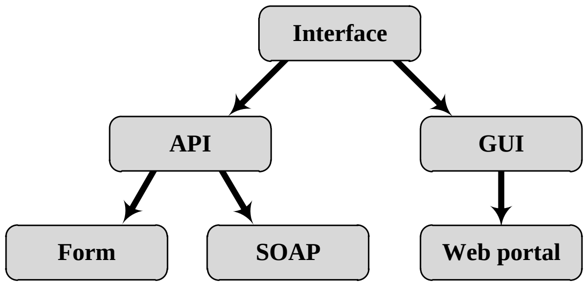
Schema 1. Departments.

6.2. Security levels

- **MO/TO:** payment processing without security measures (3D Secure check), where credit card data is passed from cardholder to merchant by mail, e-mail or telephone.
- **Non-3D Secure:** credit card payment is processed with CSC code check as the only security measure.
- **3D Secure:** in addition to CSC code check, if credit card is enrolled in 3D Secure, payment does not get processed until cardholder passes 3D Secure authentication on his card issuer bank's ACS page.
- **Cardholder Not Present:** payment processing without cardholder's initiation, for instance, in case of recurring payments. First payment is considered to be subscription and processed as a regular payment with corresponding security measures (*MO/TO* through *SOAP interface*, *Non-3D Secure*, or *3D Secure*), whereas subsequent payments occur without cardholder's initiation through *mode 9*.

6.3. Types of interface

In order to provide merchants with optimal solutions to their unique requirements for payment gateway, *Medoro* has adopted three types of interface: form, SOAP and web portal (see Schema 2). Each of them provides merchant with a different set of functions.



Schema 2. Types of interface.

- **API via forms:** from merchant's website cardholder is redirected to a form provided by *Medoro* payment gateway, where he can enter his credit card data. It is useful whenever it is necessary to collect card data from cardholder without merchant having access to it. When creating payments or tokens, forms are the only option for merchants who are not PCI DSS compliant and therefore are not allowed to collect and store card data on their servers.

  <u>This interface can be used to create new payments or tokens (see section 8).</u>

- **API via SOAP:** necessary information from cardholder is collected directly by merchant on his website and then provided to LPB system through SOAP interface. SOAP interface is exposed to all merchants; however, in case merchant is not PCI DSS compliant, it is restricted to operations that do not require card data (everything except payment and token creation).

  <u>This interface can be used to create new payments or tokens, to deposit or reverse payments, to get status of payments, etc. (see section 8).</u>

- **GUI via web portal:** each merchant has account on *Medoro* payment gateway's web portal, where he is given certain functionality.

  <u>This interface can be used to deposit or reverse payments, to fill in MO/TO form, to view information about payments, to get status of payments, to get statistics on financial days, to access merchant's profile, etc. (see section 9).</u>

6.4. Payment processing modes

New payment can be created using *web portal*, *form* or *SOAP* interface with one of the following security levels: *MO/TO*, *Non-3D Secure*, *3D Secure* or *Cardholder Not Present*. Merchant can be registered in system with one or several combination/s of interface type and security level (availability depends on merchant being PCI DSS compliant). The following table (see Table 2) displays possible combinations of interface types and security levels where each checkbox stands for a *payment processing mode* that can be chosen. However, each of the modes has to be registered separately. After registration, payments can be initiated using any of the registered modes.

Table 2. Payment processing modes.

Security / Interface	MO/TO	Non-3D Secure	3D Secure	Cardholder Not Present
Web portal	☑ *(mode 1)*	-	-	-
Form	-	☑ *(mode 3)*	☑ *(mode 5)*	-
SOAP	☑ *(mode 2)*	☑ *(mode 4)*	☑ *(mode 6)*	☑ *(mode 8,9)*

6.5. Additional features

Merchant can also choose additional features to be used in payments processing (see Table 3). First, each of the necessary features has to be enabled separately. Then, by indicating it during payment initiation or processing, each of them may be added to any of the payment processing modes if not stated otherwise.

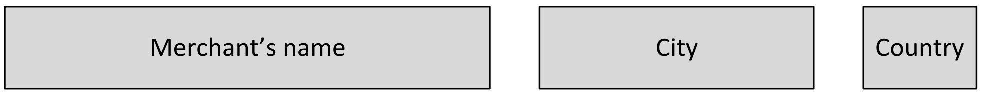
Table 3. Additional features.

Recurring payments	☑
Tokens	☑
Dynamic descriptor	☑
Original credit	☑

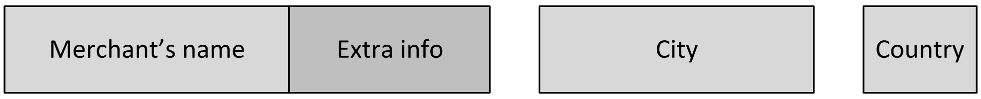
- **Recurring payments:** for cardholder to initiate first charge (called subscription), merchant has to create a new payment with one of the following modes: 2,3,4,5,6 (see section 8.4.1.1 or 8.4.1.3). Elements *Recurring→Frequency* and *Recurring→EndDate* must be included in the request. Merchant can initiate subsequent recurring payments by creating new payments with mode 9, using *Recurring→ID* from recurring payment subscription.

- **Tokens:** to use token, first, merchant has to register credit card as a token. Afterwards, merchant can make payments by choosing the registered token instead of requiring cardholder to enter his credit card data again. Tokens can be used with modes 2,3,4,5,6. When initiating payment with token through modes 2, 4, 6, CSC is optional.

- **Dynamic descriptor:** merchant can form transaction descriptor dynamically, defining content of cardholder's credit card statement for each transaction. It is accomplished by reducing possible length of merchant's name and using remaining space for extra information such as description of a product being purchased, identification of sub-merchant, identification of cardholder, etc. All symbols that are allowed to be used in dynamic descriptor, are described in XSD schemas provided with this documentation (see file *requests\form\InitiateRequest.xsd* or *requests\soap\PaymentRequest.xsd*). Dynamic descriptor can be used with modes 2,3,4,5,6.



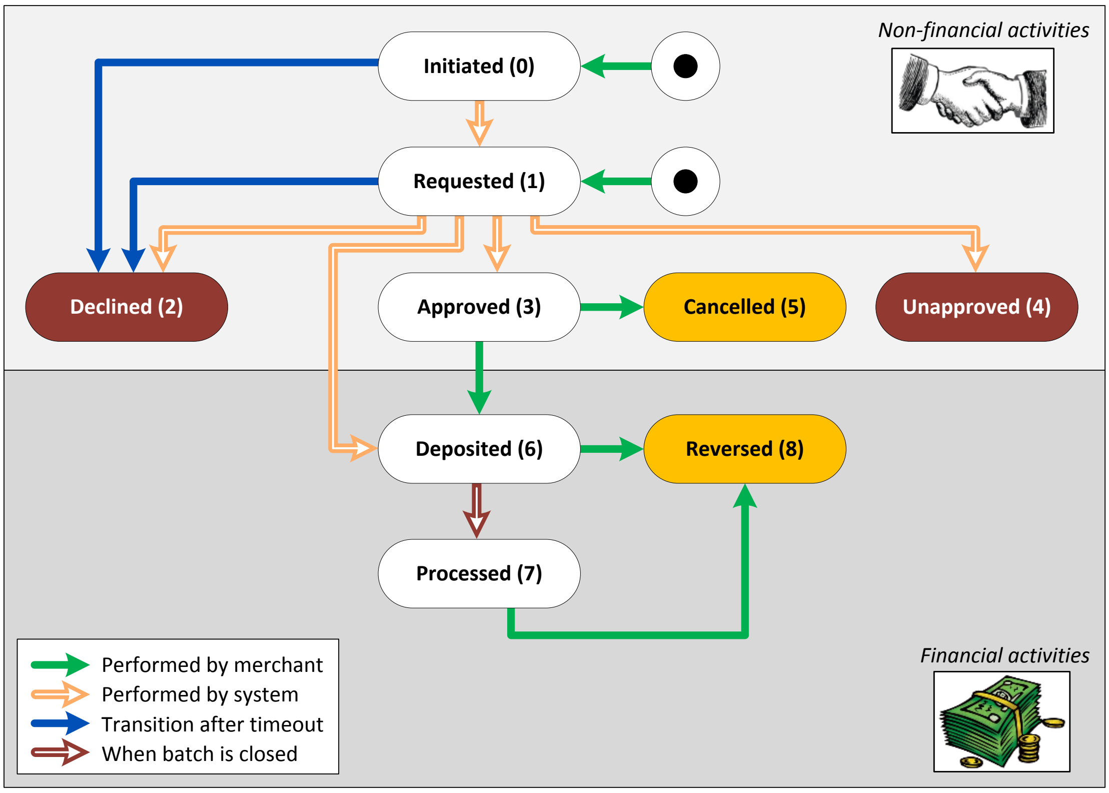
Picture 1. Standard descriptor.



Picture 2. Dynamic descriptor.

- **Original credit:** these payments are initiated the same way as usual payments, but with negative amount and they are always processed as SMS transactions. Transaction amount is limited to 50 000 dollars or its equivalent in used currency (calculated by LPB’s exchange rates). Daily, weekly and monthly limits are determined for each merchant individually and are specified in contract. These payments can be initiated only if at least one successful non-reversed payment on the same PAN exists. Original credit can be used with modes 1,2,3,4,5,6,8. For Original credit transactions CSC is optional. Reversal can be requested during 24 hours after payment initiation. *Medoro* has 3 ways to initiate an Original credit transaction. First method – you can initiate a basic transaction in mode 2,3,4,5,6 with negative payment amount. Second method – you can use a “Payout” button in ipsp.lv → Payments list. Third method – you can use a mode 8.

6.6. Payment processing cycle



Schema 3. Payment processing cycle.

Note: Non-financial activities refer to steps that do not directly involve transferring of funds. Whereas financial activities refer to steps where funds are transferred or prepared to be transferred to another account.

6.6.1. *Payment processing states*

Table 4. Payment processing states.

Payment Form

Start at *Initiated*

- **Performed by merchant:** on merchant's website (in case of *Payment Form* interface) cardholder selects an order. Cardholder is forwarded to LPB *Payment Form*.

Initiated (0)	Newly created payment. Card data is not yet provided.	Next state: ***Declined*** ***Requested***

Initiated* → *Declined

- **Transition after timeout:** if required credit card data was not provided during predefined period of time, payment is automatically declined and is regarded as **Abandoned payment** (see section 5).

