First Data Latvia Payeezy^{SM} Integration Package

Administrator's Manual

Version 2.12.0.2

24-03-2017

Confidential

1 Contents

Change History

Version	Date	Modification
1.0	2009-01-12	Initial version
1.1	2009-01-12	Added information about *frameset* usage restriction
1.2	2009-03-03	Minor changes – installation paths
1.4	2009-08-13	Added information about DMS transaction reversal and business day closure
1.5	2010-06-14	Added information about transaction status AUTOREVERSED
1.6	2010-06-29	Added SMS transaction reversal description
1.7.	2011-10-28	Added information about transaction result requesting and partial reversal of transaction amount. Minor format/style changes of the document
2.09.8	2012-10-02	DMS1 transaction reversal time
2.09.9	2013-05-08	“Back to order” functionality realization description
2.10.0	2014-06-09	Changes related to the new version of ECOMM IMA. Transaction result request description update. ECOMM IMA purpose and installation description update. Minor text and format changes. Description of additional response fields to business day closure request added.
2.11.0	2016-02-04	• ECOMM brand name changed to Payeezy^{SM} • Changed Payeezy support email address from ecomm_support@firstdata.lv to payeezy@firstdata.lv • In chapter 3.3.5 Transaction Reversal (p.15) clarified the goal of reversal submission, and updated the timeframe of reversal submission (360 days after the original transaction) • Added chapters ○ 2.3 PayeezySM Integration Package Kickstart for Java solution (p.5) ○ 3.3.6 Transaction Refund (p.16) ○ 3.4 Instructions for PayeezySM Certificate and Keystore Creation (p.23) ○ 4 Result codes (p.27) ○ 5 Payment schema (UML sequence diagram) (p.29)
2.11.0.1	2016-04-27	• The document formatted according to the First Data newest template. • Cosmetic improvements • Updated ECOMM version number to 1.2.243
2.12.0.1	2017-02-06	• Added chapter 3.3.10 Merchant Disclosure requirements (p.21)
2.12.0.2.	2017-03-15	• Updated chapter 3.3.7 Business Day closure (p.17)

1 Introduction

1.1 Purpose

This document describes installation, configuration and usage of the Payeezy Integration Package (IP).

The Payeezy^{SM} IP is used to link the merchant to the Payeezy system and enable 3D-Secure transactions in the WWW environment, so merchant could accept an online payments with cards.

1.2 Audience

This document is intended to system developers and administrators.

1.3 Applicability

This document applies to the following software modules:

- Electronic Commerce System ECOMM Version 1.2.243
- Payeezy Integration Package (IP) 2.11.0

2 Preparation of Payeezy^{SM} Integration Package

2.1 System requirements

Component	Version	Notes
Sun Java Runtime Environment (JRE)	1.5. or newer	To be installed in accordance with http://docs.oracle.com/cd/E51849_01/gg-winux/GDRAD/java.htm
PHP	PHP 4.0.2 or newer	CURL library needed

2.2 Payeezy^{SM} Integration Package Installation

2.2.1 Configuration of Payeezy^{SM} Integration Package

To install PayeezySM IP to your server, extract the archive EcommMerchant_2.11.0.zip to merchant's web shop root directory. EcommMerchant_2.11.0 folder contains several subfolders:

- **php** – files required for integration with PHP solution
- **java** – files required for integration with Java solution (PayeezySM IP is in archive ecomm_merchant.jar)
- **c#** – files for integration with .NET/c# solution
- **CertGen** – contains .bat files for certificates creation on Windows
- **doc** – contains documentation

2.2.2 Configuration of Payeezy^{SM} Integration Package

1. Merchant should generate 2048-bit private key and certificate request for the test environment. Test certificate request should be signed in Payeezy^{SM} test environment (https://secureshop-test.firstdata.lv/report/) at Certificate signing section. Signed certificate and test Payeezy CA files will be sent in reply to the provided email address. With received

files merchant should create a keystore file, which is used to identify the merchant and establish SSL connection to Payeezy server:

- For php solution - merchantIdkeystore.pem
- For java solution – merchantIdkeystore.jks
- For c#/.NET solution merchantIdkeystore.der

2. Before to go live merchant should test PayeezySM IP integration by passing in PayeezySM test environment Test Plan section all necessary tests for SMS or/and DMS transaction type. Tests should be submitted and sent to the First Data

3. PayeezySM IP main settings for Java, PHP or .NET/c# solutions:
   a. For Java solution merchant.properties file (in /java directory) should be modified:

`bank.server.url`	Payeezy server address
`https.proxy.host`	HTTPS proxy server address (not mandatory)
`https.proxy.port`	HTTPS proxy server port (not mandatory)
`https.handler`	HTTPS protocol support library (not mandatory)
`https.cipher`	HTTPS connection encryption algorithm (not mandatory). Usually „SSL_RSA_WITH_RC4_128_MD5”
`keystore.file`	keystore file, which contains signed certificate and Payeezy CA (test or production, depending on environment)
`keystore.type`	keystore file format, JKS format must be used
`keystore.password`	password used during certificate creation
`connection.timeout`	time in seconds to establish connection to the Payeezy server. This parameter (depending on platform) can be only reduced

   b. For PHP solution config.php file (php/includes directory) should be modified:

`$ecomm_server_url`	Payeezy server address
`$ecomm_client_url`	redirect URL to card data input page
`$cert_url`	full path on merchant's server to the keystore file
`$cert_pass`	keystore's password
`$currency`	transaction currency code in ISO 4217 format
`$db_user`	MySQL database user name
`$db_pass`	MySQL database password
`$db_host`	MySQL database host
`$db_database`	MySQL database name

`$db_table_transaction`	MySQL database table name for transactions
`$db_table_batch`	MySQL database table name for business day totals
`$db_table_error`	MySQL database table for errors

c. For .NET/c# solution Program.cs (c#/src directory) file should be used:

`DERCertFilePath`	full path on merchant's server to certificate file in DER format

4. Merchant should prepare dynamic HTML page (cardinfo.html) for card data input and related files. Default templates already available at PayeezySM test environment Templates (cardinfo) section. More information can be found in chapter 3.3.9 Card data input page – cardinfo.html (p.20).

5. Merchant server IP address and return URLs for tests should be specified at Payeezy^{SM} test environment Merchant section:
   - **returnOkUrl** – client will be redirected to this address after the 3D Secure authentication and the transaction (regardless of the result).
   - **returnFailUrl** – client will be redirected to this address in the case of a technical failure in the Payeezy system.

6. When tests are passed and results are sent to First Data, the merchant should create and send to Payeezy Support team (payeezy@firstdata.lv) production certificate request, merchant's production server IP address and return URLs. Payeezy Support team will sign the request and reply with signed production certificate and production Payeezy CA files, so merchant could create production keystore.

   Signed certificate will be valid for 2 years. Next time production certificate request can be signed without Payeezy Support help at https://secureshop.firstdata.lv:8443/certreq/req, to access the link, valid production certificate in P12 format should be installed to internet browser. If production certificate is expired, new production certificate request should be sent to Payeezy Support again.

