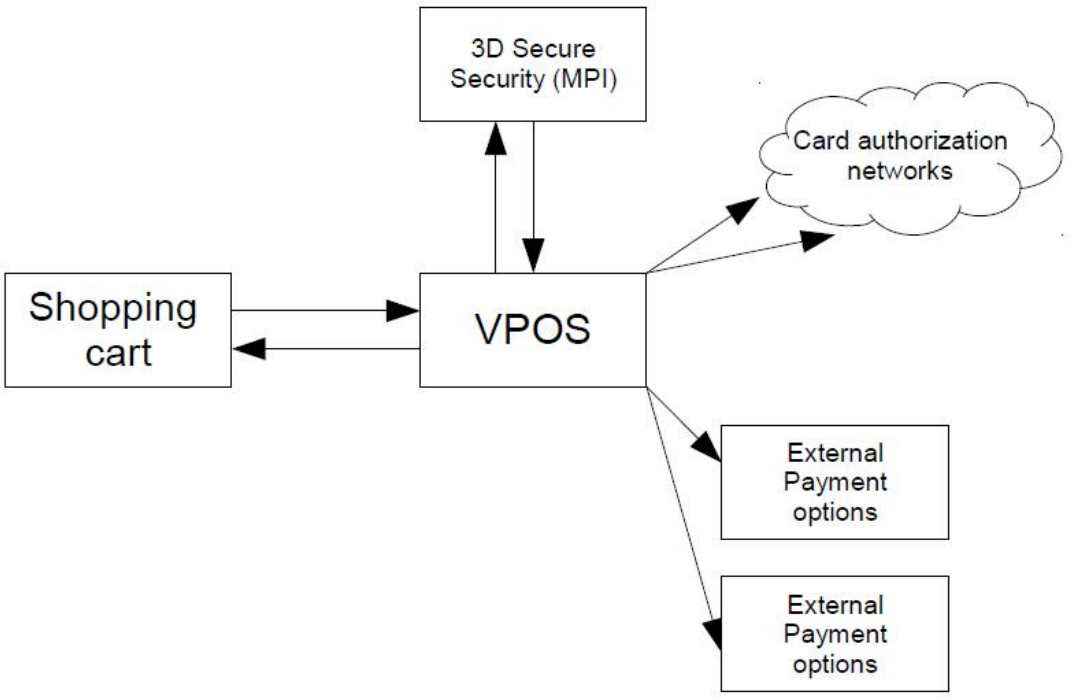
ALPHA BANK

Alpha e-Commerce

Merchant Technical Integration manual for Alpha e-Commerce Redirect model service



Alpha Bank
June 2019

Versions

Version	Date	Description / Changes
1.0	03.08.2013	Original version
1.1	16.10.2013	Changes in Merchant Id (variable in payment page)
1.2	23.12.2013	Changes in transaction amount (variable in payment page)
1.3	18.3.2014	Changes in language (variable in payment page)
1.4	20.05.2014	Changes in order description (variable in payment page)
1.5	13.8.2014	New variables in transaction type (variable in payment page)
1.6	01.10.2014	Changes in transaction type (variable in payment page)
1.7	19.10.2015	Searcing transactions with xml queries
1.8	04.04.2016	Payments with Masterpass Remove a separate view VISA Electron cards
1.8.1	17.02.2017	Recurring Payments
1.8.2	1.11.2017	Add payment method in checkout page.
1.8.3	15.6.2019	Changes to required fields in Checkout page
1.8.4	30.6.2021	Remove payments with Masterpass

The information presented in this manual is subject to changes without prior notice

CONDENT

ANNEXES

1. Procedure for merchant activation in production

The necessary steps for a merchant to be activated in production mode are the following:

1. Complete the technical integration of the merchant's website with the Alpha e-Commerce system following the instructions of this manual.