Initiated* → *Requested

- **Performed by system:** after required credit card data is provided, system automatically switches payment state to *Requested*.

SOAP, MO/TO

Start at *Requested*

- **Performed by merchant:** on merchant's website (in case of *SOAP* interface) or through merchant (in case of *MO/TO*) cardholder selects an order and enters credit card data.

Requested (1)	Newly created payment. Credit card data is provided.	Next state: ***Declined*** ***Unapproved*** ***Approved*** ***Deposited***

Requested* → *Declined

- **Transition after timeout:** if 3D Secure authentication response was not provided during predefined period of time, payment is automatically declined.
- **Performed by system:** in case payment authentication failed.

	Requested* → *Unapproved • **Performed by system:** in case payment authorization was not successful.	
	Requested* → *Approved • **Performed by system:** in case payment authorization was successful, and it was DMS transaction.	
	Requested* → *Deposited • **Performed by system:** in case payment authorization was successful, and it was SMS transaction.	
Declined (2)	During predefined period of time not all required data is entered, or 3D Secure authentication is not successful. Therefore, payment is declined.	Next state: -
Unapproved (4)	Payment authorization is not successful. Therefore, payment cannot be processed further.	Next state: -
Approved (3)	Payment is authorized (funds are put on hold on cardholder's account).	Next state: ***Deposited*** ***Cancelled***
	Approved* → *Deposited Authorization may be captured only during 30 days after being made – afterwards it is automatically cancelled by the card issuing bank. • **Performed by merchant:** merchant can capture authorized payment.	
	Approved* → *Cancelled Authorization may be cancelled only during 48 hours (in some cases during 24 hours) after being made. • **Performed by merchant:** merchant can cancel authorization.	
Deposited (6)	Authorization is captured (funds are transferred from cardholder's account to merchant's account). **Note:** Authorization may also be captured partially. Remaining funds will be released.	Next state: ***Processed*** ***Reversed***
	Deposited* → *Processed • **When batch is closed:** when batch is closed, payment state automatically switches to *Processed*. Batch can be closed by system or manually by merchant.	

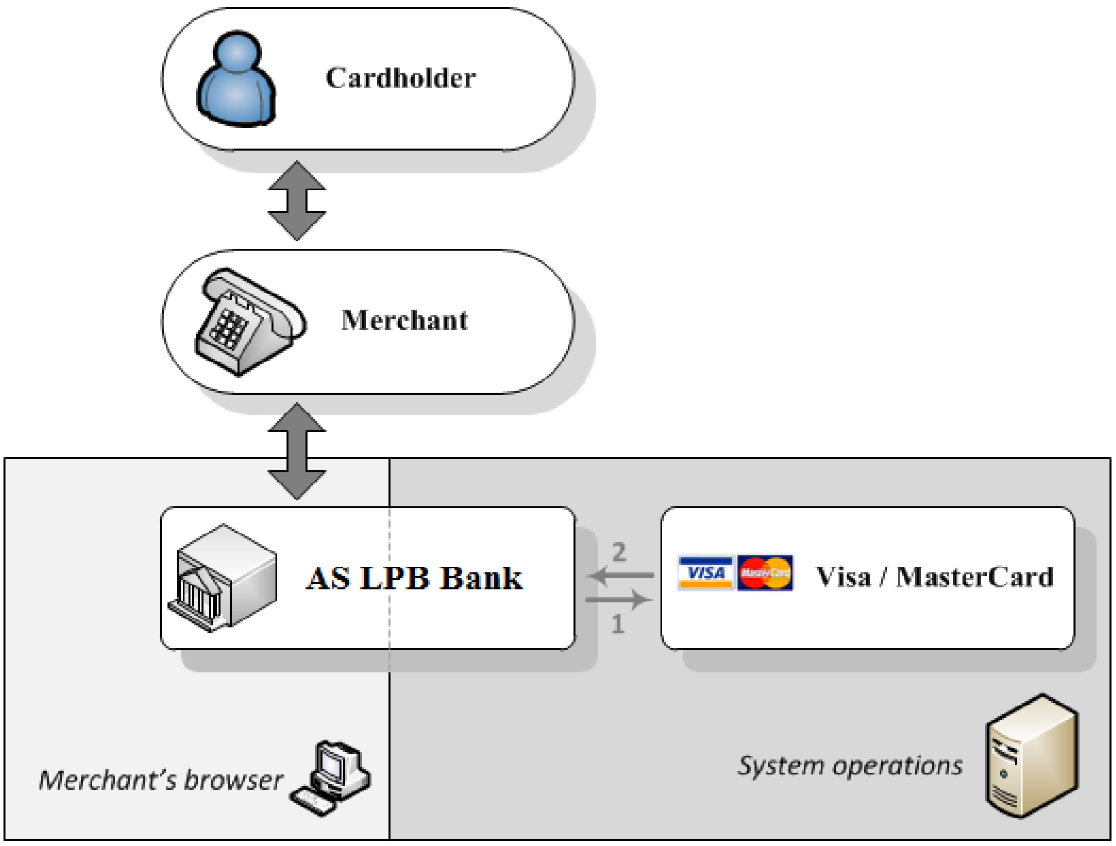
	Deposited → ***Reversed*** • **Performed by merchant:** merchant can reverse payment.	
Processed (7)	LPB’s batch is closed and funds are being transferred to merchant’s account.	Next state: ***Reversed***
	Processed → ***Reversed*** This operation costs €0.50 per transaction. • **Performed by merchant:** merchant can reverse payment.	
Cancelled (5)	After merchant’s request for payment cancellation, hold on cardholder’s funds is removed.	Next state: -
Reversed (8)	After merchant’s request for payment reversal, transferred funds are returned back to cardholder’s account. **Note:** Merchant can also request partial reversal by returning only part of deposited funds.	Next state: -

7. Payment creation process (in different modes)

Here are schemas describing sequence of each payment processing mode. Every step is briefly described below the schemas.

- Steps in "[]" refer to 3D Secure payment processing modes (5,6) and are skipped in case credit card is not enrolled in 3D Secure.
- Steps in grey refer to system operations that are not directly visible to cardholder or merchant.
- Steps in "{}" refer to Server callbacks (see section 5) and are skipped in case merchant's Server callback URL is not registered.

7.1.1. Mode 1 (MO/TO, form)

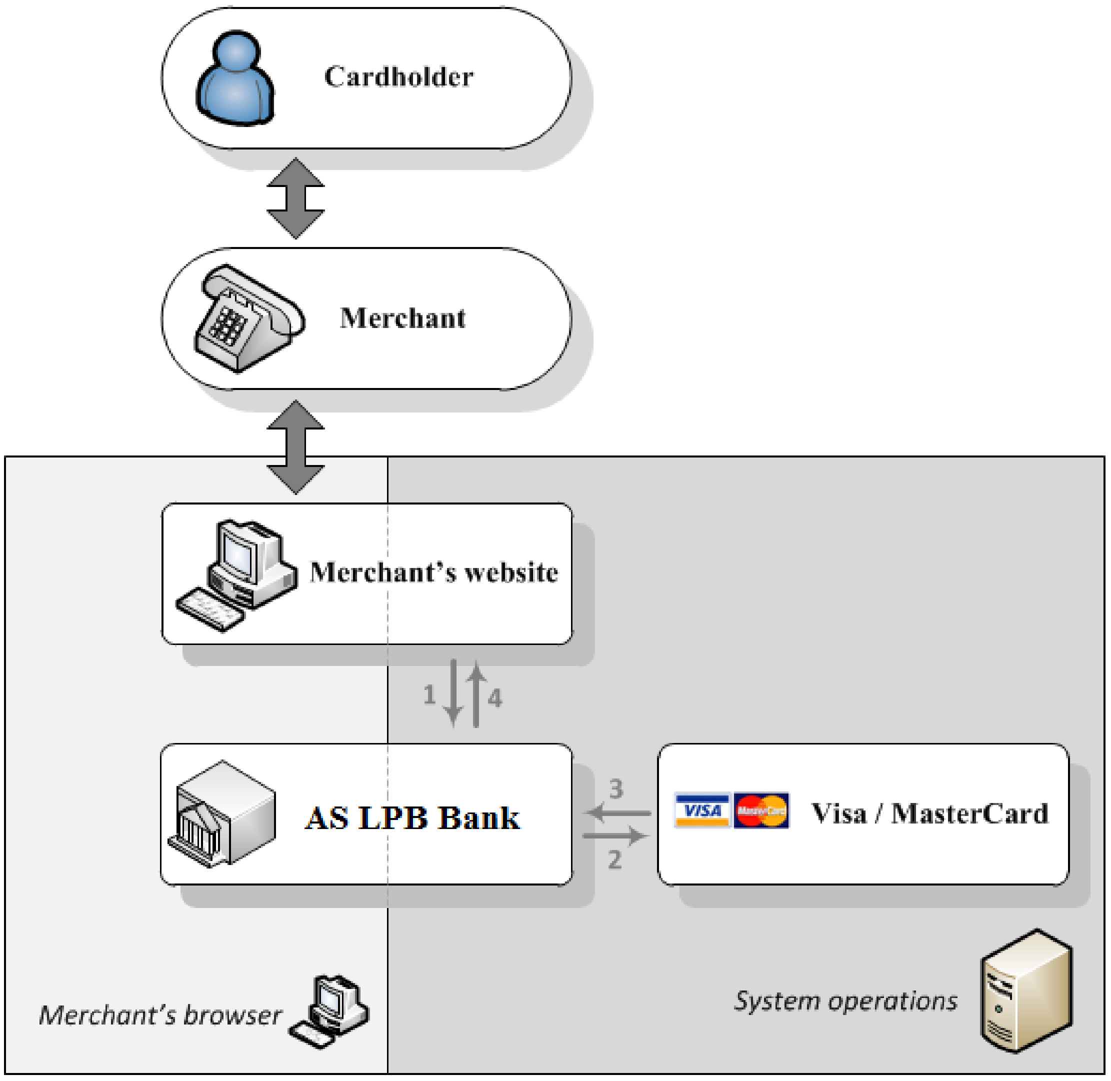


Schema 4. Mode 1.

1 After cardholder provides credit card data to merchant through phone or mail, merchant enters received data into LPB *Payment Form* on *Medoro* web portal. Then LPB contacts Visa / MasterCard for credit card verification and payment processing.

2 Visa / MasterCard returns response to LPB, which appears in merchant's account, thus enabling him to notify cardholder on payment processing status.