2.3 PayeezySM Integration Package Kickstart for Java solution

2.3.1 Download and install Java Runtime Environment (JRE) 7

Full Installation instructions and system requirements for Solaris, Microsoft Windows and Linux OS are available at: http://docs.oracle.com/javase/7/docs/webnotes/install/index.html

1. Download JRE
   You can download JRE from
   http://www.oracle.com/technetwork/java/javase/downloads/jre7-downloads-1880261.html

2. Install JRE from RPM or exe file, to `/usr/java/jre7/` (Linux) or `C:\Program Files\Java\jre7` (Windows)

3. Update the PATH variable

Windows	Edit the system environment variables in Windows: Add the location of bin folder of JRE installation for PATH in User Variables and System Variables. A typical value for PATH is: `C:\Program Files\Java\jre7\bin`
Linux	put line "`export PATH=$PATH:/usr/java/jre7/bin`" to ~/.bash_profile file

2.3.2 Install the Payeezy^{SM} Integration Package

Full Installation instructions can be found above in chapter 2.2 Payeezy^{SM} Integration Package Installation (p.3).

2.3.3 Test the application from command line

This chapter provides an example for Java solution.

2.3.3.1 Navigate to Payeezy IP directory

Windows

```
cd c:\ecomm
```

Linux

```
cd /usr/java/ecomm
```

2.3.3.2 Make SMS transaction

1. Register SMS transaction:
   Execute:
   ```
   java -jar ecomm_merchant.jar merchant.properties -v 10 978 127.0.0.1
   ```
   The result:
   ```
   TRANS_ID: WMZlrBOaxKUp7NHeHSooMOIvYKU=
   ```

2. Replace non-allowed HTTP symbols by URLencode. Let's denote the original transaction id value by <transaction id>, and URLencoded value by <transaction id encoded> (in the example above it would be WMZlrBOaxKUp7NHeHSooMOIvYKU%3D)

3. Open URL, which is created as follows:
   ```
   https://secureshop-
   test.firstdata.lv/ecomm/ClientHandler?trans_id=<transaction id
   encoded>
   ```

4. Submit the card information

5. Check the transaction result:
   ```
   java -jar ecomm_merchant.jar ecomm.merchant -c <transaction id>
   ```

```
<ip_address>
```

Note, that this action is needed to confirm the payment, see 3.3.4 Request for Transaction Results (p.13).

For example, the request might be

```
Java –jar 7comm_merchant.jar ecomm.merchant -c
WMZlrBOaxKUp7NHeHSooMOIvYKU= 127.0.0.1
```

and the result:

```
RESULT: OK
RESULT_CODE: 000
3DSECURE: OK
RRN: 706400119988
APPROVAL_CODE: 002469
CARD_NUMBER: ****************
```

2.3.3.3 Make DMS transaction

1. Register DMS transaction (authorization):
   Execute:
   ```
   java -jar ecomm_merchant.jar merchant.properties -a 10 978 127.0.0.1
   ```

   The result:
   ```
   TRANS_ID: qDSx7GYBWkqmKr/VCh5sYM68txM=
   ```

2. Replace non-allowed HTTP symbols by URLencode. Let's denote the original transaction id value by <transaction id>, and URLencoded value by <transaction id encoded> (in the example above it would be qDSx7GYBWkqmKr/VCh5sYM68txM%3D)

3. Open URL, which is created as follows:
   ```
   https://secureshop-
   test.firstdata.lv/ecomm/ClientHandler?trans_id=<transaction id
   encoded>
   ```

4. Submit the card information

5. Check the transaction result similarly as in the chapter 2.3.3.2 Make SMS transaction (p.6). This is optional, if the DMS transaction is executed not more than 3 minutes after the DMS registration.

6. Execute DMS transaction.
   In command line type:
   ```
   java -jar ecomm_merchant.jar merchant.properties -t <transaction id>
   ```

```
10 428 127.0.0.1
```

In the example:

```
java -jar ecomm_merchant.jar merchant.properties -t
qDSx7GYBWkqmKr/VCh5sYM68txM=  10 428 127.0.0.1
```

The result:

```
RESULT: OK
RESULT_CODE: 000
RRN: 706400119993
APPROVAL_CODE: 143828
CARD_NUMBER: ****************
```

3 Integration of Payeezy^{SM} to the merchant solution

3.1 SMS and DMS transaction types

3.1.1 SMS

SMS transactions are performed with the command –v. In the PHP solution it is the function startSMSTrans(). When an SMS transaction is performed, the money is debited from cardholder's account immediately.

3.1.2 DMS

DMS transactions are authorised with the command –a. In the PHP solution it is the function startDMSAuth(). When this authorization request is made, the money in cardholder's account is reserved (blocked).

DMS transactions are approved with the command –t. In the PHP solution it is the function makeDMSTrans(). When this request is executed, the reserved (blocked) money is debited from client's account. Merchant shall execute DMS transaction:

- When goods are shipped to client but not later 30 days from command –a/ startDMSAuth() was made
- If goods are not be shipped to client (for example goods are delivered electronically) then not later than 3 days from command –a/ startDMSAuth() was made

3.2 General procedure

1. Client has selected product and is ready to pay for the purchase. When the 'checkout' button/link is clicked, the management is passed to merchant's solution.
2. Merchant registers the transaction in the Payeezy system (specifying amount, currency, client's IP address, brief description of transaction (not mandatory), and receives the transaction identifier in the reply.
3. Client (with transaction identifier specified) is redirected to the Payeezy payments server for entering the card data in accordance with the form template delivered by the merchant (cardinfo.html).
   When the card data is entered, data is verified and result is generated. If merchant supports 3D Secure, once card data is entered, client authentication takes place as part of 3D Secure. The results of authentication are communicated to Payeezy system.
4. Client is redirected back to the merchant site (with transaction identifier indicated).

5. Merchant, by using received transaction identifier, requests the transaction result from Payeezy.
6. In case of DMS transaction, an additional transaction (DMS2) should be performed in order to receive the money from the client (command –t, in PHP it is the function makeDMSTrans()).
7. Merchant is able to reverse the transaction, if necessary.
8. Merchant must send business day closure request once per day to the Payeezy server.

3.3 Integration

To connect to Payeezy payment server you can use PayeezySM Integration Package in several ways:

1. By calling Java archive ecomm_merchant.jar from the command line, as it is shown in the chapter 2.3 PayeezySM Integration Package Kickstart for Java solution (p.5).

2. By calling directly class lv.konts.ecomm.merchant.Merchant class service methods. Configuration file name has to be assigned to a Merchant class when this class is being created. If Configuration file name is incorrect then error message appears.