   The necessary test details are:

   a. FORM Action POST URL: https://alphaecommerce-test.cardlink.gr/vpos/shophandlermpi
   mid: (provided to merchant already)
   Shared secret key: (provided to merchant already)

   b. Test BackOffice tool access:
   BackOffice URL: https://alphaecommerce-test.cardlink.gr/mpivposmngr
   Username: (provided to merchant already)
   Password: (provided to merchant already)

   c. Test card numbers :

Type	Card Number	Exp. Date	CVV2	Card holder name	3D password	Authorization message
Visa	4012000000012003001	12/23	123	test	Secret33!	Success
Visa	4012000000012011046	12/23	123	test	Not required	Fail
Visa	4012000000012011000	12/23	123	test	Not required	Success
Visa	4012000000012011004	12/23	123	test	Not required	Success
Visa	4012000000012011012	12/23	123	test	Not required	Fail
Visa	4012000000012011020	12/23	123	test	Not required	Fail
Visa	4012000000012011038	12/23	123	test	Not required	Fail
American Express	37075510000002	12/23	123	test	Not required	Success
American Express	37075510000010	12/23	123	test	Not required	Fail
MasterCard	5544330000000011	12/23	123	test	Not required	Success
MasterCard	5900070000000029	12/23	123	test	Not required	Fail

To avoid any possible problems when using the cards, please make transactions with small amounts (under 0,20€)

2. Make at least one successful test transaction
3. Merchant must inform Alpha Bank on the successful completion of the technical integration by sending an email to ecommercesupport@alpha.gr while declaring the desired date for going live in the production mode
4. Alpha Bank will check if the merchant's site complies with the following requirements:

   1. SERVICES OR PRODUCT DESCRIPTION
   2. MERCHANT'S CONTACT DETAILS – PHONE NUMBER
   3. MERCHANT'S CONTACT DETAILS - E-MAIL
   4. MERCHANT'S CONTACT DETAILS – PHYSICAL ADDRESS
   5. CANCELLATION POLICY
   6. DELIVERY POLICY
   7. TERMS OF USE
   8. CUSTOMER ACCEPTS TERMS OF USE BEFORE PAYMENT
   9. SECURITY OF TRANSACTIONS
   10. ACCEPTED PAYMENT METHODS
   11. ACCEPTED CARDS LOGO DISPLAY*

*All major card payment schemes require their logo to be displayed in a merchant's website. Merchant has already received a "Technical Integration.zip" file where you can find all the logos required to be displayed in the website

12. RECEIPT FOR A SUCCESSFUL TRANSACTION (CUSTOMER SHOULD BE ABLE TO SAVE OR PRINT THIS RECEIPT)
13. AT LEAST ONE SUCCESSFUL TRANSACTION IN THE TEST ENVIRONMENT

2. Merchant Technical Integration

2.1 Technical Integration models

Alpha Bank offers its merchants the ability to accept credit, debit or prepaid cards payments in their website, 24 hours a day / 365 days a year. In order to accept those cards, merchants have to technically connect their website to the Alpha e-Commerce service.

The Bank offers the following models of connection:

1. Redirect Model

When the customer selects a card as a payment option, he is automatically redirected (through HTTP Post) to the Bank's payment page where he enters his card data (card pan, expiry date etc.).

2. Direct Model (XML Web Service)

The customer remains in the merchant website and enters his card dara there. Merchant is obliged to support 256 bit SSL encryption and follow all Visa, Mastercard and American Express PCI Security Standards.

In this manual, the technical integration of the Redirect Model is described

2.2 What does the merchant have to implement?

Merchant has to implement the following:

Obligatory

Merchant has to implement the following web pages on his website:

a. Checkout page
b. OK page
c. NOTOK page

The OK and NOTOK pages are sent as variables from the checkout page to the Alpha e-Commerce service in every transaction. For more info, please read Chapter 2.

Optionally:

Merchant can partially customize the payment page so as to have the look and feel of his website. In this case, merchant has two options:

a. Customize the default css file of the Bank's payment page (the css file can be sent to the merchant upon request). The new customized css file can either be sent to the Bank (at ecommercesupport@alpha.gr) to be stored at the Bank server or the merchant can store it to its server and call it in every transaction through the cssUrl variable
b. Customize the default xslt file of the Bank's payment page (the xslt file can be sent to the merchant upon request). The new customized xslt file must be sent to the bank so as to be stored in its server

Please note that in the case of a customized payment page the Alpha Bank logo and the Alpha e-Commerce logo must be displayed. These logos are located in the "Logos" folder of the attached "Technical Integration.zip" file.
In case the merchant does not want to change the payment page, then the Bank's default payment page will be used (the default payment page can be seen in section C of the Appendix).