7.1.2. Mode 2 (MO/TO, SOAP)



Schema 5. Mode 2.

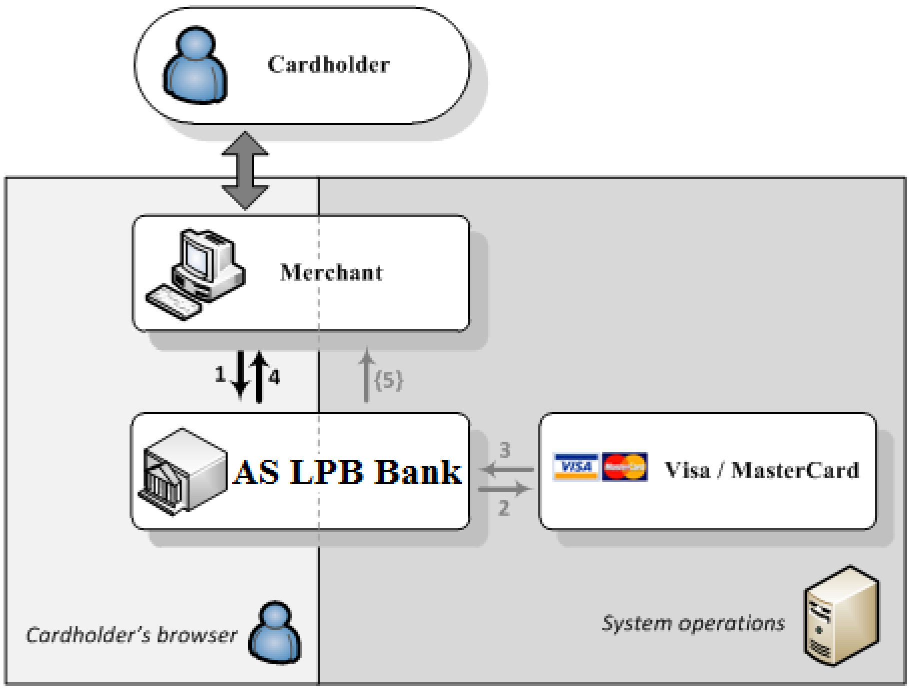
1 After cardholder provides credit card data to merchant through phone or mail, merchant enters received data into his website. Then merchant sends corresponding data to LPB (see section 8.4.1.3) by making SOAP request (see section 8.2).

2 LPB contacts Visa / MasterCard for credit card verification and payment processing.

3 Visa / MasterCard returns response to LPB.

4 LPB returns response to merchant (see section 8.4.2.1), thus enabling him to notify cardholder on payment processing status.

7.1.3. Mode 3 (Non-3D Secure, form)



Schema 6. Mode 3.

1 After cardholder enters merchant's website, chooses items he intends to purchase and decides to pay for them, merchant sends corresponding data to LPB (see section 8.4.1.1) by redirecting cardholder to LPB *Payment Form* (see section 8.1) for entering credit card data.

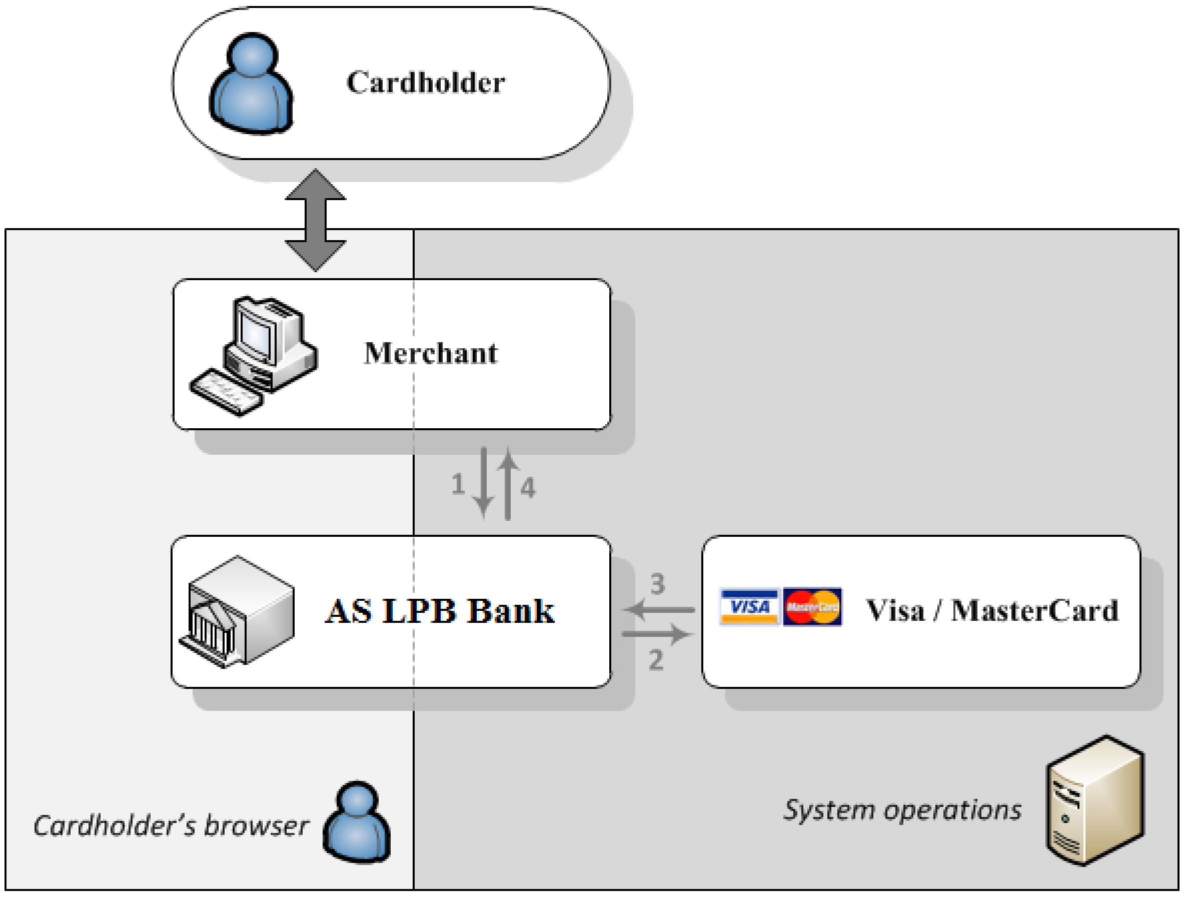
2 LPB contacts Visa / MasterCard for credit card verification and payment processing.

3 Visa / MasterCard returns response to LPB.

4 LPB returns response to merchant (see section 8.4.2.1) by redirecting cardholder back to merchant's website where he sees payment processing status.

{5} LPB sends update on payment status to merchant's System callback URL.

7.1.4. Mode 4 (Non-3D Secure, SOAP)



Schema 7. Mode 4.

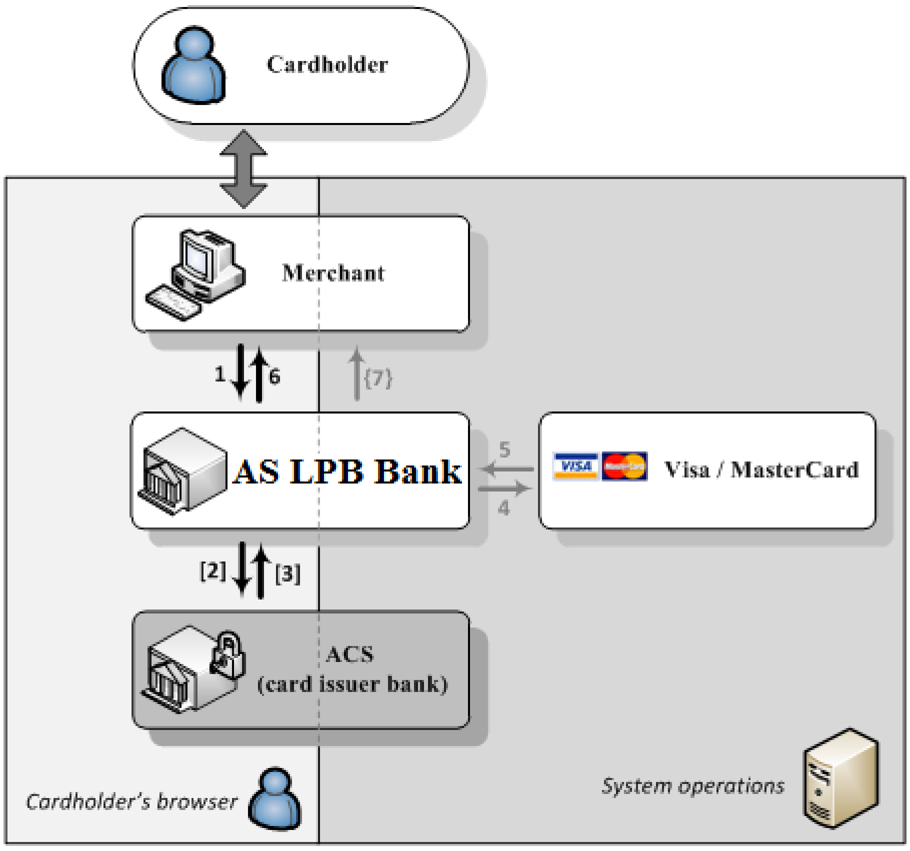
1 After cardholder enters merchant's website, chooses items he intends to purchase and decides to pay for them, he enters credit card data directly into merchant's website. Then merchant sends corresponding data to LPB (see section 8.4.1.3) by making SOAP request (see section 8.2).

2 LPB contacts Visa / MasterCard for credit card verification and payment processing.

3 Visa / MasterCard returns response to LPB.

4 LPB returns response to merchant (see section 8.4.2.1) and cardholder is shown payment processing status.

7.1.5. Mode 5 (3D Secure, form)



Schema 8. Mode 5.

1 After cardholder enters merchant's website, chooses items he intends to purchase and decides to pay for them, merchant sends corresponding data to LPB (see section 8.4.1.1) by redirecting cardholder to LPB *Payment Form* (see section 8.1) for entering credit card data.

[2] LPB checks whether credit card is enrolled in 3D Secure. If credit card is enrolled, cardholder is redirected to his card issuer bank's ACS page.

[3] After card authentication, ACS returns response to LPB and redirects cardholder back to LPB's website.

