ECOMM MERCHANT INTEGRATION MANUAL

Version 06.2025

Table of contents

1 Introduction

Purpose of the document

This document describes the steps to install and run the Integrated Merchant Agent (IMA).
IMA system is designed to integrate merchants into the systems ECOMM 3D-Secure for transactions originated in the WWW environment.

Audience

The document is intended for development teams of e-commerce merchants.

2 Preparing IMA for operation

2.1 System requirements

2.1.1 The required third-party software

COMPONENT	VERSION
JAVA SE JDK	**17** or higher

2.2 Installing the system

2.2.1 Installing the IMA product

To install IMA, unpack the ‘cs.bc.ecomm_merchant-<version>.zip’ archive in the working directory. As a result, the ‘cs.bc.ecomm_merchant-<version>’ subdirectory will be created along with the directories as follows:
‘jars’ – JAR file for IMA to function (IMA resides in the 'ecomm_merchant.jar' archive),
‘doc’ – stores the present documentation.
‘merchant_cardinfo’ – stores the basic version of the card input page template

2.2.2 Additional Java class files

The following Java library files are required:
jcert.jar, jnet.jar, jsse.jar
These class files are usually included in Java standard installation. They should be placed in the same directory where ecomm_merchant.jar file is located.

2.2.3 Configuring the IMA product

1 Each merchant must receive a bank-generated file (keystore) and a password to use the file. This file is used to establish the TLS connection to ECOMM server and identify the merchant in ECOMM system.
2 ‘merchant.properties’ file must be changed (this file resides in the ‘doc’ directory):
bank.server.url – address to access the ECOMM server (this information could be located in a file which resides in the ‘doc’ directory):
keystore.file – path to the certificate file (keystore) received from the bank
keystore.type – certificate file type. We recommend that you always apply “PKCS12”.
keystore.password – password received from the bank.
3 Create and send to the bank a template in HTML format (cardinfo.html) and the supporting files to enter card information. For more details, refer to the section 5 *Generating cardinfo.html*.
(example template could be found in “merchant_cardinfo” directory:
Fibank_integration_package\cs.bc.ecomm_merchant-1.3.7\merchant_cardinfo)
4 Bank must receive information about URLs used to readdress a client back to the merchant. Remember that readdressing is performed applying the HTTP POST method. This means that URL cannot contain parameters.
returnOkUrl – the client will be readdressed to this address after 3D Secure authentication and transaction are complete (regardless of the result: OK, FAILED, DECLINED).
returnFailUrl – the client will be readdressed to this address should a technical failure occur in the ECOMM system or 3D Secure authentication process is unsuccessful.

2.2.4 ECOMM environment communication addresses

The addresses for communication with the Test and Production ECOMM systems are specified in the file “ECOMM_addresses_for_communication.txt” which resides in the “doc” directory within the integration package: Fibank_integration_package\cs.bc.ecomm_merchant-1.3.7\doc.

3 Integrating IMA into Merchant solution for online card payments

3.1 ECOMM card payments functional scheme

1. Client selects a product and is ready to pay for the purchase. On pressing the ‘Checkout’ button/link, control is transferred to the merchant solution.
2. Merchant registers a transaction in ECOMM system (indicates the amount, currency, IP address of a client, and gives a concise transaction description and language parameter), receiving transaction identifier in response.
○ Merchant has the options to define and send payment details; these options are described in the documentation of the solution.
3. Client (with transaction identifier being specified) is readdressed to ECOMM payment server so that to enter card data. Data is entered using the template provided by the merchant.
4. Once card data is entered, client authentication takes place as part of 3D Secure. The results of authentication are communicated to ECOMM system.
5. Transaction takes place if authentication is successfully completed.
6. Client is readdressed back to the merchant (with transaction identifier being indicated).
7. Merchant, having the transaction identifier, receives information on transaction results from ECOMM (whether completed or not)

3.2 Integration

ECOMM payment server can be called through IMA (resides in the ecomm_merchant.jar archive) in several ways.
1 Calling the Java arhive ecomm_merchant.jar from a command line.
2 Direct calling the lv.tietoenator.cs.ecomm.merchant.Merchant class service methods.
Configuration file name has to be assigned to a Merchant class when this class is being created. File name lets to initialize IMA, and ConfigurationException is returned if an error occurs.