JAVA example:

```
Merchant merchant;
  try
  {
    merchant = new Merchant(propFile);
  } catch (ConfigurationException e)
  {
    System.err.println("error: " + e.getMessage());
    return;
  }

  String result = merchant.sendTransData(amount, currency,
client_ip, description);
```

PHP example:

```
$merchant = new Merchant($ecomm_url, $cert_url, $cert_pass, 1);

$resp = $merchant -> startDMSAuth(
$amount,
$currency,
$client_ip_addr,
$description,
$language
);
echo "$resp \n";
```

3.3.1 Execution of SMS transaction

3.3.1.1 Command line parameters

-v	identifies transaction registration request
amount	transaction amount in minor values, mandatory (up to 12 digits)
currency	transaction currency code, mandatory (ISO 4217) (3 digits)
client_ip_addr	client's IP address, mandatory (15 characters)
description	brief description of transaction, not mandatory, should be urlencoded (up to 125 characters)
language	authorization language identifier, not mandatory (up to 32 characters)

3.3.1.2 Java method call

```
public String
startSMSTrans(String amount, String currency, String ip, String desc,
String language)

// old method for backward compatibility
public String
sendTransData(String amount, String currency, String ip, String desc,
String language)
```

3.3.1.3 PHP method call

```
$merchant = new Merchant($ecomm_server_url, $cert_url, $cert_pass, 1);
$resp = $merchant -> startSMSTrans($amount, $currency, $ip, $description,
$language);
```

3.3.1.4 Response

The format

```
 TRANSACTION_ID: <trans_id>
```

where

trans_id	transaction identifier (28 characters base64 encoding)

In the case of an error, the returned symbol string starts with 'error:'

Response example:

```
TRANSACTION_ID: bAt6JLX52DUbibbzD9gDFl5Ppr4=
```

3.3.2 DMS transaction authorization (DMS1)

3.3.2.1 *Command line parameters*

`-a`	identifies transaction registration request
`amount`	transaction amount in minor values, mandatory (up to 12 digits)
`currency`	transaction currency code, mandatory (ISO 4217) (3 digits)
`client_ip_addr`	client’s IP address, mandatory (15 characters)
`description`	brief description of transaction, not mandatory, should be urlencoded (up to 125 characters)
`language`	authorization language identifier, not mandatory (up to 32 characters)

3.3.2.2 *Java method call*

```
public String
startDMSAuth(String amount, String currency, String ip, String desc, String
language)
```

3.3.2.3 *PHP method call*

```
$merchant = new Merchant($ecomm_server_url, $cert_url, $cert_pass, 1);
$resp = $merchant -> startDMSAuth($amount, $currency, $ip, $description,
$language);
```

3.3.2.4 *Response*

```
TRANSACTION_ID: <trans_id>
```

where

`trans_id`	transaction identifier (28 characters base64 encoding)

In the case of an error, the returned symbol string starts with ‘error: ‘

Result example:

```
TRANSACTION_ID: bAt6JLX52DUbibbzD9gDFl5Ppr4
```

3.3.3 DMS transaction execution (DMS2)

3.3.3.1 *Command line parameters*

`-t`	identifies transaction registration request
`auth_id`	identifies authorization for which the financial transaction is performed
`amount`	transaction amount in minor values, mandatory (up to 12 digits)

currency	transaction currency code, mandatory (ISO 4217) (3 digits)
client_ip_addr	client's IP address, mandatory (15 characters)

3.3.3.2 Java method call

```
public String
makeDMSTrans(String auth_id, String amount, String currency, String ip)
```

3.3.3.3 PHP method call

```
$merchant = new Merchant($ecomm_server_url, $cert_url, $cert_pass, 1);
$resp = $merchant -> makeDMSTrans($auth_id, $amount, $currency, $ip);
```

3.3.3.4 Response

```
  RESULT: <result>
  RESULT_CODE: <result_code>
  RRN: <rrn>
  APPROVAL_CODE: <app_code>
```

result	Transaction result. Possible values:

OK	successful transaction
FAILED	transaction failed

result_code	result code returned from authorization system (3 digits)
rrn	retrieval reference number returned authorization system (12 characters)
app_code	approval code returned from authorization system (max 6 characters)

Fields RESULT_CODE are informative only. Fields RRN and APPROVAL_CODE are only shown if the transaction is successful and they are informative only in order to make tracking of the transaction in the authorization system easier. The decision on whether the transaction has been successful or failed may be made only on the basis of the value of the field RESULT.

In the case of an error, the returned symbol string starts with 'error:'

In the case of a warning, the returned symbol string starts with 'warning:'

Result example:

```
RESULT: OK
RESULT_CODE: 000
RRN: 123456789012
APPROVAL_CODE: 123456
```

3.3.4 Request for Transaction Results

SMS transaction result should be requested within 3 minutes after client returns after payment to the merchant page. Keep in mind, if transaction result is not requested within 3 minutes, transaction will be automatically reversed.

In case of DMS1 transaction you also should request transaction result within 3 minutes. But if after DMS1 within 3 minutes you perform DMS2 transaction, you don't need to request additionally DMS1 transaction result. For DMS2 transaction result is returned automatically.

3.3.4.1 Command line parameters

`-c`	identifies transaction registration request
`trans_id`	transaction identifier, mandatory (28 characters)
`client_ip_addr`	client's IP address, mandatory (15 characters)

3.3.4.2 Java method call

```
public String
getTransResult(String trans_id, String ip)
```

3.3.4.3 PHP method call

```
$merchant = new Merchant($ecomm_server_url, $cert_url, $cert_pass, 1);
$resp = $merchant -> getTransResult(urlencode($trans_id), $client_ip_addr);
```

3.3.4.4 Response

```
RESULT: <result>
RESULT_CODE: <result_code>
3DSECURE: <3dsecure>
AAV: <aav>
RRN: <rrn>
APPROVAL_CODE: <app_code>
```

`result` Request result. Possible values:

`OK`	successful transaction
`FAILED`	transaction failed
`CREATED`	transaction just registered in the system

PENDING	transaction not yet performed
DECLINED	transaction declined because its ECI value is included in the list of blocked ECI values (Payeezy server configuration)
REVERSED	transaction already reversed
AUTOREVERSED	transaction is automatically reversed if the result was not requested within specified time (3 min.)
TIMEOUT	transaction declined due to timeout

result_code	result code returned from authorization system (3 digits)
3dsecure	3D Secure status:

AUTHENTICATED	successful 3D Secure authorization
DECLINED	3D Secure authorization is unsuccessful
NOTPARTICIPATED	Non-participation on 3D scheme
NO_RANGE	Not Enrolled Transactions
ATTEMPTED	Valid authentication attempt
UNAVAILABLE	Authentication Unavailable
ERROR	3-D Secure Errors
SYSERROR	System Errors
UNKNOWNSCHEME	Unknown Card Schemes
FAILED	status after timeout

rrn	retrieval reference number returned authorization system (12 characters)
approval_code	approval code returned from authorization system (max 6 characters)
card_number	fully or partly masked card number

Fields RESULT_CODE and 3DSECURE are informative only (may be not showed). Fields RRN and APPROVAL_CODE are only shown if the transaction is successful and they are informative only on order to make tracking of the transaction in the authorization system. The decision whether the transaction has been successful or failed may be made only based on the value of the field RESULT.

Note. Transaction result should NOT be requested unless a client returns to merchant's returnOkUrl/returnFailUrl..In case if client does not return to returnOkUrl/ returnFailUrl then result may be requested after 13 minutes.