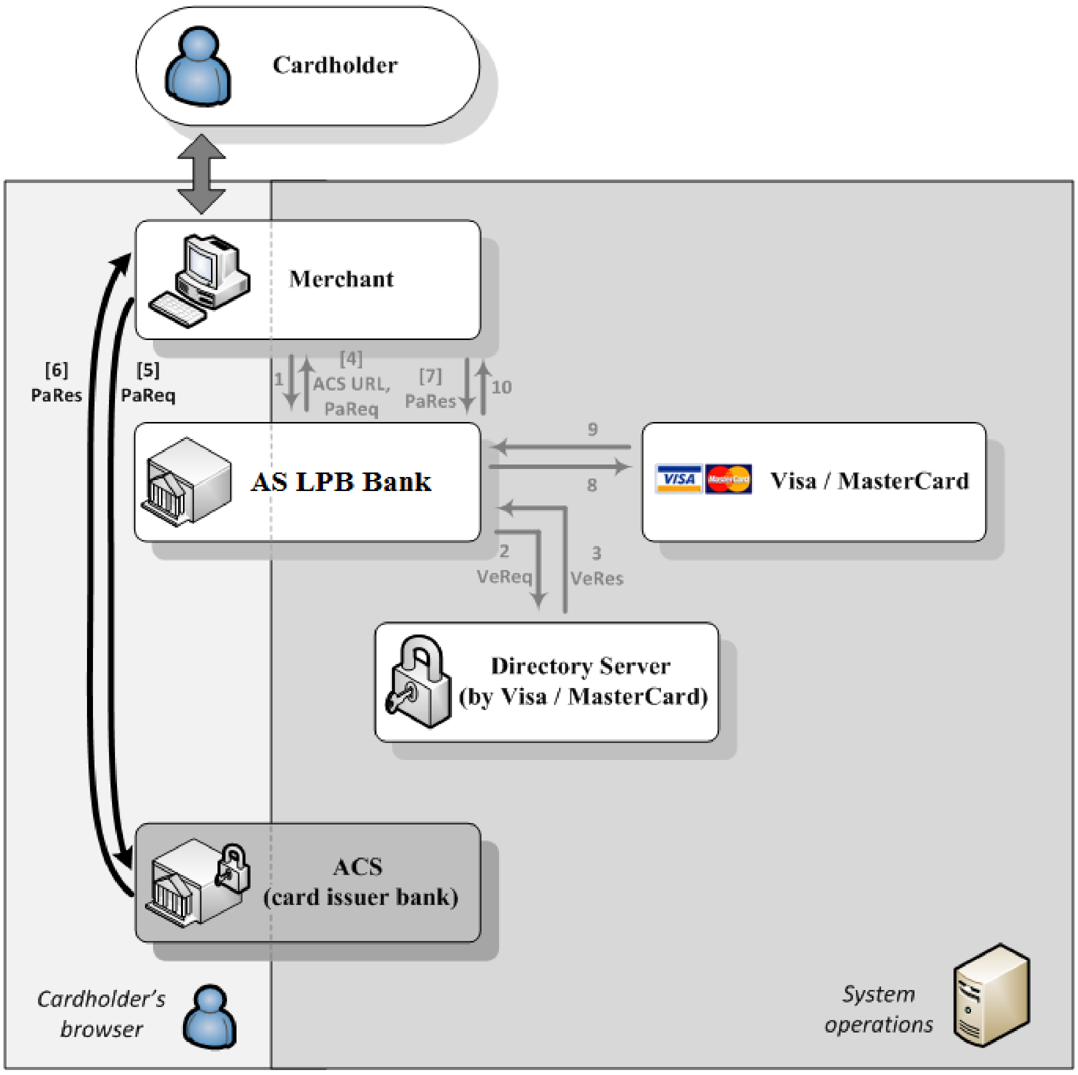
4 LPB contacts Visa / MasterCard for credit card verification and payment processing.

5 Visa / MasterCard returns response to LPB.

6 LPB returns response to merchant (see section 8.4.2.1) by redirecting cardholder back to merchant's website where he sees payment processing status.

{7} LPB sends update on payment status to merchant's System callback URL.

7.1.6. Mode 6 (3D Secure, SOAP)



Schema 9. Mode 6

1 After cardholder enters merchant's website, chooses items he intends to purchase and decides to pay for them, he enters credit card data directly into merchant's website. Then merchant sends corresponding data to LPB (see section 8.4.1.3) by making SOAP request (see section 8.2).

2 LPB sends VeReq (Verify Enrollment Request) to Directory Server in order to check whether credit card is enrolled in 3D Secure.

3 Directory Server returns VeRes to LPB.

[4] LPB returns response with ACS URL and PaReq to merchant (see section 8.4.2.1).

[5] Merchant redirects cardholder to his card issuer bank's ACS page. It is done by making POST request with the following values:

Table 5. Values for sending to card issuer bank's ACS page.

Value	Description
PaReq	Cardholder's authentication request.
TermUrl	URL to which cardholder will be redirected at the conclusion of 3D Secure authentication.
MD	Free text parameter that has to be supplied and will be echoed back when cardholder is redirected back to the TermUrl. **Some ACS may not work correctly if this value is not passed.**

[6] After card authentication, ACS redirects cardholder back to merchant's website (to TermUrl sent by merchant) by making POST request with the following values:

Table 6. Values received from card issuer bank's ACS page.

Value	Description
PaRes	Cardholder's authentication response. This value should be provided to LPB.
MD	The value supplied previously if included in the POST parameters in the request to card issuer bank's ACS page.

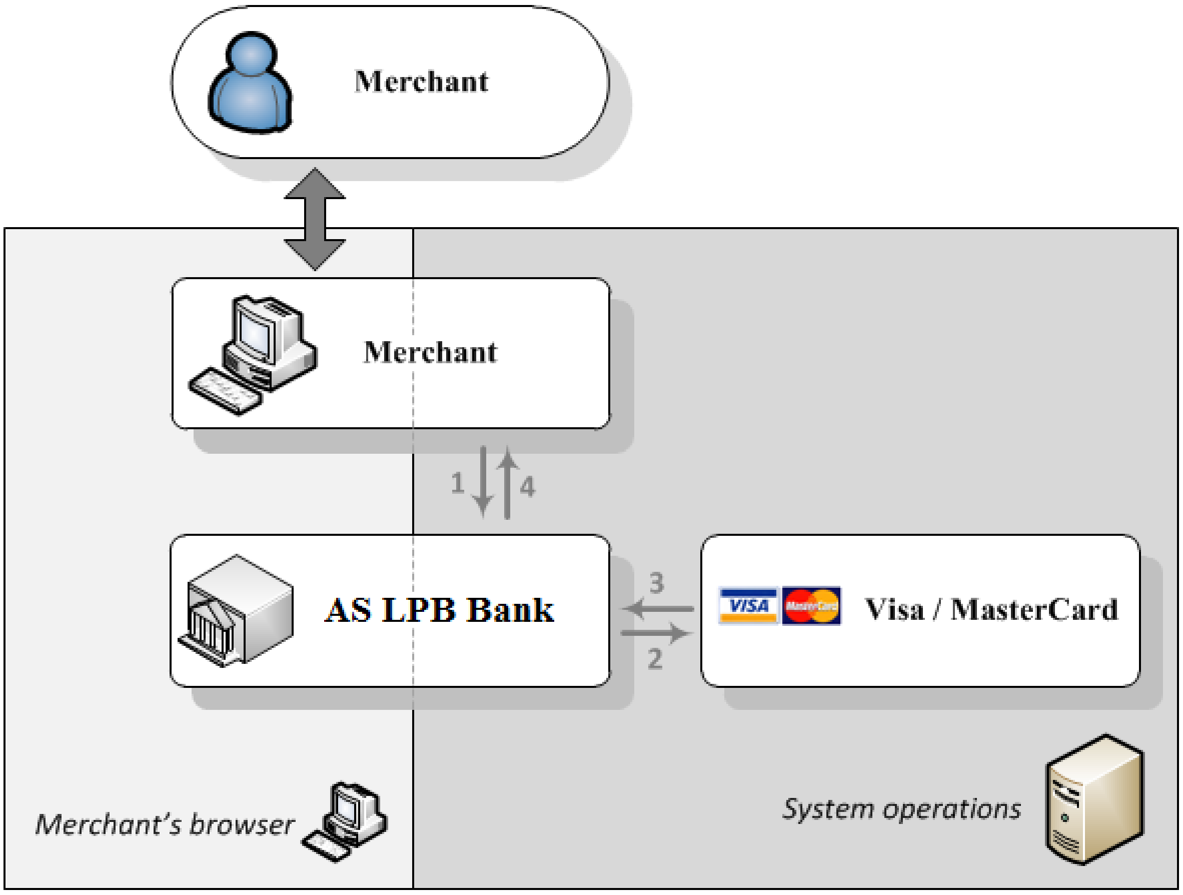
[7] Merchant returns 3D Secure authentication response to LPB (see section 8.4.1.4).

8 LPB contacts Visa / MasterCard for credit card verification and payment processing.

9 Visa / MasterCard returns response to LPB.

10 LPB returns response to merchant (see section 8.4.2.1) and cardholder is shown payment processing status.

7.1.7. Mode 8 (Cardholder Not Present, SOAP)



Schema 10. Mode 8.