2.3 E-Commerce transaction flow

The typical e-Commerce transaction flow consists of the following steps:

1. Customer selects the products or services he wants to buy, accepts the Terms of Use of the website and finally selects the payment method in the checkout page
2. In checkout page customer selects a payment method «**Payment with Card**»

To secure the transactions, Alpha e-Commerce service uses the "digest" method (see Chapter 5) which ensures that the transaction parameters are valid.

3. If the digest check is successful, Alpha e-Commerce service displays the payment page in the customer's browser and asks for the card details. **(If the digest check fails, the transaction is cancelled).**

For Alpha Bank cards:

- Card's logo is shown.
- Card's Loyalty Programme and the points available in their card

If the customer's card is registered in the 3D Secure service (valid for Visa / MasterCard / American Express/Diners cards), the customer's browser redirects him to the issuer's 3D Secure page where he must enter his 3D Secure password. If the authentication is successful, then the transaction is sent to the Bank for authorization. If the authentication fails, then the transaction is cancelled

4. The Bank processes the transaction:

   a. If the transaction is authorized, Alpha e-Commerce Service sends the payment response to the merchant's OK page. Customer is also redirected there. Finally Alpha e-Commerce service sends a confirmation mail to the merchant about the transaction and, optionally, to the customer (mail templates can be found in sections A1 and A2 of the Appendix)

   b. I f the transaction is not authorized, Alpha e-Commerce Service system sends the payment response to the merchant's NOTOK page. Customer is also redirected there.

Let us inform you that Alpha e-Commerce payment page has timeout at 30 minutes.

2.4 Recurring Payments.

Alpha e-Commerce also supports recurring payments of a fixed amount.
Recurring payments are used in order to provide cardholders with the capability to authorise your business online, via your website, to charge the card that they declare, at regular intervals specified when the transaction is made.

In that case, when cardholder makes an order in your website, selects recurring payments (between the available choices shown there), by accepting the relevant terms, then:

1) Payment occurs as an e-commerce payment (so a 3D-Secure password is required)
2) Alpha e-Commerce will schedule the recurring payments with the frequency chosen in the website.

To activate recurring payments in Alpha e-Commerce profile:

1) you have to send the variables **extRecurringfrequency** and **extRecurringenddate**
2) company must request the recurring payments' activation by sending an e-mail to ecommerce@alpha.gr.

If a recurring payment is declined, two more trials will be executed during the day. If all three trials are declined company have to contact with the client.

In section 3 and 4.1 you will find all the variables you must send and receive from Alpha e-Commerce platform.

3. Checkout Page

Communication between the merchant's website and Alpha e-Commerce service is executed through http post messages method.

Merchant must send to the Alpha e-Commerce service a valid html code containing all the transaction details to notify the system that there is a transaction in progress.

The variables that the merchant must send (some variables are Required and some are Optional) are the following:

Checkout page fields				
A/A	**Field (HTTP POST parameter)**	**Required / Optional**	**Field length**	**Description**
1	version	R		Value=2
2	mid	R	N10[1]	Unique Merchant Code (as provided by the Bank)
3	lang	O	A2	Language code according to the ISO 639-1 format *Greek: el* *English: en*
4	deviceCategory	O	N1	Device category 0 for www browser 1 for mobile browser *If nothing is sent, then the default value is 0*
5	orderid	R	AN…50	Unique transaction ID defined by the merchant. *Spaces are not allowed.* *The value of this variable must be unique for every transaction*
6	orderDesc	O	AN…128	Order description text
7	orderAmount	R	N…15	Transaction amount *Decimal number with 2 decimal digits divided by either comma or full stop (e.g. 0,10 10,00 10.45)*
8	currency	R	A3	Currency code according to the ISO 4217 alphabetic code. *For Euro use EUR*