Example:

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

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

It is also possible to call ECOMM server without using Integrated Merchant Agent, but making direct http calls (using TLS). Request with all the necessary parameters in this case is passed using POST method. Merchant client certificate needs to be included in request (see common standards for X509 certificates sending with POST requests) for correct authentication on Ecomm server side.

Example of calling Ecomm server directly (CURL):

```
  curl --data {POST_PARAMETERS} --cert {MERCHANT_CERTIFICATE} {ECOMM_URL}
```

where:
POST_PARAMETERS – Http post parameters for command (example is given for each in section 4 *Merchant payment requests to ECOMM*).

MERCHANT_CERTIFICATE – merchant certificate in correct format (PKCS12 format is used for IMA, but certificate can be converted to other formats if needed – e.g. PEM format).
ECOMM_URL – address to access the ECOMM server (refer to section 2.2.4 *ECOMM environment communication addresses*).

Example of certificate format conversion:

```
From PKCS12 to PEM format, using openssl:

openssl pkcs12 -in keystore.pkcs12 -out keystore.pem -passin pass:XXXXXX -passout pass:XXXXXX
```

4 Merchant payment requests to ECOMM

Merchant system communicates with ECOMM through three types of requests: request for registration of transaction for payment; request to ECOMM's client handler for customer redirection to the payment page of ECOMM system; request for transaction result after customer is readdressed back to Merchant's OK/Fail url.

4.1 Request for transaction registration for payment (SMS):

Command line parameters:

-v	identifies a request for SMS transaction registration
amount	transaction amount in fractional units (up to 12 digits)
currency	transaction currency code (ISO 4217 / 3 digits)
client_ip_addr	client's IP address (15 characters)
description	transaction details (up to 125 characters / latin letters or digits)
language	cardinfo template language identifier (ISO language name / default value is 'Default')
3ds_requestor_challenge_ind	(OPTIONAL FOR THIS COMMAND): parameter utilized within the EMV 3D-Secure authentication of the customer, sent as additional property parameter (two digits: "04"). Used to request 3D authentication of the customer through challenge flow. Could be applied in payment re-attempt after declined transaction due to customer frictionless 3D authentication attempts lifecycle is reached.

Method call:

```
 public String startSMSTrans (String amount, String currency, String ip, String
description, String language, Properties properties)
```

HTTP POST request parameters:

```
command=v&amount=<amount>&currency=<currency>&client_ip_addr=<ip>&description=<descri
ption>&language=<language>&msg_type=SMS(&<3ds_requestor_challenge_ind>=<04>&<property
_name>=<property_value>)*
```

Result:

```
TRANSACTION_ID: <trans_id>
```

trans_id	transaction identifier (28 characters in base64 encoding)

In case of an error, the returned string of symbols begins with ‘error:‘.

Example of the result:

```
TRANSACTION_ID: bAt6JLX52DUbibbzD9gDFl5Ppr4=
```

4.2 Transaction result

Command line parameters:

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

Method call:

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

HTTP POST request parameters:

```
command=c&trans_id=<trans_id>&client_ip_addr=<ip>(&<property_name>=<property_value>)
```

Result:

```
RESULT: <transaction result>
RESULT_CODE:  <result code>
3DSECURE: <3d secure authentication result>
RRN: <rrn>
APPROVAL_CODE: <approval code>
CARD_NUMBER: <card mask>
```

RESULT parameter status:

OK – successfully completed transaction,
FAILED – transaction has failed,
CREATED – transaction just registered in the system,
PENDING – transaction is not accomplished yet,
DECLINED – transaction declined by ECOMM, because ECI is in blocked ECI list (ECOMM server side configuration),
REVERSED – transaction is reversed,
AUTOREVERSED – transaction is reversed by autoreversal,
TIMEOUT – transaction was timed out

RESULT_CODE parameter – transaction result code returned from Payment Server (3 digits). Description of all response codes from Payment Server can be found in the "ResponseCodes.txt" file: Fibank_integration_package\cs.bc.ecomm_merchant-1.3.7\doc