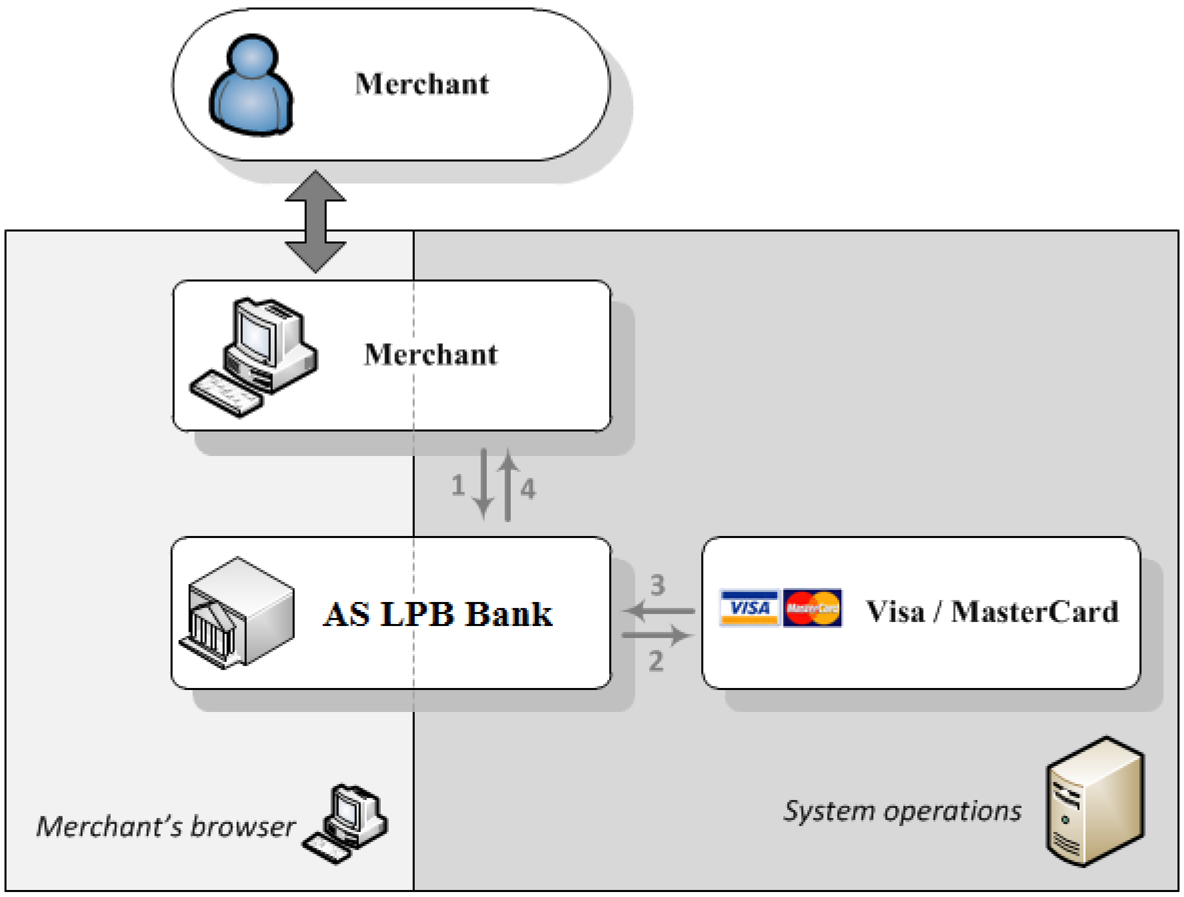
1 Merchant sends payment details to LPB (see section 8.4.1.3) by making SOAP request (see section 8.2). However, merchant can initiate Original credit payment to a card only in case he has at least one successful non-reversed payment (state 6,7) on that card. Therefore, when initiating Original credit payment, instead of sending credit card data, merchant makes reference to data entered during such previously conducted payment.

2 LPB contacts Visa / MasterCard for credit card verification and payment processing.

3 Visa / MasterCard returns response to LPB.

4 LPB returns response to merchant (see section 8.4.2.1) with payment processing status.

7.1.8. *Mode 9 (Cardholder Not Present, SOAP)*



Schema 11. Mode 9.

1 After cardholder has subscribed for recurring payments, merchant can initiate subsequent payments himself. Merchant sends payment details to LPB (see section 8.4.1.3) by making SOAP request (see section 8.2). However, instead of sending credit card data, he makes reference to data entered during subscription.

2 LPB contacts Visa / MasterCard for credit card verification and payment processing.

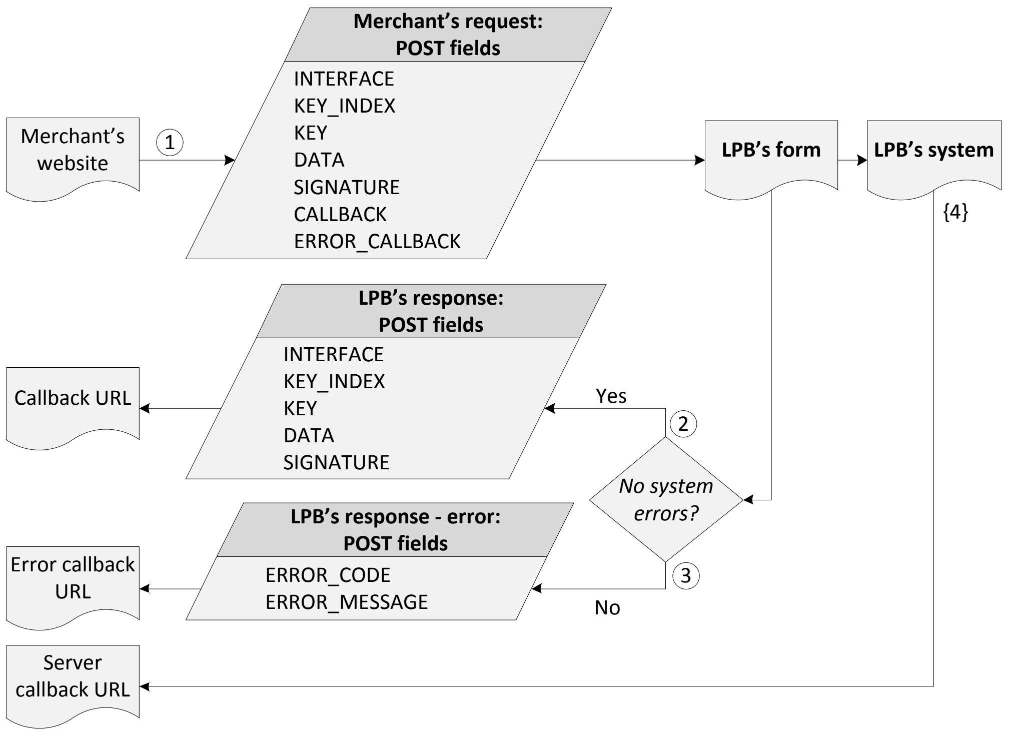
3 Visa / MasterCard returns response to LPB.

4 LPB returns response to merchant (see section 8.4.2.1) with payment processing status.

8. API

8.1. API via form

This is the basic solution for conducting payments in case merchant is not PCI DSS compliant. It enables him to create new payments or tokens through *Payment Form* or *Token Registration Form* respectively. Merchant's request and LPB's response occur in the following sequence:



Schema 12. Data flow diagram of request and response for API via form.

1 When cardholder chooses to pay for order or to register a new token, merchant makes HTTP POST request to *Payment Form* or to *Token Registration Form* by sending merchant's request with corresponding data to LPB (see section 8.3.1). Cardholder is redirected to a corresponding LPB's form.

2 In case request ends successfully, cardholder is redirected back to merchant's callback URL through HTTP POST request with LPB's response (see section 8.3.2).

3 If system error happens, payment stays in its current state or token is not created, and cardholder is redirected back to merchant's error callback URL through HTTP POST request with LPB's response indicating error (see section 8.3.3).

{4} In case of *Payment Form* LPB sends update on payment status to merchant's System callback URL.

Merchant can choose to redirect cardholder to the following LPB's forms through HTTP POST request:

- **Payment Form**

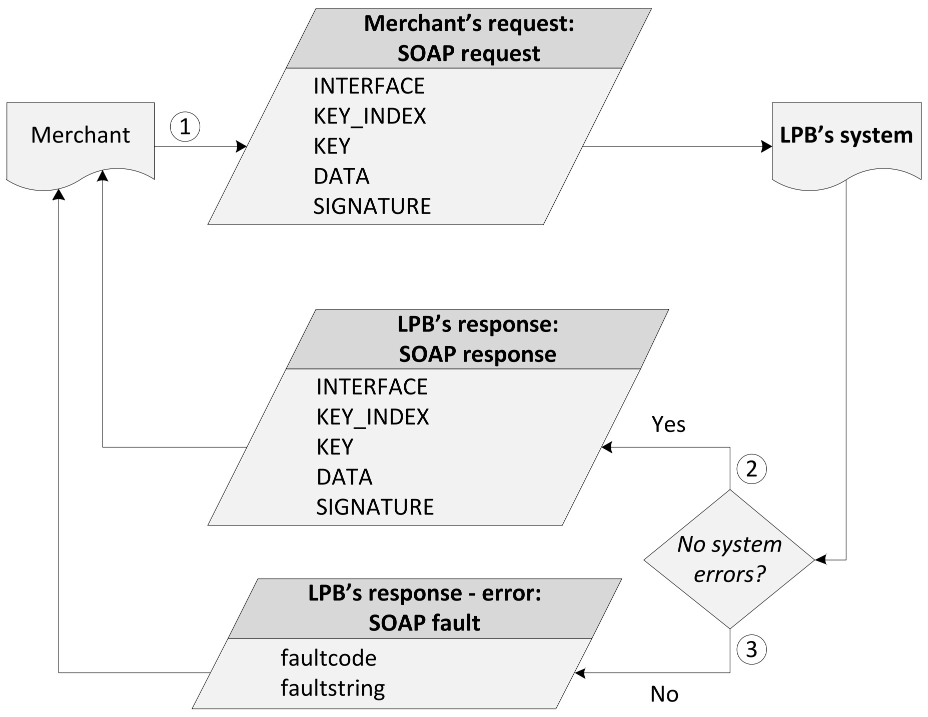
  Merchant initiates payment that starts at payment processing state *Initiated*. Cardholder is redirected from merchant's website to LPB system's *Payment Form* where he would have to enter his credit card data and approve payment details.

- **Token Registration Form**

  Merchant allows cardholder to register new token that is linked to his credit card data so that merchant can use this token for initiating new payments instead of requesting cardholder to provide credit card data again. Cardholder is forwarded from merchant's website to LPB system's *Token Registration Form* where he would have to enter his credit card data and approve token registration. However, this request can be sent only in case token feature is enabled.

8.2. API via SOAP

SOAP interface is available for all merchants; however, operations that require card data are available only for PCI DSS compliant merchants. Through SOAP requests merchants can create new payments or tokens, make deposits or reversals, get status of payments, etc. Merchant's request and LPB's response occur in the following sequence:



Schema 13. Data flow diagram of request and response for API via SOAP.

1 Merchant makes SOAP request, sending corresponding data to LPB (see section 8.3.1).

2 In case request ends successfully, merchant receives LPB's response (see section 8.3.2).

3 If system error happens, payment stays in its current state or token is not created, and merchant receives LPB's response indicating error (see section 8.3.3).

Merchant can choose to carry out the following SOAP operations that are sent to LPB through SOAP request.

- **Payment**

  After cardholder has entered his credit card data or chosen token that would be used, and approved payment details directly on merchant's website, merchant initiates payment that starts at payment processing state *Requested*.