[1] AN: Alphanumeric characters, A: Alphabetic characters, N: Numeric characters

9	payerEmail	O	AN…64	Customer's e-mail address
10	payerPhone	O	N…30	Customer's phone number
11	billCountry	R	A2	Billing address country code according to the ISO-3166-1-alpha-2 format (e.g. GR, US, IT)
12	billState	O	AN…50	Billing address state (str 2 3166-2 country subdivision code). this value only applies to countries that have states (e.g USA). For Greece, strongly recommended to be omitted
13	billZip	R	A…16	Billing Address Zip code
14	billCity	R	AN…64	Billing Address City
15	billAddress	R	AN…100	Billing Address Street
16	weight	O	N…12	Product's weight in kilos *Decimal number with 2 decimal digits divided by either comma or full stop (e.g. 0,10 10,00 10.45)*
17	dimensions	O	N…25	Product's dimensions in mm *Integer numbers in the Width:Height:Depth format (e.g. 200:200:200)*
18	shipCountry	O	A2	Shipping address country code according to the ISO-3166-1-alpha-2 format (e.g. GR, US, IT)
19	shipState	O	AN…50	Shipping Address State
20	shipZip	O	A…16	Shipping Address Zip code
21	shipCity	O	AN…64	Shipping Address City
22	shipAddress	O	AN…100	Shipping Address Street
23	addFraudScore	O	N…12	Fraud risk index. *Sent in case the Fraud Scoring Server is activated (future option)*
24	maxPayRetries	O	N…2	*Future Option*
25	reject3dsu	O	A…1	*Future Option*
26	payMethod	R	A…12	*You do not need to use this parameter*

27	trType	O	N1	Transaction type 1=sale[2] 2=authorization[3] *If nothing is sent, the default value is 1 (sale)*
28	extInstallmentoffset	O	N…2	Number of the offset period in months. First installment will be charged after that period *Applicable only if there is an agreement with the Bank*
29	extInstallmentperiod	O/R	N...2	Number of monthly installments (applicable only if there is an agreement with the Bank) **This variable is required if variable "extInstallmentoffset" is sent.** **Installments parameters and recurring parameters cannot be sent simultaneously**
30	extRecurringfrequency	O	N..3	The frequency of recurring payment in days (the value for monthly recurring payments is 28) *Applicable only if there is an agreement with the Bank*
31	extRecurringenddate	O/R	N8	The end date of the recurring payments **This variable is required if variable "extRecurringfrequency " is sent.** **Date format is YYYYMMDD and must must not exceed five years (1825 days)** **Recurring parameters and installments parameters cannot be sent simultaneously**
32	blockScore	O	N…9	The fraud score that blocks a transaction *Sent in case the Fraud Scoring Server is activated (future option)*
33	cssUrl	O	AN…128	The URL address of a custom css stylesheet to be used to display a customized payment page

[2] The customer's card is charged directly

[3] The amount is withheld in the customer's card and the charge of the card takes place in later time

				Note: If payment page is SSL secured make sure that the URL is also SSL secured or else browsers may show unsecure element object warning
34	confirmUrl	R	AN…128	The confirmation URL where payment confirmation is sent in case of a successful payment (OK page)
35	cancelUrl	R	AN…128	The cancel URL where payment failure is sent in case of a failed payment (NotOK page)
36	var1	O	AN…255	In var1 field, merchant can send the number of the customer's tax card which is issued by the Ministry of Finance (future option)
37	var2	O	AN…255	Free variable (in variables var2 – var5 merchant can send whatever data he wants) **Please note that the var1-var5 variables values are not returned in the Bank's response**
38	var3	O	AN…255	Free variable (in variables var2 – var5 merchant can send whatever data he wants) **Please note that the var1-var5 variables values are not returned in the Bank's response**
39	var4	O	AN…255	Free variable (in variables var2 – var5 merchant can send whatever data he wants) **Please note that the var1-var5 variables values are not returned in the Bank's response**
40	var5	O	AN…255	Free variable (in variables var2 – var5 merchant can send whatever data he wants) **Please note that the var1-var5 variables values are not returned in the Bank's response**
41	digest	R		The calculation method is described in Chapter 5

NOTES:

- The above mentioned variables are sent to the FORM Action Post URL which is https://alphaecommerce-test.cardlink.gr/vpos/shophandlermpi (valid only for the test environment)
- All parameters in the post must be in form default encoding and form must be submitted with utf-8 encoding

form.action="{supplied vpos service url}"
form.method="POST"
form.enctype="application/x-www-form-urlencoded"
form.accept-charset="UTF-8"