In the case of an error, the returned symbol string starts with 'error: '

In the case of a warning, the returned symbol string starts with 'warning:'

Result example:

```
  RESULT: OK
  RESULT_CODE: 000
  3DSECURE: ATTEMPTED
  RRN: 123456789012
  APPROVAL_CODE: 123456
  CARD_NUMBER: 555555*******4444
```

3.3.5 Transaction Reversal

Transaction reversals are used to negate or cancel a Transaction when there has been a **technical error**.

1. DMS authorization (DMS1) could be reversed only during first 72 h from authorization registration (Step 1). After 72 h First Data system will decline authorization reversal with response code 914.

2. SMS and DMS transaction can be reversed independently if business day is closed or not.

3. DMS and SMS reversal can be done 360 days from transaction date. After 360 days system will reject reversals. Reversal can be sent just one time for each transaction.

3.3.5.1 Command line parameters

-r	identifies transaction registration request
trans_id	transaction identifier, mandatory (28 characters)
amount	reversal amount in minor values, mandatory (up to 12 characters) . Merchant may or may not be able to return partial amount depending on capabilities of its acquirer/processor. Please contact your acquirer/processor to clarify this capability.

3.3.5.2 Java method call

```
public String
reverse(String trans_id, String amount)
```

3.3.5.3 PHP method call

```
$merchant = new Merchant($ecomm_server_url, $cert_url, $cert_pass, 1);
$resp = $merchant -> reverse($trans_id, $amount);
```

3.3.5.4 Response

```
RESULT: <result>
RESULT_CODE: <result_code>
```

where

`result`	Request result. Possible values:

OK	Transaction is reversed
REVERSED	Transaction was already reversed
FAILED	transaction reversing failed

`result_code`	result code returned from authorization system (3 digits)

In the case of an error, the returned symbol string starts with ‘error: ‘

In the case of a warning, the returned symbol string starts with ‘warning:’

Result example:

```
RESULT: OK
RESULT_CODE: 400
```

3.3.6 Transaction Refund

According to scheme rules Refunds are used for customer service or legal reasons: to credit a Cardholder’s account for returned products or cancelled services, or for price adjustments, related to a prior purchase. MasterCard explicitly states, that refunds can be used only for these purposes.

3.3.6.1 Command line parameters

-k	identifies transaction refund request
trans_id	transaction identifier, mandatory (28 characters)
amount	optional parameter – refund transaction amount in fractional units (up to 12 characters) . If not specified, full original transaction amount will be refunded.

3.3.6.2 Java method call

```
public String
refund(String trans_id, String amount)
```

3.3.6.3 PHP method call

```
$merchant = new Merchant($ecomm_server_url, $cert_url, $cert_pass, 1);
$resp = $merchant -> refund($trans_id, $amount);
```

3.3.6.4 Response

Output format

```
RESULT: <result>
RESULT_CODE: <result_code>
REFUND_TRANS_ID: <refund_transaction_id>
```

- In the case of an error, the returned symbol string starts with ‘error: ‘
- In the case of a warning, the returned symbol string starts with ‘warning:’

Parameters:

`result`	Possible values:
	`OK` — successful refund transaction
	`FAILED` — refund transaction failed
`result_code`	result code returned from authorization system (3 digits)
`refund_transaction_id`	Refund transaction identifier – applicable for obtaining refund payment details or to request refund payment reversal

Response example:

```
RESULT: OK
RESULT_CODE: 400
```

3.3.7 Business Day Closure

Business day closure is necessary to close the last open batch for the merchant. Until batch is open, information about merchant transactions will not be sent to the First Data and processed by banks. Business day closure must be done once per day. Next batch opens with the first successful transaction.

Merchants responsibility is to compare data:

- Which are included in Business Day closure result/report created after business day closing

- And Merchants data about provided service or products to clients for the same business day
- And bank data, which are included in settlement from bank for the same business day

In case of any discrepancies, please contact First Data Latvia support team:

- payeezy@firstdata.lv
- +371 67092 597

3.3.7.1 Command line parameters

-b	identifies business day closure request

3.3.7.2 Java method call

```
public String
  closeDay()
```

3.3.7.3 PHP method call

```
$merchant = new Merchant($ecomm_server_url, $cert_url, $cert_pass, 1);
$resp = $merchant -> closeDay();
```

3.3.7.4 Response

```
  RESULT: <result>
  RESULT_CODE: <result_code>
  FLD_074: <fld_074>
  FLD_075: <fld_075>
  FLD_076: <fld_076>
  FLD_077: <fld_077>
  FLD_086: <fld_086>
  FLD_087: <fld_087>
  FLD_088: <fld_088>
  FLD_089: <fld_089>
```

where

`result` Request result. Possible values:

OK	Business day is closed
FAILED	Business day closing failed

`result_code`	result code returned from authorization system (3 digits)
`FLD_074`	the number of credit transactions (up to 10 digits), shown only if result_code begins with 5
`FLD_075`	the number of credit reversals (up to 10 digits), shown only if result_code begins with 5
`FLD_076`	the number of debit transactions (up to 10 digits), shown only if result_code begins with 5
`FLD_077`	the number of debit reversals (up to 10 digits), shown only if result_code begins with 5
`FLD_086`	total amount of credit transactions (up to 16 digits), shown only if result_code begins with 5
`FLD_087`	total amount of credit reversals (up to 16 digits), shown only if result_code begins with 5
`FLD_088`	total amount of debit transactions (up to 16 digits), shown only if result_code begins with 5
`FLD_089`	total amount of debit reversals (up to 16 digits), shown only if result_code begins with 5

In the case of an error, the returned symbol string starts with 'error: '

Result example:

```
  RESULT: OK
  RESULT_CODE: 500
  FLD_074: 0
  FLD_075: 8
  FLD_076: 464
  FLD_077: 0
  FLD_086: 0
  FLD_087: 151100
  FLD_088: 24461939
  FLD_089: 0
```

3.3.8 Client redirection

Clients can be redirected (to enter card data) to the URL specified by the bank by using both the GET and POST methods. It is important that upon redirecting the variable trans_id that contain the identifier of the transaction to be paid is transferred. (It has to be taken into account that trans_id can contain '+', '=' and '/' that, prior to sending, must be replaced with web-environment-friendly strings (e.g. '=' with '%3D'. In Java environment, it can be done by using the method URLEncoder.encode, in PHP environment urlencode()). When redirecting, additional parameters can be sent. Such parameters will be sent back to the merchant, when redirecting the client back to the merchant's website, parameters can be received with POST method.