3DSECURE parameter status:
AUTHENTICATED – successful 3D Secure authorization
DECLINED – failed 3D Secure authorization
NOTPARTICIPATED – cardholder is not a member of 3D Secure scheme
NO_RANGE – card is not in 3D secure card range defined by issuer
ATTEMPTED – cardholder 3D secure authorization using attempts ACS server
UNAVAILABLE – cardholder 3D secure authorization is unavailable
ERROR – error message received from ACS server
SYSERROR – 3D secure authorization ended with system error
UNKNOWNSCHEME – 3D secure authorization was attempted by wrong card scheme (Dinners club, American Express)
FAILED – failed to check 3D Secure status
RRN parameter – retrieval reference number returned from Payment Server
APPROVAL_CODE parameter – approval code returned from Payment Server (max 6 characters)
CARD_NUMBER parameter – masked card number

The RESULT_CODE and 3DSECURE fields are informative only. The fields RRN and APPROVAL_CODE appear for successful transactions only, for informative purposes, and they facilitate tracking the transactions in the Payment Server. The decision as to whether a transaction was successful or failed must be based on the value of RESULT field only.
In case of an error, the returned string of symbols begins with 'error:'.
Merchant system must preserve all parameters of the response from ECOMM with transaction result for every card payment.

Example of the result:

```
RESULT: OK
RESULT_CODE: 000
3DSECURE: AUTHENTICATED
RRN: 611111407831
APPROVAL_CODE: B30361
CARD_NUMBER: 4***********6789
```

4.3 Applying the Properties parameter

Through the Properties parameter more details can be provided for Payment Server. Parameter application is conditional on a solution and described in the documentation thereof. One of the applications is flight ticket itinerary details.
Properties parameters can be entered in the command line calls as follows:

```
<Ecomm command line call>  --<property_name1>=<property_value1>--<property_name2>=<property_value2> …
```

4.4 Readdressing the client

The client can be readdressed (to enter card data) to the bank-specified URL applying the GET or POST method. It is important that the trans_id variable is transferred during readdressing. This variable contains the identifier of a transaction which has to be paid up. (Note that trans_id can include the characters ‘+’, ‘=’ and ‘/’ which must be replaced with web-friendly series (for example, ‘=’ with ‘%3D’) before it is sent. In Java environment this can be done applying the URLEncoder.encode method).
Additional parameters can be transferred during the readdressing, which will be returned back to the merchant as the client is being readdressed to the merchant page.

```
<!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 ECOMM</title>
<script type="text/javascript" language="javascript">
function redirect() {
  document.returnform.submit();
}
</script>
</head>
<body onLoad="javascript:redirect()">
<form name="returnform" action="ECOMM-address-for-communication" 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>
```

The string of symbols %%trans_id%% in the example has to be replaced with transaction identifier.

4.5 Transaction refund

Command line parameters:

-k	identifies a request for transaction refund
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.

Method call:

```
 public String
 refund(String trans_id)

 public String
 refund(String trans_id, Properties properties)

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

HTTP POST request parameters:

```
command=k&trans_id=<trans_id>&amount=<amount>(&<property_name>=<property_value>)
```

Result:

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

Refund results:

OK - successful refund transaction
FAILED - failed refund transaction

result_code – result code returned from Card Suite Processing RTPS (3 digits)

refund_transaction_id refund transaction identifier – applicable for obtaining refund payment details or to request refund payment reversal.

In case of an error, the returned string of symbols begins with ‘error:‘.

The string of symbols %%trans_id%% in the example has to be replaced with transaction identifier.

4.6 Transaction reversal

-r	identifies a request for transaction reversal
trans_id	transaction identifier, mandatory (28 characters)
amount	optional parameter – amount for reversal in fractional units (up to 12 characters). If not specified, full original transaction amount will be reversed.

Method call:

```
public String
reverse(String trans_id)

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

HTTP POST request parameters:

```
command=r&trans_id=<trans_id>&amount=<amount>(&<property_name>=<property_value>)
```

Result:

```
RESULT:  <result>
RESULT_CODE: <result code>
```

Reversal results:

OK – successful reversal transaction
REVERSED – transaction has already been reversed
FAILED – failed to reverse transaction (transaction status remains as it was)
result_code – reversal result code returned from payment server.
In case of an error, the returned string of symbols begins with ‘error:‘.

Example of the result:

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

5 Generating cardinfo.html

Card data is entered into a dynamically generated HTML template whose design can be adapted to the needs of the merchant. A sample template can be found in Fibank_integration_package\cs.bc.ecomm_merchant-1.3.7\merchant_cardinfo.