AUTHORIZATION TRANSACTION EXAMPLE WITH VISA CARD (PAYMENT AMOUNT=5€)

```
<!--BEGINNING OF PAYMENT FORM-->
<form id="shopform1" name="demo" method="POST" action="https://alphaecommerce-test.cardlink.gr/vpos/shophandlermpi" accept-charset="UTF-8" >
<input type="hidden" name="version" value="2"/>
<input type="hidden" name="mid" value="1011010"/>
<input type="hidden" name="lang" value="en"/>
<input type="hidden" name="deviceCategory value="0"/>
<input type="hidden" name="orderid" value="1234567"/>
<input type="hidden" name="orderDesc" value="DVD and Book"/>
<input type="hidden" name="orderAmount" value="5,00"/>
<input type="hidden" name="currency" value="EUR"/>
<input type="hidden" name="payerEmail" value="john@test.gr"/>
<input type="hidden" name="billCountry" value="GR"/>
<input type="hidden" name="billState" value="Attiki"/>
<input type="hidden" name="billZip" value="10559"/>
<input type="hidden" name="billCity" value="Athens"/>
<input type="hidden" name="billAddress" value="11 Sofokleous str"/>
<input type="hidden" name="trType" value=" 2"/>
<!-- Optional Fields-->
<input type="hidden" name="extInstallmentoffset"1"/>
<input type="hidden" name="extInstallmentperiod"12"/>
<input type="hidden" name="extRecurringfrequency"12"/>
<input type="hidden" name="extRecurringenddate"20120512"/>
<!-- Optional Fields-->
<input type="hidden" name="cssUrl" value="http://www.merchatshop.gr/css/eshop.css"/>
<input type="hidden" name="confirmUrl" value="http://www.merchatshop.gr/OK.jsp"/>
<input type="hidden" name="cancelUrl" value="http://www.merchatshop.gr/NOTOK.jsp"/>
<input type="hidden" name="var1" value="test1"/>
<input type="hidden" name="digest" value="dhdhfkjhljkhkljhlkjh="/>
</form>
<<!-- END OF PAYMENT FORM-->
```

The code to be used by the merchant for sending the transaction to the Bank can be developed in the same way the merchant has developed his e-shop (e.g. in JSP, ASP)

Note: Never implement the connection using technology that may be visible to the customer (e.g. html in javascript only).

Examples of the above implementation (in ASP, JSP, PHP) can be found in the “Sample Files” folder of the “Technical Integration.zip” file that has already been sent to the merchant.

4. Confirmation URL

Alpha e-Commerce system informs the merchant for the result of a transaction by sending a confirmation notification to the e-shop. To enable the system to send this confirmation merchant must have two separate pages stored in his website:

- a page that receives the confirmation in case of a successful transaction (Success Confirmation URL) and
- a page that receives the confirmation in case of a failed transaction (Failure Confirmation URL)

As the user/ cardholder may close the browser or encounter a problem during the redirection to the Success Confirmation URL / Failure Confirmation URL, we recommend to activate the background confirmation process. In this case, immediately after the initial response message, you will receive the response message again in confirmUrl or cancelUrl, depending on the transaction result. For the above, you must enable the re-call feature of Url. In order to enable background confirmation, please let us know to activate this feature in your test code.

4.1 Success Confirmation URL

The Success Confirmation URL is stored in the merchant's website server and receives notifications for successful transactions from Alpha e-Commerce system. The Success Confirmation URL must be sent from the merchant in every transaction via the "confirmUrl" variable.

After a successful transaction, Alpha e-Commerce system returns the following variables to the Success Confirmation URL:

Success Confirmation URL fields		
A/A	**Field (HTTP POST parameter)**	**Description**
1	version	Value=2
2	mid	The value of the original post request is returned
3	orderid	The value of the original post request is returned
4	status	Result of the transaction: **AUTHORIZED** if a PRE-AUTH transaction was requested **CAPTURED** if a sale transaction was requested
5	orderAmount	The value of the original post request is returned
6	currency	The value of the original post request is returned
7	paymentTotal	The value of the "orderAmount" variable in the original post request is returned
8	message	Additional info for the transaction (all possible messages can be found in section E of the Apendix)
9	riskScore	Fraud risk index *Sent in case the Fraud Scoring Server is activated (future option)*
10	payMethod	Payment method • visa • mastercard • maestro • amex • diners

11	txId	Transaction id generated by the system (unique for every transaction)
12	paymentRef	Transaction's Approval Code
13	digest	Value calculated using all of the above fields (for more details see chapter 5.1)

When a transaction is completed successfully, merchant must display to the customer a page that will inform him of the successful result (OK Page).

In this page the merchant must also display a "Transaction Receipt" (as described in section B of the Appendix) which the customer can print or save for future use

Recurring Transaction Fields

In case of a recurring transaction, the variables returned to the Success Confirmation URL are the following.