Example of automatic redirection with the POST method, using JavaScript:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
```

```
"http://www.w3.org/TR/html4/strict.dtd">

<html>
<head>
<title>Merchant example post template to Payeezy</title>
</head>
<BODY onload="javascript:document.returnform.submit()">
<form name="returnform" action="%%post_url%%" method="POST">
  <input type="hidden" name="trans_id" value="%%trans_id%%">

<!-- To support javascript unaware/disabled browsers -->
<noscript>
    <center>Please click the submit button below.<br>
    <input type="submit" name="submit" value="Submit"></center>
</noscript>
</form>

</body>
</html>
```

To provide customer return from cardinfo.html to order list functionality, it's allowed to use the following code:

```
<a href="#" onclick="javascript:history.go(-1)">Back to order</a>
<a href="#" onclick="javascript:history.go(-2)">Back to order</a>
<a href="#" onclick="javascript:history.go(-3)">Back to order</a>
```

3.3.9 Card data input page – cardinfo.html

After selection and confirmation of goods, merchant's clients will be redirected to the card data input page, which is dynamically generated HTML page form. Cardinfo.html page design can be customized according merchant website design. IT can be done in Payeezy^{SM} test environment in Templates (cardinfo) section. The Payeezy server will recognise the following symbol rows in the template:

Matcher variable	**Replaced with HTML code**
`%%formdef%%`	Opening tags for the card input form `<form action=<url> method="post" onSubmit="return FormValidator(this)"> <input type="hidden" name="trans_id" value="<trans_id>" readonly/>`
`%%cardname%%`	Input field for the cardholder's name `<input type="text" name="cardname" size="19" maxlength="50"/>`
`%%cardnr%%`	Input field for the card number

	`<input type="text" name="cardnr" size="19" maxlength="19"/>`
`%%expmonth%%`	Input field for the card's expiry date month `<input type="text" name="validMONTH" size="2" maxlength="2"/>`
`%%expyear%%`	Input field for the card's expiry date year `<input type="text" name="validYEAR" size="2" maxlength="2"/>`
`%%cvc2%%`	Input field for the card's CVC2 `<input type="text" name="cvc2" size="3" maxlength="3"/>`
`%%amount%%`	Transaction amount (read only)
`%%ccyalpha%%`	Transaction currency (read only)
`%%description%%`	Transaction description sent by the merchant to the Payeezy server (read only)

Mandatory form validator JavaScript must be included in cardinfo.html:

```
<script src="/template/javascripts/formvalidator_en.js"></script>
```

where 'en' is an interface language.

Merchants are not allowed to integrate payment page into an IFRAME / FRAMESET.

NOTE! It is not allowed to include self-provided Javascript, event handling attributes (e.g. "onclick","onmouseover", etc.) and external links – all of these elements will be removed upon uploading them to Payeezy server.

3.3.10 Merchant Disclosure requirements

MasterCard and Visa have revised their rules and mandated[1] that e-commerce merchants must display the address, including the country, of their permanent establishment (the fixed place of business through which an e-commerce or mail / phone order merchant conducts its business) on their website, regardless of website or server locations.

Merchants must display their merchant address on the checkout screen used to present the final transaction amount, or within the sequence of web pages the cardholder accesses during the checkout process.

Merchants must clearly disclose to the cardholder the identity of the merchant at all points of interaction, including:

- The merchant's name, so that cardholders can easily distinguish the merchant from another party, such as a supplier of products or services to the merchant
- The merchant's location (physical address), to allow cardholders to easily determine, among other things, whether the transaction will be a domestic transaction or a cross-border transaction. Merchants must disclose their merchant location before prompting the cardholder to provide card information
- The merchant name and location displayed to the cardholder must be the same as the name they provide for both authorization and clearing
- Merchants must ensure that cardholders can easily understand that the merchant is responsible for the transaction, including delivery of the goods (whether physical or digital) or

[1] https://www.visaeurope.com/media/images/providing-the-proper-location-of-your-merchant-business-vbs-73-40136.pdf

provision of the services that are the subject of the transaction, and for customer service and dispute resolution, all in accordance with the terms applicable to the transaction

Merchant Location—Examples of e-commerce Merchant Disclosure.

The following examples illustrate an acceptable disclosure of merchant location for a card not present merchant:

- This online merchant is located in [insert country name and address]
- This is a [insert country name] retail site located in [address]
- This is a [insert country name] retail site located in [address], non [insert country name] card-holders may be subject to international fees

3.3.11 Localization

Starting from Payeezy IP version 2.08, Payeezy supports user interface in a number of languages. This is ensured by creating a number of html template sets where the visual contents are provided in the language as selected by the user.

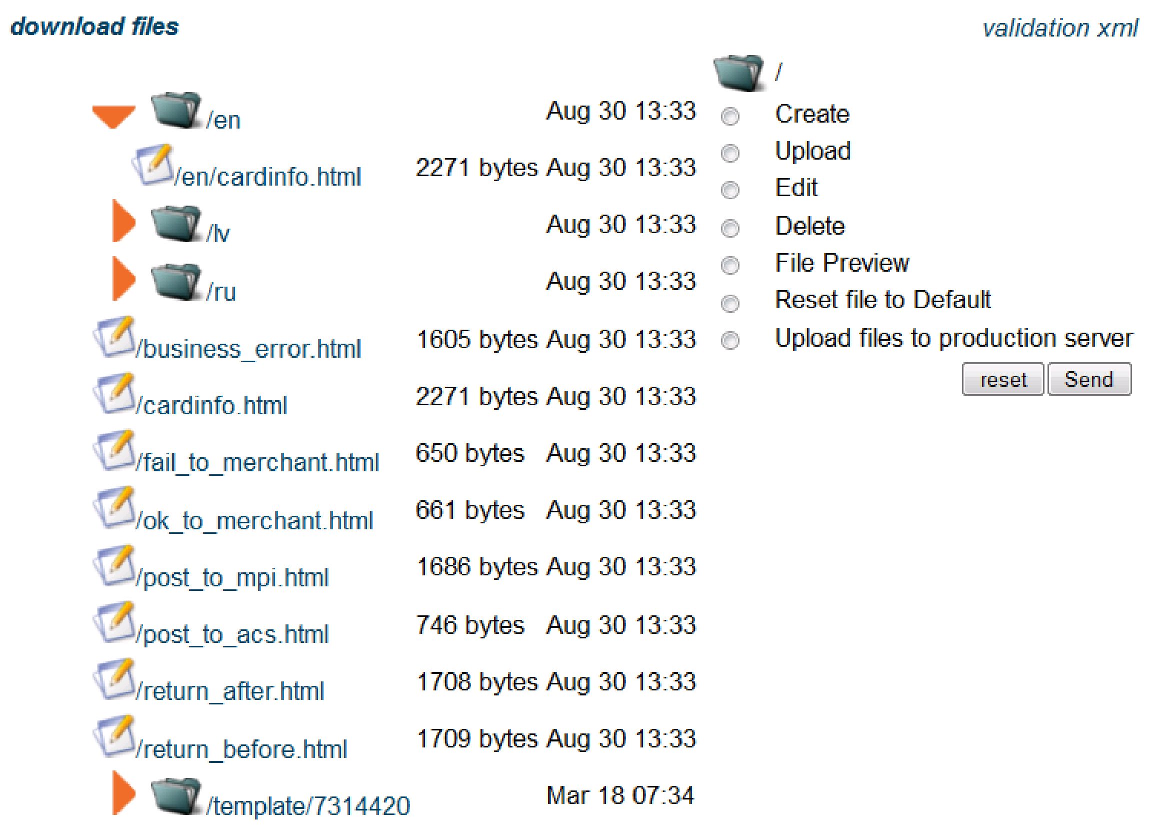
To make it possible to display cardinfo.html page for client in preferred language, different languages folders can be used for related cardinfo.html page.

On the server, every language may have one identifier that does not exceed 32 ASCII symbols. The language identifier may only contain lowercase letters, digits and the underscore sign (_) in ASCII encoding. When making transaction language parameter must be the same value as folder name on the server where "cardinfo.html" is located. For example, language=en (See Picture 1.).

If the Payeezy server does not recognise the language selected by the merchant (required language templates have not been set), the user interface will use the default templates.

If Payeezy IP is called from a command line, also the description parameter should be set if the language parameter is set. This is because both the description and the language identifiers are not mandatory transaction parameters. If the description is not required but the language identifier is required, the description is to be specified as an empty row ("").

The language differentiation functionality is available only in the end-user's interface. For business day closure and reversals, no language selection is available.



download files