  In case of 3D Secure payment processing mode where cardholder's credit card is enrolled in 3D Secure, merchant is also required to send to LPB results of cardholder's 3D Secure authentication. It is done through SOAP operation *Authenticate*.

- **Authenticate**

  Merchant sends LPB results of cardholder's 3D Secure authentication. Payment processing state is changed from *Requested* to *Approved* or *Declined*.

- **Deposit**

  Merchant confirms order proceeding, and cardholder's funds that were put on hold are now sent to LPB's batch for being transferred to merchant's account. Payment processing state is changed from *Approved* to *Deposited*.

  Note: Merchant can also confirm partial transfer of funds by approving only part of payment and removing hold from the remaining funds.

- **Reverse**

  Merchant requests for payment cancellation or reversal. It removes hold on cardholder's funds or returns transferred funds back to cardholder's account. Payment processing state is changed from *Approved* to *Cancelled* or from *Deposited* / *Processed* to *Reversed*.

  Note: Merchant can also request partial reversal by cancelling only part of payment and processing remaining funds as intended, while payment is in payment processing state *Deposited* or *Processed*.

- **GetPayment**

  Merchant requests current payment status. It can be requested any time.

- **RegisterToken**

  After cardholder has entered his credit card data and approved new token registration directly on merchant's website, merchant allows cardholder to register a new token that is linked to his credit card data so that cardholder can use this token for initiating new payments instead of entering credit card data again. However, this request can be sent only in case token feature is enabled.

8.3. External data structures

8.3.1. *Merchant's request*

Table 7. Merchant's request.

POST fields / SOAP request	Description
INTERFACE	Merchant's interface code. *It is a unique code that LPB system provides to merchant for his identification.*
KEY_INDEX	Index of merchant's RSA public key. *Index of merchant's RSA public key that he has uploaded to his account. Key has to be 2048 bit long and uploaded in .pem format.*
KEY	Base64 encoded RSA encrypted RC4 key. *Single-use RC4 key that is encrypted by Medoro payment gateway's RSA public key and then base64 encoded.*
DATA	Base64 encoded RC4 encrypted data. *Request data (see section 8.4.1) that is encrypted by merchant's RC4 key (see KEY field) and then base64 encoded.*
SIGNATURE	Base64 encoded RSA digital signature of unencrypted data. *RSA digital signature of unencrypted request data (see DATA field) that is made with merchant's RSA private key (see KEY_INDEX field) and then base64 encoded.*
CALLBACK	Default return URL. *Web address cardholder would be redirected to when payment creation ends. It occurs when payment state is changed to "Approved", "Unapproved" or "Declined".* ***This field refers only to form interface.***
ERROR_CALLBACK	Return URL in case of error. *Web address cardholder would be redirected to when transaction fails due to a system error (incorrect data, encryption/decryption error, etc). In this case payment remains in its current state or is not actually created.* ***This field refers only to form interface.***

8.3.2. LPB's response

Table 8. LPB's response.

POST fields / SOAP response	Description
INTERFACE	Merchant's interface code. *It is a unique code that LPB system provides to merchant for his identification.*
KEY_INDEX	Index of merchant's RSA public key. *Index of merchant's RSA public key that he used for data encryption of the payment in progress.*
KEY	Base64 encoded RSA encrypted RC4 key. *Single-use RC4 key that is encrypted by merchant's RSA public key and then base64 encoded.*
DATA	Base64 encoded RC4 encrypted data. *Response data (see section 8.4.2) that is encrypted by Medoro payment gateway's RC4 key (see KEY field) and then base64 encoded.*
SIGNATURE	Base64 encoded RSA digital signature of unencrypted data. *RSA digital signature of unencrypted response data (see DATA field) that is made with Medoro payment gateway's RSA private key (see KEY_INDEX field) and then base64 encoded.*

8.3.3. LPB's response in case of error

Table 9. LPB's response in case of error.

POST fields / SOAP fault	Description
ERROR_CODE / faultcode	Error code. *Code that identifies encountered system error. For error codes with descriptions see Appendix 1: Error codes.*
ERROR_MESSAGE / faultstring	Error message. *Description of encountered error, e.g. incorrect data, encryption/decryption error.*

8.4. Internal data structures

Request and response data should be sent in XML format and contain elements described in sections 8.4.1 and 8.4.2 where:

- Leftmost elements should be included in XML document root element: <data>. Other elements should be included in corresponding parent elements.
- In case of fields named "Choice", only one of the options must be present in XML document: elements that are included under *"Choice 1"* or *"Choice 2"*.
- Elements marked with "*****" are required to be included in requests (unless corresponding parent element is not mandatory), while in responses fields that are not marked as required in corresponding XSD schemas may or may not be included in XML depending on request and current payment state.

8.4.1. Request data

In order to carry out any operation, merchant is required to send LPB corresponding request data that is included in XML document with elements specified below.

8.4.1.1. Payment Form: Payment Initiation request data

Table 10. Payment Form: Payment Initiation request data.

Elements		Possible values; descriptions
<AutoDeposit>		Boolean (true or false) *True in case of SMS, false in case of DMS (see section 5). If this element is not included in XML, by default its value is set to "true".*
<Payment>*		
	<Mode>*	Unsigned short (3 or 5) *Number of payment processing mode (see 6.4).*
	<Descriptor>	String (20 characters), maximum length depends on merchant's name and city *Dynamic descriptor that would appear on cardholder's credit card statement for this transaction. However, this element is allowed to be included in XML only in case dynamic descriptor feature is enabled (see section 6.5).*
<Recurring>		*This element is allowed to be included in XML only in case recurring payments feature is enabled and cardholder wants to subscribe for recurring payments (see section 6.5).*

<Frequency>*	Unsigned integer, minimum is 1 day *Minimal number of days that have to pass between subsequent payments after recurring payment registration.*
<EndDate>*	Number (8 digits), YYYYMMDD *Date after which subscription will end and no new subsequent payments would be permitted.*
<Order>*	
<ID>*	String (50 characters) *Unique code for order identification.*
<Amount>*	Long *Payment amount in **minor** currency units (e.g. 1050 for 10.50 EUR). In case of negative amount Original credit payment will be initiated.*
<Currency>*	String (3 characters), ISO 4217 currency 3 char alpha code *Payment currency.*
<Description>*	String (65535 characters) *Order description that will be shown on Payment Form.*
<Card>	*This element should be included only in case cardholder wants to use credit card data from previously registered token (see section 6.5). On Payment Form cardholder would have to confirm the masked credit card data instead of entering it manually.*
<Token>*	String (50 characters) *Name of token to be used.*
<BillingAddress>	*Cardholder's billing address.*
<Name>	String (255 characters) *Name of a person who will receive bill or invoice.*
<Address>	String (255 characters) *Address where bill or invoice would be sent.*
<City>	String (40 characters)
<Country>	String (3 characters) *Country in ISO 3166-1 alpha 3 format.*
<PostIndex>	String (20 characters)
<Phone>	String (40 characters) *Contact phone number.*
<Email>	String (255 characters) *Contact email.*
<Notification>	String (65535 characters) *In case this element is included in XML, its content will be shown on Payment Form with a checkbox that cardholder would have to check in order to submit the form. It can be used for showing cardholder custom notifications or terms and conditions.*

Note: For XSD schema see file *requests\form\InitiateRequest.xsd* provided with this documentation.

Example of XML code

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<data>
   <AutoDeposit>false</AutoDeposit>
   <Payment>
      <Mode>3</Mode>
      <Descriptor>QWERTY</Descriptor>
   </Payment>
   <Recurring>
      <Frequency>30</Frequency>
      <EndDate>20150325</EndDate>
   </Recurring>
   <Order>
      <ID>45278bc0-7777-11e3-981f-0800200c9a66</ID>
      <Amount>5000</Amount>
      <Currency>EUR</Currency>
      <Description>Payment #1234354</Description>
   </Order>
   <Card>
      <Token>d9e90ef0-7777-11e3-981f-0800200c9a66</Token>
   </Card>
   <BillingAddress>
      <Name>John Griffith Chaney</Name>
      <Address>200 Fake Street</Address>
      <City>San Francisco</City>
      <Country>USA</Country>
      <PostIndex>CA 94102</PostIndex>
      <Phone>+1-202-555-0156</Phone>
      <Email>john.griffith@domain.dom</Email>
   </BillingAddress>
   <Notification>I agree with terms and conitions</Notification>
</data>
```

8.4.1.2. Token Registration Form: Token Registration request data

Table 11. Token Registration Form: Token Registration request data.

Elements	Possible values; descriptions
<Card>*	
<Token>	String (50 characters) *Name of the new token. If this field is not included in XML, UUID is generated instead.*

Note: For XSD schema see file *requests\form\RegisterTokenRequest.xsd* provided with this documentation.

Example of XML code

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<data>
  <Card>
    <Token>d9e90ef0-7777-11e3-981f-0800200c9a66</Token>
  </Card>
</data>
```

8.4.1.3. SOAP: Payment request data

Table 12. Payment request data.