Note that to get these variables you must declare to the bank the url you want this information to be sent.

Success Confirmation URL fields		
A/A	**Field (HTTP POST parameter)**	**Description**
1	version	Value=2
2	mid	The value of the original post request is returned
3	orderid	The value of the original post request is returned
4	status	Result of the transaction: **AUTHORIZED** if a PRE-AUTH transaction was requested **CAPTURED** if a sale transaction was requested
5	orderAmount	The value of the original post request is returned
6	currency	The value of the original post request is returned
7	paymentTotal	The value of the "orderAmount" variable in the original post request is returned
8	message	Additional info for the transaction (all possible messages can be found in section E of the Apendix)
9	riskScore	Fraud risk index *Sent in case the Fraud Scoring Server is activated (future option)*
10	payMethod	Payment method • visa • mastercard • maestro • amex • diners
11	txId	Transaction id of the parent transaction
12	sequence	Sequence number of recurring payment (parent has a value of 1, the first child transaction has a value of 2 , etc.)
13	seqTxId	Transaction id of the child transaction
14	paymentRef	Transaction's Approval Code
15	digest	Value calculated using all of the above fields (for more details see

		chapter 5.1)

4.2 Failure confirmation URL

The Failure Confirmation URL is stored in the merchant's website server and receives notifications for failed transactions from Alpha e-Commerce system. The Failure Confirmation URL must be sent from the merchant in every transaction via the "cancelUrl" variable.

In case of a failed transaction, merchant must display to the customer a page that will inform him of the failed result (NotOK Page). Please note that for security reasons this page must not contain any sensitive information (such as the reason of failure).

After a failed transaction, Alpha e-Commerce system returns the following variables to the Failure Confirmation URL:

Failure Confirmation URL fields		
A/A	**Field (HTTP POST parameter)**	**Περιγραφή**
1	version	Value=2
2	mid	The value of the original post request is returned
3	orderid	The value of the original post request is returned
4	status	Result of the transaction (see next table)
5	orderAmount	The value of the original post request is returned
6	currency	The value of the original post request is returned
7	paymentTotal	The value of the "orderAmount" variable in the original post request is returned
8	message	Additional info for the transaction (all possible messages can be found in section E of the Apendix)
9	riskScore	Fraud risk index *Sent in case the Fraud Scoring Server is activated (future option)*
10	payMethod	Payment method • visa • mastercard • maestro • amex • diners
11	txId	Transaction id generated by the system (unique for every transaction)
12	paymentRef	Transaction's Approval Code
13	digest	Value calculated using all of the above fields (for more details see chapter 5.1)

In "status" filed the following values may be returned by Alpha e-Commerce system:

Message	**Περιγραφή**

CANCELED	Customer cancelled the transaction
REFUSED	Transaction was not approved by issuing Bank
ERROR	There was a technical error during the transaction

A typical NotOK page to display to the customer is the following:

Η ΣΥΝΑΛΛΑΓΗ ΜΕ ΤΗ ΠΙΣΤΩΤΙΚΗ ΣΑΣ ΚΑΡΤΑ ΔΕΝ ΕΙΝΑΙ ΕΦΙΚΤΗ
Η ΣΥΝΑΛΛΑΓΗ ΔΕΝ ΕΓΚΡΙΝΕΤΑΙ
Η ΣΥΝΑΛΛΑΓΗ ΔΕΝ ΠΡΑΓΜΑΤΟΠΟΙΗΘΗΚΕ

Συνήθεις Λόγοι Μη πραγματοποίησης της συναλλαγής

1. ***Λανθασμένη Καταχώριση των στοιχείων της Κάρτας σας***
2. ***Υπέρβαση του Πιστωτικού Ορίου της Κάρτας σας***
3. ***Χαμηλή Ποιότητα της Σύνδεσης σας με το Internet***
4. ***Ξαφνική Διακοπή της Σύνδεσης σας με το Internet***

YOUR CREDIT CARD IS NOT APPROVED
THE DEBIT OF YOUR CREDIT CARD IS NOT POSSIBLE
THE TRANSACTION WAS NOT COMPLETED

Typical reasons for processing interruption

1. ***Wrong card details***
2. ***Exceeding of your credit limit***
3. ***Low quality of your internet connection***
4. ***Unexpected interruption of your internet connection***

Sample pages (σε ASP, JSP, PHP) you can find in «Sample Files» included in «Technical Integration.zip».