```
/en                                   Aug 30 13:33
  /en/cardinfo.html        2271 bytes Aug 30 13:33
/lv                                   Aug 30 13:33
/ru                                   Aug 30 13:33
/business_error.html       1605 bytes Aug 30 13:33
/cardinfo.html             2271 bytes Aug 30 13:33
/fail_to_merchant.html     650 bytes  Aug 30 13:33
/ok_to_merchant.html       661 bytes  Aug 30 13:33
/post_to_mpi.html          1686 bytes Aug 30 13:33
/post_to_acs.html          746 bytes  Aug 30 13:33
/return_after.html         1708 bytes Aug 30 13:33
/return_before.html        1709 bytes Aug 30 13:33
/template/7314420                     Mar 18 07:34
```

validation xml

/

- ○ Create
- ○ Upload
- ○ Edit
- ○ Delete
- ○ File Preview
- ○ Reset file to Default
- ○ Upload files to production server

Figure 1 Templates (cardinfo) section in Payeezy^{SM} test environment

3.4 Instructions for Payeezy^{SM} Certificate and Keystore Creation

3.4.1 For PHP/.Net–based websites (using OpenSSL)

To create keystore file for php or .NET/c# solution on Linux, you can use same standard OpenSSL commands as for Windows, just path to openssl will be different.

For example, on Windows:

```
"C:\Program Files\GnuWin32\bin\openssl.exe" pkcs12 -in 1234567.p12 >
1234567keystore.pem
```

On Linux:

```
openssl pkcs12 -in 1234567.p12 > 1234567keystore.pem
```

For Linux openssl.cfn is optional.

To create Payeezy^{SM} certificate for php or .NET/c# on Windows, use our openssl.cnf configuration file and standard OpenSSL commands.

3.4.1.1 Creation of TEST Payeezy^{SM} certificate

Let's use the following denominations:

- %path% - full path to the openssl.exe file on Your computer
- 1234567 – Your merchant ID
- Shop.com – the website address of Your website, integrated with Payeezy^{SM}

Then, the steps to create the test certificate are following:

1. Open command line console and navigate to the folder where test certificate will be created. Make sure that openssl.cfn file is placed there, too.

2. Execute
```
"%path%\openssl.exe" req -sha1 -newkey rsa:2048 -keyout
1234567key.pem -out 1234567req.pem -subj
"/C=LV/O=shop.com/CN=1234567" -outform PEM -config openssl.cnf
```

3. Sign certificate request (1234567req.pem) in Payeezy^{SM} test environment in section “Certificate Signing” and get files by e-mail

4. Place received files to the same folder, where you have created your key

5. Execute
```
"%path%\openssl.exe" pkcs12 -export -in 1234567.pem -out 1234567.p12
-certfile ECOMM-test.pem -inkey 1234567key.pem
```

6. Execute
```
"%path%\openssl.exe" pkcs12 -in 1234567.p12 > 1234567keystore.pem
```
For .Net/C# solution create DER certificate:
```
openssl x509 -in 1234567.pem -outform der -out 1234567.der
```

7. Place 1234567keystore.pem to your server, specify in Payeezy^{SM} configuration file full path to your keystore file and its password.

3.4.1.2 Creation of PRODUCTION Payeezy^{SM} certificate

Use the same denominations as in the chapter 3.4.1.1 Creation of TEST Payeezy^{SM} certificate (p.23).

Follow the steps:

1. Open command line console and navigate to the folder where test certificate will be created. Make sure that openssl.cfn file is placed there, too.

2. Execute
```
"%path%\openssl.exe" req -sha1 -newkey rsa:2048 -keyout
1234567key.pem -out 1234567req.pem -subj
"/C=LV/O=shop.com/CN=1234567" -outform PEM -config openssl.cnf
```

3. Send Your production certificate request (1234567req.pem) to payeezy@firstdata.lv

4. Place received files into the same folder where the key was created

5. Execute
```
"%path%\openssl.exe" pkcs12 -export -in 1234567.pem -out 1234567.p12
-certfile ECOMM.pem -inkey 1234567key.pem
```

6. Execute

```
"%path%\openssl.exe" pkcs12 -in 1234567.p12 > 1234567keystore.pem
```
For .Net/C# environment create DER file
```
openssl x509 -in 1234567.pem -outform der -out 1234567.der
```

7. Place 1234567keystore.pem to your server, specify in PayeezySM configuration file full path to your keystore file and its password.

3.4.2 For Java-based websites (using Java Keytool)

To create keystore for Java solution on Windows, instead of keytool write full path to keytool.exe

For example on Linux:
```
keytool -import -v -noprompt -alias ima -file 1234567.pem -keystore
1234567keystore.jks
```

On Windows:
```
"C:\Program Files\Java\jre7\bin\keytool.exe" -import -v -noprompt -alias
ima -file 1234567.pem -keystore 1234567keystore.jks
```

Important:
- To be able to create certificates, OpenSSL full version should be installed on your computer.
- Remember and use your private key password during all keystore file creation process.
- You should specify correct password for your keystore in Payeezy^{SM} configuration file.

3.4.2.1 Creation of TEST Payeezy^{SM} certificate
Let's use the following denominations:

- %path% - full path to the openssl.exe file on Your computer
- 1234567 – Your merchant ID
- Shop.com – the website address of Your website, integrated with Payeezy^{SM}

Follow this step by step instruction:
1. Execute
```
keytool -genkey -keystore 1234567keystore.jks -keyalg RSA -sigalg
SHA1withRSA -keysize 2048 -dname "CN=1234567, O=shop.com, C=LV" -
alias ima -storetype JKS
```

2. Execute
```
keytool -certreq -file 1234567req.csr -keystore 1234567keystore.jks
-alias ima
```

3. Sign certificate request (1234567req.pem) in Payeezy^{SM} test environment in section "Certificate Signing" and get files by e-mail

4. Place received files to the same folder where You have generated certificate request and keystore

5. Execute
```
keytool -import -v -noprompt -trustcacerts -alias root -file ECOMM-
```