Elements			Possible values; descriptions
<AutoDeposit>			Boolean (true or false) *True in case of SMS, false in case of DMS (see section 5). If this element is not included in XML, by default its value is set to "true".*
<Payment>*			
	<Mode>*		Unsigned short (2, 4, 6 or 9) *Number of payment processing mode (see 6.4).*
	<Descriptor>		String (20 characters), maximum length depends on merchant's name and city *Dynamic descriptor that would appear on cardholder's credit card statement for this transaction. However, this element is allowed to be included in XML only in case dynamic descriptor feature is enabled (see section 6.5).*
<Recurring>			*This element is allowed to be included in XML only in case recurring payments feature is enabled (see section 6.5).*
	Choice 1*		*These elements should be included in case cardholder wants to subscribe for a new recurring payment.*
		<Frequency>*	Unsigned integer, minimum is 1 day *Minimal number of days that have to pass between subsequent payments after recurring payment registration.*
		<EndDate>*	Number (8 digits), YYYYMMDD *Date after which subscription will end and no new subsequent payments would be permitted.*
	Choice 2*		*This element should be included in case merchant wants to initiate subsequent recurring payment.*

<ID>*	Unsigned long *Identification number of previously subscribed recurring payment that would be used for initiating new subsequent recurring payment.*
<Order>*	
<ID>*	String (50 characters) *Unique code for order identification.*
<Amount>*	Long *Payment amount in **minor** currency units (e.g. 1050 for 10.50 EUR). In case of negative amount Original credit payment will be initiated.*
<Currency>*	String (3 characters), ISO 4217 currency 3 char alpha code *Payment currency.*
<Description>*	String (65535 characters) *Order description that will be shown on Payment Form.*
<Card>	*This element should not be provided in case of mode 8 or 9.*
Choice 1*	*These elements should be included only in case cardholder wants to use credit card data from previously registered token (see section 6.5).*
<Token>*	String (50 characters) *Name of token to be used.*
<CSC>	Number (3 digits) *Card security code*
Choice 2*	
<Name>*	String (50 characters) *Name on cardholder's credit card.*
<Number>*	Number (10-19 digits) *Cardholder's credit card number.*
<Expiry>*	Number (4 digits), YYMM *Credit card expiration date.*
<CSC>	Number (3 digits) *Card security code. For Original credit transactions and Maestro cards this field is optional.*
<OCfromExistingPayment>	*This element should be provided only in mode 8.*
Choice 1*	
<PaymentID>	Unsigned long *Payment ID from previously conducted payment.*
Choice 2*	
<OrderID>	String (50 characters) *Order ID from previously conducted payment.*

<BillingAddress>		*Cardholder's billing address.*
	<Name>	String (255 characters) *Name of a person who will receive bill or invoice.*
	<Address>	String (255 characters) *Address where bill or invoice would be sent.*
	<City>	String (40 characters)
	<Country>	String (3 characters) *Country in ISO 3166-1 alpha 3 format.*
	<PostIndex>	String (20 characters)
	<Phone>	String (40 characters) *Contact phone number.*
	<Email>	String (255 characters) *Contact email.*
<RemoteAddress>*		String, IPv4 address (dotted-decimal notation) *Cardholder's IP address. If cardholder's IPV4 address is not available, or cardholder is not present, IP address of merchant's system should be passed instead.*

Note: For XSD schema see file *requests\soap\PaymentRequest.xsd* provided with this documentation.

Example of XML code (for modes 2,4 or 6)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<data>
  <AutoDeposit>false</AutoDeposit>
  <Payment>
    <Mode>4</Mode>
    <Descriptor>QWERTY</Descriptor>
  </Payment>
  <Recurring>
    <Frequency>30</Frequency>
    <EndDate>20150325</EndDate>
  </Recurring>
  <Order>
    <ID>45278bc0-7777-11e3-981f-0800200c9a66</ID>
    <Amount>5000</Amount>
    <Currency>EUR</Currency>
    <Description>Payment #1234354</Description>
  </Order>
  <Card>
    <Name>John Griffith Chaney</Name>
    <Number>4111111111111111</Number>
    <Expiry>1905</Expiry>
    <CSC>123</CSC>
  </Card>
  <BillingAddress>
    <Name>John Griffith Chaney</Name>
    <Address>200 Fake Street</Address>
    <City>San Francisco</City>
    <Country>USA</Country>
```

```xml
        <PostIndex>CA 94102</PostIndex>
        <Phone>+1-202-555-0156</Phone>
        <Email>john.griffith@domain.dom</Email>
    </BillingAddress>
    <RemoteAddress>192.168.2.100</RemoteAddress>
</data>
```

Example of XML code (for mode 9)

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<data>
    <AutoDeposit>false</AutoDeposit>
    <Payment>
        <Mode>9</Mode>
        <Descriptor>QWERTY</Descriptor>
    </Payment>
    <Recurring>
        <ID>12345</ID>
    </Recurring>
    <Order>
        <ID>45278bc0-7777-11e3-981f-0800200c9a66</ID>
        <Amount>5000</Amount>
        <Currency>EUR</Currency>
        <Description>Payment #1234354</Description>
    </Order>
    <BillingAddress>
        <Name>John Griffith Chaney</Name>
        <Address>200 Fake Street</Address>
        <City>San Francisco</City>
        <Country>USA</Country>
        <PostIndex>CA 94102</PostIndex>
        <Phone>+1-202-555-0156</Phone>
        <Email>john.griffith@domain.dom</Email>
    </BillingAddress>
    <RemoteAddress>192.168.2.100</RemoteAddress>
</data>
```

8.4.1.4. SOAP: Authenticate request data

Table 13. Authenticate request data.

Elements			Possible values; descriptions
Choice 1*			
	<Payment>*		
		<ID>*	Unsigned long *Unique code for payment identification.*
Choice 2*			
	<Order>*		
		<ID>*	String (50 characters) *Unique code for order identification.*
<D3D>*			
	<ACS>*		
		<PaRes>*	String (65535 characters) *Payment authentication response received from cardholder's card issuer bank's ACS.*

Note: For XSD schema see file *requests\soap\AuthenticateRequest.xsd* provided with this documentation.

Example of XML code

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<data>
  <Payment>
    <ID>12345678</ID>
  </Payment>
  <D3D>
    <ACS>
      <PaRes>PaRes</PaRes>
    </ACS>
  </D3D>
</data>
```

8.4.1.5. SOAP: Deposit request data

Table 14. Deposit request data.

Elements			**Possible values; descriptions**
Choice 1*			
	<Payment>*		
		<ID>*	Unsigned long *Unique code for payment identification.*
	<Order>		
		<DepositAmount>*	Unsigned long *Amount that would be transferred to merchant's account. Hold from remaining funds would be removed immediately. Deposit amount should be smaller than payment amount and larger than 0, written in **minor** currency units. In case it is not included in XML, full payment amount would be transferred.*
Choice 2*			
	<Order>*		
		<ID>*	String (50 characters) *Unique code for order identification.*
		<DepositAmount>	Unsigned long *Amount that would be transferred to merchant's account. Hold from remaining funds would be removed immediately. Deposit amount should be smaller than payment amount and larger than 0, written in **minor** currency units. In case it is not included in XML, full payment amount would be transferred.*

Note: For XSD schema see file *requests\soap\DepositRequest.xsd* provided with this documentation.

Example of XML code

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<data>
   <Payment>
      <ID>12345678</ID>
   </Payment>
   <Order>
      <DepositAmount>4000</DepositAmount>
   </Order>
</data>
```

8.4.1.6. *SOAP: Reverse request data*

Table 15. Reverse request data.

Elements			**Possible values; descriptions**
Choice 1*			
	<Payment>*		
		<ID>*	Unsigned long *Unique code for payment identification.*
	<Order>		
		<ReversalAmount>*	Unsigned long *This element can be included in XML if payment is in payment processing state Deposited or Processed. It indicates amount that would be returned to cardholder's account. Reversal amount should be less than payment amount and more than 0, written in **minor** currency units. If it is not included, full payment amount would be returned.*
Choice 2*			
	<Order>*		
		<ID>*	String (50 characters) *Unique code for order identification.*
		<ReversalAmount>	Unsigned long *This element can be included in XML if payment is in payment processing state Deposited or Processed. It indicates amount that would be returned to cardholder's account. Reversal amount should be less than payment amount and more than 0, written in **minor** currency units. If it is not included, full payment amount would be returned.*

Note: For XSD schema see file *requests\soap\ReverseRequest.xsd* provided with this documentation.

Example of XML code

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<data>
   <Order>
      <ID>45278bc0-7777-11e3-981f-0800200c9a66</ID>
      <ReversalAmount>3000</ReversalAmount>
   </Order>
</data>
```

8.4.1.7. SOAP: GetPayment request data

Table 16. GetPayment request data.

Elements			**Possible values; descriptions**
Choice 1*			
	<Payment>*		
		<ID>*	Unsigned long *Unique code for payment identification.*
Choice 2*			
	<Order>*		
		<ID>*	String (50 characters) *Unique code for order identification.*

Note: For XSD schema see file *requests\soap\GetPaymentRequest.xsd* provided with this documentation.

Example of XML code

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<data>
   <Order>
      <ID>45278bc0-7777-11e3-981f-0800200c9a66</ID>
   </Order>
</data>
```

8.4.1.8. SOAP: RegisterToken request data

Table 17. RegisterToken request data.

Elements	Possible values; descriptions
<Card>*	
<Token>	String (50 characters) *Name of the new token (see section 6.5). If this field is not included in XML, UUID is generated instead.*
<Name>*	String (50 characters) *Name on cardholder's credit card.*
<Number>*	Number (10-19 digits) *Cardholder's credit card number.*
<Expiry>*	Number (4 digits), YYMM *Credit card expiration date.*

Note: For XSD schema see file *requests\soap\RegisterTokenRequest.xsd* provided with this documentation.