5. Digest calculation

Validity testing of the details of a transaction is an additional way of preventing a fraud transaction to be authorized by the system. If someone wants to interfere with a valid transaction, he will attempt to corrupt the transaction details and send to the system fraudulent data. Alpha e-Commerce service can hinder anyone from corrupting the incoming transactions by making a validity testing in the transaction details.

The key to a secure exchange of data between the merchant and Alpha e-Commerce service is the variable “Digest”. “Digest” variable is sent by the merchant’s checkout page along with the other transaction variables.

The Digest validity process is as follows:

1) The Bank assigns a specific password, the Shared Secret Key, for every merchant. Only the Bank and the Merchant know this password.
2) In every transaction, the merchant’s checkout page calculates and sends the “Digest” variable along with the rest variables. “Digest” variable cannot be calculated without the Shared Secret Key password.
3) Alpha e-Commerce service checks all the variables sent by the merchant’s checkout page and with the use of the shared secret key password verifies the validity of the transaction. If the transaction is considered valid, then the customer is redirected to the payment page. If not, the transaction is cancelled.

The above validation process is hidden from the customer.

The Digest calculation requires three parameters:

1) String of the Post field values sent by the checkout page

A string that includes all variables sent by the checkout page is created

2) Shared Secret Key

Shared Secret Key is a password that only the merchant and the Bank know. This password secures the data transfer from the merchant to the Bank. The Shared Secret key is provided by the Bank and must be included in the end of the string mentioned above.

3) Base64 and sha-256 formulas

a. base64: Encoding method that converts binary digits to ASCII and vice versa
b. sha-256: cryptographic hush algorithm used for digital signatures creation

ATTENTION :

Never implement the digest calculation in browser using javascript or similar as you may expose the shared secret key to the world. Implementation with server side scripting technologies (php, asp, etc.) is strongly recommended.

The digest variable is calculated by implementing the following steps:

1. Sorting of all the variables sent exactly as the sorting in the Checkout Page fields table (page 7)
2. Creation of a string including all variables sent
3. Addition of the Shared Secret Key
4. String encoding and production of the digest variable

1. Sorting of all the variables sent exactly as the sorting in the Checkout Page fields table

Merchant must send through the checkout page all the required variables and the optional too if he wants.

Example:

Corrrect

A/A	Post field	Post field value
1	version	2
2	mid	9123456
3	lang	el
4	deviceCategory	0
5	orderid	98765
6	orderDesc	Order No 98765
7	orderAmount	1,23
…	…	…
…	…	…
34	confirmUrl	http://www.alpha.gr/succes
35	cancelUrl	http://www.alpha.gr/fail

False

A/A	Post field	Post field value
1	version	2
2	mid	9123456
3	lang	el
4	deviceCategory	0
5	orderid	98765
6	orderDesc	Order No 98765
7	orderAmount	1,23
…	…	…
…	…	…
34	confirmUrl	http://www.alpha.gr/succes
35	cancelUrl	http://www.alpha.gr/fail

2. Creation of a string including all variables sent

Based on all variables sent by the checkout page, the relevant string is created. Please note that:

- The string values must not have a separator between them. All values are added one next to another
- All variables sent by the merchant must be included in the string

- The sorting of the variables must be exactly the same as the Checkout page field table (page 7)
- UTF-8 character encoding must be used for the string creation

Example 1:

Correct
String = “29123456el98765Order No 987651,23…http://www.alpha.gr/succeshttp://www.alpha.gr/fail”

All variables have been placed in the right order

False
String = “29123456 el98765Order No 987651,23…http://www.alpha.gr/succeshttp://www.alpha.gr/fail”

There is a space between variables mid and lang

Example 2:

All variables sent should be included in the string (even the ones that are empty)

A/A	Post field	Post field value
1	version	2
2	mid	9123456
3	lang	el
4	deviceCategory	0
5	orderid	98765
6	orderDesc	(space)
7	orderAmount	1,23
…	…	…
…	…	…
34	confirmUrl	http://www.alpha.gr/succes
35	cancelUrl	http://www.alpha.gr/fail

Correct
String=“29123456el98765 1,23…http://www.alpha.gr/succeshttp://www.alpha.gr/fail”

Even though the variable orderDesc is empty, it has a space and that space should be included in the string