```
test.pem -keystore 1234567keystore.jks
```

6. Execute
```
keytool -import -v -noprompt -alias ima -file 1234567.pem -keystore
1234567keystore.jks
```

3.4.2.2 *Creation of PRODUCTION Payeezy^{SM} certificate*

1. Execute
```
keytool -genkey -keystore 1234567keystore.jks -keyalg RSA -sigalg
SHA1withRSA -keysize 2048 -dname "CN=1234567, O=domain.com, C=LV" -
alias ima -storetype JKS
```

2. Execute
```
keytool -certreq -file 1234567req.csr -keystore 1234567keystore.jks
-alias ima
```

3. Send your certificate request to payeezy@firstdata.lv and get files by e-mail

4. Place received files to the same folder where You have generated certificate request and keystore

5. Execute
```
keytool -import -v -noprompt -trustcacerts -alias root -file
ECOMM.pem -keystore 1234567keystore.jks
```

6. Execute
```
keytool -import -v -noprompt -alias ima -file 1234567.pem -keystore
1234567keystore.jks
```

3.4.3 Certificate creation with scripts (only on Windows, PHP website)

This certificate creation method can be used on Windows OS only.

1. For easier certificate creation download from PayeezySM -> Download section ZIP archive - https://secureshop-test.firstdata.lv/download/CertGen_2.0.zip

2. Unzip the folder on your computer with Windows OS and read Description.txt file

3. Depending on certificate you want to create, choose production or test subfolder and run start.bat

4. To finish with keystore creation run finish.bat file in chosen folder

More detailed instructions about certificate/keystore creation read in instructions%.txt file in chosen folder.

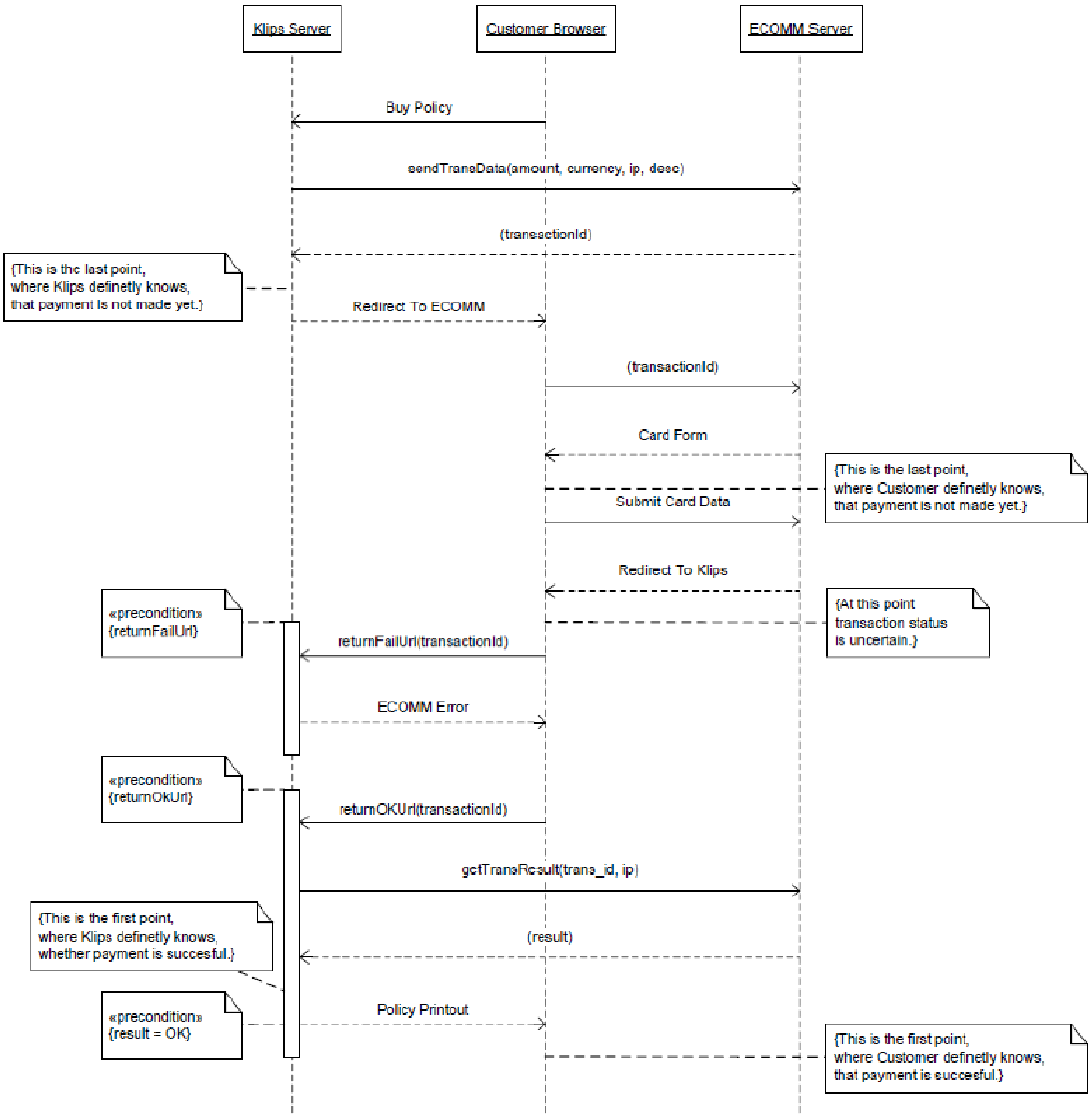
4 Result codes

Code	Action	Description
000	Approved	Approved
001	Approved with ID	Approved, honour with identification
002	Approved	Approved for partial amount
003	Approved	Approved for VIP
004	Approved	Approved, update track 3
005	Approved	Approved, account type specified by card issuer
006	Approved	Approved for partial amount, account type specified by card issuer
007	Approved	Approved, update ICC
100	Declined	Decline (general, no comments)
101	Declined	Decline, expired card
102	Declined	Decline, suspected fraud
103	Declined	Decline, card acceptor contact acquirer
104	Declined	Decline, restricted card
105	Declined	Decline, card acceptor call acquirer's security department
106	Declined	Decline, allowable PIN tries exceeded
107	Declined	Decline, refer to card issuer
108	Declined	Decline, refer to card issuer's special conditions
109	Declined	Decline, invalid merchant
110	Declined	Decline, invalid amount
111	Declined	Decline, invalid card number
112	Declined	Decline, PIN data required
113	Declined	Decline, unacceptable fee
114	Declined	Decline, no account of type requested
115	Declined	Decline, requested function not supported
116	Declined	Decline, not sufficient funds
117	Declined	Decline, incorrect PIN
118	Declined	Decline, no card record
119	Declined	Decline, transaction not permitted to cardholder
120	Declined	Decline, transaction not permitted to terminal
121	Declined	Decline, exceeds withdrawal amount limit
122	Declined	Decline, security violation
123	Declined	Decline, exceeds withdrawal frequency limit
124	Declined	Decline, violation of law
125	Declined	Decline, card not effective
126	Declined	Decline, invalid PIN block
127	Declined	Decline, PIN length error
128	Declined	Decline, PIN kay synch error
129	Declined	Decline, suspected counterfeit card
198	Declined	Decline, call Card Processing Centre
197	Declined	Decline, call AmEx
202	Pick-up	Pick-up, suspected fraud
203	Pick-up	Pick-up, card acceptor contact card acquirer
204	Pick-up	Pick-up, restricted card
205	Pick-up	Pick-up, card acceptor call acquirer's security department
206	Pick-up	Pick-up, allowable PIN tries exceeded
207	Pick-up	Pick-up, special conditions
208	Pick-up	Pick-up, lost card
209	Pick-up	Pick-up, stolen card

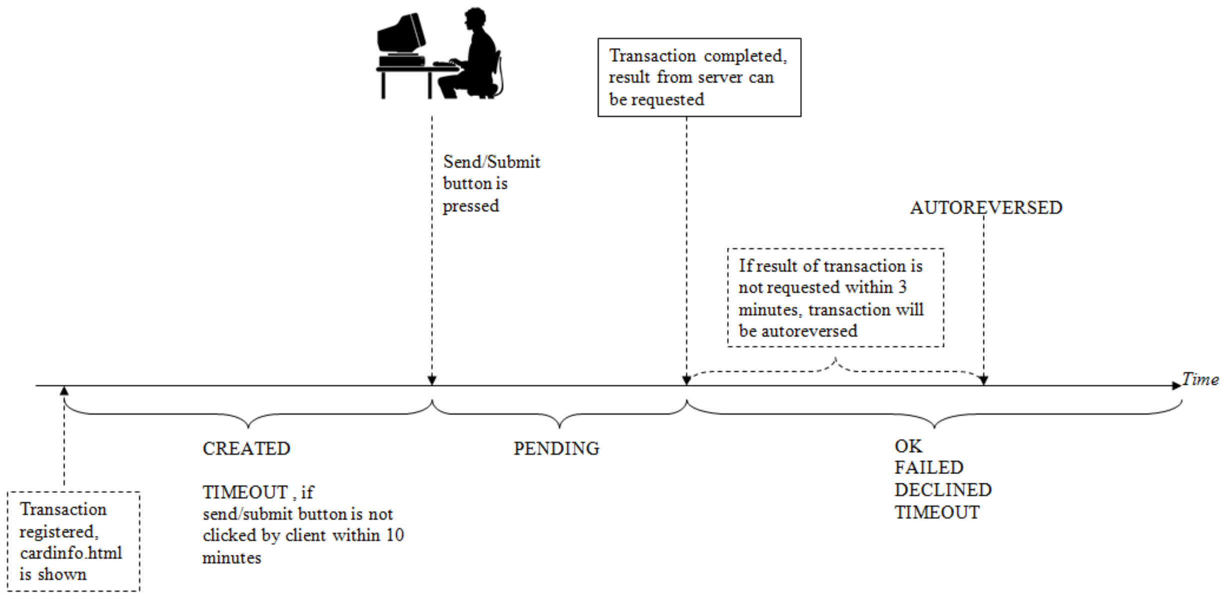
Code	Action	Description
210	Pick-up	Pick-up, suspected counterfeit card
300	Call acquirer	Status message: file action successful
301	Call acquirer	Status message: file action not supported by receiver
302	Call acquirer	Status message: unable to locate record on file
303	Call acquirer	Status message: duplicate record, old record replaced
304	Call acquirer	Status message: file record field edit error
305	Call acquirer	Status message: file locked out