Example of XML code

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<data>
   <Card>
      <Token>d9e90ef0-7777-11e3-981f-0800200c9a66</Token>
      <Name>John Griffith Chaney</Name>
      <Number>4111111111111111</Number>
      <Expiry>1905</Expiry>
   </Card>
</data>
```

8.4.2. Response data

There are two only types of response data that LPB can send to merchant. First type contains *Payment response data* and is sent in response to any of requests excluding ones that refer to token registration. Meanwhile second type contains *Token Registration response data* and is sent in response to token registration. Both of them can refer to requests through forms and to SOAP requests.

8.4.2.1. Payment response data

Table 18. Payment response data.

Elements		Possible values; descriptions
<AutoDeposit>		Boolean (true or false)
<Payment>*		
	<ID>*	Unsigned long *Unique code for payment identification.*
	<State>*	Unsigned short *Number of payment processing state (see section 6.6) that payment is currently in.*
	<Mode>	Unsigned short *Number of payment processing mode (see 6.4) that payment was made in.*
	<StartDate>	String, YYYY-MM-DD HH:MM:SS *Date and time (in EET – Eastern European Time) when payment was initiated.*
	<LastDate>	String, YYYY-MM-DD HH:MM:SS *Date and time (in EET – Eastern European Time) when payment was last updated.*
	<Descriptor>	String *Dynamic descriptor that appears on cardholder's credit card statement for this transaction (see section 6.5).*
<Recurring>		
	<ID>	Unsigned long *Unique code for recurring payments (see section 6.5) identification.*
	<Frequency>	Unsigned integer *Minimal number of days that have to pass between subsequent payments after subscription to recurring payments.*

	<EndDate>	Number, YYYYMMDD *Date after which subscription will end and no new subsequent payments would be permitted.*
<Order>		
	<ID>	String *Unique code for order identification.*
	<Amount>	Long *Payment amount in* ***minor*** *currency units (e.g. 1050 for 10.50 LVL).*
	<DepositAmount>	Unsigned long *Full or partial deposit amount in* ***minor*** *currency units.*
	<ReversalAmount>	Unsigned long *Full of partial reversal amount in* ***minor*** *currency units.*
	<Currency>	String, ISO 4217 currency 3 char alpha code *Payment currency.*
	<Description>	String *Order description.*
<Card>		
	<Token>	String *Name of token that is used (see section 6.5).*
	<Name>	String *Name on cardholder's credit card.*
	<Number>	String *Cardholder's masked credit card number.*
<BillingAddress>		
	<Name>	String
	<Address>	String
	<City>	String
	<Country>	String
	<PostIndex>	String
	<Phone>	String
	<Email>	String
<D3D>		*This section is included only in case of 3D Secure payment processing mode (see 6.4).*
	<XID>	String *Transaction identification number.*
	<TransactionDate>	String *Transaction date.*
	<MessageID>	String *Message ID.*
	<Enrolled>	String *Status on whether cardholder's credit card is enrolled in 3D Secure or not (see Table 19).*

	<Status>		String *Status on whether cardholder passed 3D Secure authentication or not (see Table 20).*
	<CAVV>		String *Cardholder's authentication verification value.*
	<ECI>		String *Electronic commerce indicator (see Table 20).*
	<ACS>		
		<URL>	String *Cardholder's card issuer bank's ACS web address where cardholder was sent for 3D Secure authentication.*
		<PaReq>	String *Payment authentication request.*
		<PaRes>	String *Payment authentication response received from cardholder's card issuer bank's ACS.*
<Auth>			
	<ApprovalCode>		String *Approval code, ISO 8583 field 38.*
	<ActionCode>		Number *Authorization action code, ISO 8583 field 39. For action codes with descriptions see Appendix 2: Action codes.*
	<Description>		String *Textual representation of action code.*
<Notification>			String *Custom notifications or terms and conditions that cardholder has approved when submitting payment.*
<RemoteAddress>			String, IPv4 address (dotted-decimal notation)

Note: For XSD schema see file *responses\PaymentResponse.xsd* provided with this documentation.

Table 19. 3D Secure Enrollment Statuses and Messages.

Status	Message	3D Secure	Payment processed?
Y	Authentication Available	Available	No
N	Cardholder Not Enrolled	Not available	Yes
U	Unable to Authenticate	Not available	Yes
E	any error message	Not available	No

Table 20. 3D Secure Authentication Statuses.

Visa ECI	MC ECI	Authentication status	Authentication message	Description	Payment processed?
05	02	Y	Authentication Successful	Cardholder was successfully authenticated.	Yes
06	01	A	Attempts Processing Performed	Authentication could not be performed but a proof of authentication attempt was provided.	Yes
-	-	N	Authentication Failed	Cardholder authentication failed.	No
07	01	U	Authentication Could Not Be Performed	Authentication could not be performed due to a technical error or other problem.	Yes
-	-	E	any error message here	An error occurred during the authentication process.	No

Example of XML code

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<data>
   <Payment>
      <ID>12345678</ID>
      <State>8</State>
      <Mode>5</Mode>
      <StartDate>2013-07-31 22:53:43</StartDate>
      <LastDate>2013-07-31 22:53:43</LastDate>
      <Descriptor>QWERTY</Descriptor>
   </Payment>
   <Recurring>
      <ID>12345</ID>
      <Frequency>30</Frequency>
      <EndDate>20150325</EndDate>
   </Recurring>
   <Order>
      <ID>45278bc0-7777-11e3-981f-0800200c9a66</ID>
      <Amount>5000</Amount>
      <DepositAmount>4000</DepositAmount>
      <ReversalAmount>3000</ReversalAmount>
      <Currency>EUR</Currency>
      <Description>Payment #1234354</Description>
   </Order>
   <Card>
      <Token>d9e90ef0-7777-11e3-981f-0800200c9a66</Token>
      <Name>John Griffith Chaney</Name>
      <Number>411111XXXXXX1111</Number>
   </Card>
   <BillingAddress>
      <Name>John Griffith Chaney</Name>
      <Address>200 Fake Street</Address>
      <City>San Francisco</City>
      <Country>USA</Country>
      <PostIndex>CA 94102</PostIndex>
      <Phone>+1-202-555-0156</Phone>
      <Email>john.griffith@domain.dom</Email>
   </BillingAddress>
   <Auth>
      <ApprovalCode>123ABC</ApprovalCode>
      <ActionCode>000</ActionCode>
      <Description>approved</Description>
   </Auth>
   <RemoteAddress>192.168.2.100</RemoteAddress>
</data>
```

8.4.2.2. Token Registration response data

Table 21. Token Registration response data.

Elements	Possible values; descriptions
<Card>*	
<Token>*	String *Name of the new token (see section 6.5).*
<Name>*	String *Name on cardholder's credit card.*
<Number>*	String *Masked credit card number of cardholder's credit card that would be used for making payments with use of token instead of entering card data.*

Note: For XSD schema see file *responses\RegisterTokenResponse.xsd* provided with this documentation.

Example of XML code

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<data>
  <Card>
    <Token>d9e90ef0-7777-11e3-981f-0800200c9a66</Token>
    <Name>John Griffith Chaney</Name>
    <Number>411111XXXXXX1111</Number>
  </Card>
</data>
```

9. GUI

9.1. GUI via web portal

Each merchant has account on *Medoro* payment gateway's web portal, where he can make certain actions, for instance, make deposits or reversals, fill in MO/TO form, view information about payments, get status of payments, get statistics on financial days, access merchant's profile, initiate an Original credit payment, etc.

10. Appendices

10.1. Appendix 1: Error codes

General error codes:

100000 — Internal server error
100001 — Lock timeout
200000 — Entity not found
300000 — Bad request
400000 — Integrity violation

Integrity violation:

400011 — Order ID must be unique
400014 — Token is already registered

Payment operations

200201 — Token not found
200301 — Directory service not found
200401 — Key not found. Either INTERFACE or KEY_INDEX is incorrect.
200601 — Recurring not found
505102 — Could not initiate payment, state flow violated
605103 — Could not initialize authentication of non—3d payment
505104 — Could not authenticate payment, state flow violated
605105 — Could not authenticate non—3d payment
505106 — Could not deposit payment, state flow violated
505107 — Could not cancel payment, state flow violated
705109 — Card name is required
705110 — Card number is required
705111 — Card expiry is required
305112 — Invalid card expiry date
305113 — Invalid card number
505115 — Could not reverse payment, state flow violated
505116 — Merchant interface is closed
605117 — Invalid currency
605118 — Original credit operations are not allowed on this interface
705120 — CSC is required
305122 — CSC is invalid

Request data:

710101 — Payment.Mode is required
710102 — Payment.Order is required
710103 — Payment.Order.ID is required
710104 — Payment.Order.Amount is required
710105 — Payment.Order.Currency is required
710106 — Payment.Order.Description is required
710109 — D3D.ACS.PaRes is required
510114 — Recurring expired
510115 — Too frequent recurring
710116 — Recurring.EndDate required
710117 — Recurring.Frequency required
610118 — Recurring.EndDate is incorrect
510119 — Recurring.EndDate must be in future
510120 — Recurring.Frequency must be greater than 0
710121 — Recurring.ID is required
710122 — RemoteAddress is required

Communication:

110201 — Invalid signature
710203 — Request is required
710204 — INTERFACE is required
710205 — KEYINDEX is required
710206 — KEY is required
710207 — DATA is required
710208 — SIGNATURE is required
110209 — XML deserialization failed
110210 — XML serialization failed
110301 — Data decryption failed
110302 — Data encryption failed
110401 — Signature verification failed
110402 — Signing failed

SOAP:

210501 — Payment not found
610602 — Could not authenticate payment, invalid Payment.Mode
710701 — Payment is required
610702 — Could not create payment, invalid Payment.Mode
111101 — Tokens are disabled

Forms:

211201 — Session not found
711202 — CALLBACK is required
711301 — Payment is required
611302 — Could not create payment, invalid mode
511401 — Could not update payment, state flow violated
111501 — Tokens are disabled
120102 — Session not found
120103 — Session invalid

Note: *this list is not final and is subjects to change.*

10.2. Appendix 2: Action codes

ISO 8583 authorization action codes

000	Approved
100	Decline (general, no comments)
101	Decline, expired card
102	Decline, suspected fraud
103	Decline, card acceptor contact acquirer
104	Decline, restricted card
105	Decline, card acceptor call acquirer's security department
107	Decline, refer to card issuer
108	Decline, refer to card issuer's special conditions
109	Decline, invalid merchant
110	Decline, invalid amount
111	Decline, invalid card number
113	Decline, unacceptable fee
114	Decline, no account of type requested
115	Decline, requested function not supported
116	Decline, not sufficient funds
118	Decline, no card record
119	Decline, transaction not permitted to cardholder
120	Decline, transaction not permitted to terminal
121	Decline, exceeds withdrawal amount limit
122	Decline, security violation
123	Decline, exceeds withdrawal frequency limit
124	Decline, violation of law
125	Decline, card not effective
129	Decline, suspected counterfeit card
180	Decline, by cardholders wish
900	Advice acknowledged, no financial liability accepted
901	Advice acknowledged, finansial liability accepted
902	Decline reason message: invalid transaction
903	Status message: re-enter transaction
904	Decline reason message: format error
907	Decline reason message: card issuer or switch inoperative
908	Decline reason message: transaction destination cannot be found for routing
909	Decline reason message: system malfunction
910	Decline reason message: card issuer signed off
911	Decline reason message: card issuer timed out
912	Decline reason message: card issuer unavailable
913	Decline reason message: duplicate transmission

914 — Decline reason message: not able to trace back to original transaction
918 — Decline reason message: no communication keys available for use
922 — Decline reason message: message number out of sequence
923 — Status message: request in progress
950 — Decline reason message: violation of business arrangement

Private action codes used in declined payments

181 — 3d secure authentication is currently unavailable *(Enrollment status = E)*
182 — 3d secure authentication failed *(Authentication status = N)*
940 — could not obtain agreement for this payment
941 — agreement suspended or closed
942 — original credit transaction is illegible for this PAN *(at least one deposited payment required)*
945 — card enrollment verification in 3d secure failed *(Incorrect VeRes; DS not available; merchant is not participating in 3d)*
946 — 3d secure field extraction from PaRes failed *(Incorrect PaRes; Incorrect certificate)*
948 — suspected fraud
949 — abandoned *(Timed out)*

Note: *this list is not final and is subject to change.*