306	Call acquirer	Status message: file action not successful
307	Call acquirer	Status message: file data format error
308	Call acquirer	Status message: duplicate record, new record rejected
309	Call acquirer	Status message: unknown file
400	Accepted	Accepted (for reversal)
500	Call acquirer	Status message: reconciled, in balance
501	Call acquirer	Status message: reconciled, out of balance
502	Call acquirer	Status message: amount not reconciled, totals provided
503	Call acquirer	Status message: totals for reconciliation not available
504	Call acquirer	Status message: not reconciled, totals provided
600	Accepted	Accepted (for administrative info)
601	Call acquirer	Status message: impossible to trace back original transaction
602	Call acquirer	Status message: invalid transaction reference number
603	Call acquirer	Status message: reference number/PAN incompatible
604	Call acquirer	Status message: POS photograph is not available
605	Call acquirer	Status message: requested item supplied
606	Call acquirer	Status message: request cannot be fulfilled - required documentation is not available
700	Accepted	Accepted (for fee collection)
800	Accepted	Accepted (for network management)
900	Accepted	Advice acknowledged, no financial liability accepted
901	Accepted	Advice acknowledged, finansial liability accepted
902	Call acquirer	Decline reason message: invalid transaction
903	Call acquirer	Status message: re-enter transaction
904	Call acquirer	Decline reason message: format error
905	Call acquirer	Decline reason message: acqiurer not supported by switch
906	Call acquirer	Decline reason message: cutover in process
907	Call acquirer	Decline reason message: card issuer or switch inoperative
908	Call acquirer	Decline reason message: transaction destination cannot be found for routing
909	Call acquirer	Decline reason message: system malfunction
910	Call acquirer	Decline reason message: card issuer signed off
911	Call acquirer	Decline reason message: card issuer timed out
912	Call acquirer	Decline reason message: card issuer unavailable
913	Call acquirer	Decline reason message: duplicate transmission
914	Call acquirer	Decline reason message: not able to trace back to original transaction
915	Call acquirer	Decline reason message: reconciliation cutover or checkpoint error
916	Call acquirer	Decline reason message: MAC incorrect
917	Call acquirer	Decline reason message: MAC key sync error
918	Call acquirer	Decline reason message: no communication keys available for use
919	Call acquirer	Decline reason message: encryption key sync error

Code	Action	Description
920	Call acquirer	Decline reason message: security software/hardware error - try again
921	Call acquirer	Decline reason message: security software/hardware error - no action
922	Call acquirer	Decline reason message: message number out of sequence
923	Call acquirer	Status message: request in progress
940	Not accepted	Decline, blocked by fraud filter
950	Not accepted	Decline reason message: violation of business arrangement
XXX	Undefined	Code to be replaced by card status code or stoplist insertion reason code

5 Payment schema (UML sequence diagram)


Klips Server
Customer Browser
ECOMM Server
Buy Policy
sendTransData(amount, currency, ip, desc)
(transactionId)
{This is the last point, where Klips definetly knows, that payment is not made yet.}
Redirect To ECOMM
(transactionId)
Card Form
Submit Card Data
{This is the last point, where Customer definetly knows, that payment is not made yet.}
Redirect To Klips
«precondition» {returnFailUrl}
{At this point transaction status is uncertain.}
returnFailUrl(transactionId)
ECOMM Error
«precondition» {returnOkUrl}
returnOKUrl(transactionId)
getTransResult(trans_id, ip)
{This is the first point, where Klips definetly knows, whether payment is succesful.}
(result)
«precondition» {result = OK}
Policy Printout
{This is the first point, where Customer definetly knows, that payment is succesful.}

6 Timescale of the Transaction Result


Transaction completed, result from server can be requested
Send/Submit button is pressed
AUTOREVERSED
If result of transaction is not requested within 3 minutes, transaction will be autoreversed
Time
CREATED
PENDING
OK
FAILED
DECLINED
TIMEOUT
TIMEOUT , if send/submit button is not clicked by client within 10 minutes
Transaction registered, cardinfo.html is shown