False
String=“29123456el987651,23…http://www.alpha.gr/succeshttp://www.alpha.gr/fail”

Even though the variable orderDesc is empty, it has a space and that space is not included in the string

Example 3:

The sorting of the variables must be exactly the same as the Checkout page field table (page 7)

A/A	Post field	Post field value
1	Version	2
2	mid	9123456
3	lang	el
4	deviceCategory	0

5	orderid	98765
6	orderDesc	Order No 98765
7	orderAmount	1,23
…	…	…
…	…	…
34	confirmUrl	http://www.alpha.gr/succes
35	cancelUrl	http://www.alpha.gr/fail

Correct
String = "29123456el98765Order No 987651,23…http://www.alpha.gr/succeshttp://www.alpha.gr/fail"

The string values are sorted according to the checkout page field table (page 7)

False
String = "2912345698765Order No 987651,23el…http://www.alpha.gr/succeshttp://www.alpha.gr/fail"

The lang variable should not be after the orderAmount variable but after the mid variable.

3. Add shared secret key

The Shared Secret key is provided by the Bank and must be included in the end of the string.

Example

A/A	Post field	Post field value
1	version	2
2	mid	9123456
3	lang	el
4	deviceCategory	0
5	orderid	98765
6	orderDesc	Order No 98765
7	orderAmount	1,23
…		
…		
34	confirmUrl	http://www.alpha.gr/succes
35	cancelUrl	http://www.alpha.gr/fail

If the shared secret key is **123qwerty456**, then the string should be:

String = "29123456el098765Order No
987651,23…http://www.alpha.gr/succeshttp://www.alpha.gr/fail**123qwerty456**"

Attention: The shared secret key should not be posted to the Bank. It is only used in the string to produce the Digest variable.

4. Encode String and create the Digest variable

The final step for the digest production is the use of the base64 and sha-256 formulas. The formulas are used in the following way:

a. base64 (sha-256 (string*))

*string = "checkout variables + shared secret key

Example

A/A	Post field	Post field value
1	version	2
2	mid	9123456
3	lang	el
4	deviceCategory	0
5	orderid	98765
6	orderDesc	Order No 98765
7	orderAmount	1,23
…		
…		
34	confirmUrl	http://www.alpha.gr/succes
35	cancelUrl	http://www.alpha.gr/fail

Shared Secret Key = 123qwerty456

Correct
The Digest calculation is as follows:

Digest = base64 (sha-256 (“29123456el98765grOrder No 987651,23….http://www.alpha.gr/succeshttp://www.alpha.gr/fail123qwerty456”))

The digest production complies to all requirements

False
Digest = base64 (“29123456el98765grOrder No 987651,23….http://www.alpha.gr/succeshttp://www.alpha.gr/fail”)

Sha-256 formula as well as the Shared Secret Key are not used to produce the Digest

5.1 Checking the Digest in the Bank's response

The Digest sent in the Bank's response can be used by the merchant to check the authenticity of the response that the Bank sends to him after a transaction request.
When the Bank sends the response to the merchant, along with the other variables, it sends the digest variable too which contains all the other variables of the response.

To check the authenticity of the Bank's reponse, the merchant must calculate the "Digest" based on all the variables he receives in the Bank's response. After calculating the Digest, he compares the calculated value with the one that he received from the Bank. If the two values match, then the response is valid and considered as authentic. If not, then merchant must display to the customer the NOK page and perform manual reverse of the transaction from the BackOffice Merchant Tool.

Please note that merchant must never compare the digest he sent in the transaction request to the digest he received in the Bank's response as those two actions have different variables and consequently different digests. Merchant must calculate again the digest based on the variables he receives and then compare the digest value with the one received by the Bank.

Example

Suppose merchant receives the following Bank's Response variables:

A/A	Post field	Post field value
1	version	2
2	mid	9123456
3	orderid	98765
4	status	CAPTURED
5	orderAmount	1.23
6	currency	EUR
7	paymentTotal	1.23
8	message	Success
9	riskScore	0
10	payMethod	visa
11	txId	112233
12	paymentRef	888555
13	digest	9HsWSgls51pM/VN+UP1kZ+0H82o4Ra6lItFCndQaRhM=

If the merchant wants to check the authenticity of the Bank's Response, he must calculate the digest based on these variables and then compare it to the value of variable 12 (digest).

If Shared Secret Key = 123qwerty456, calculation is as follows:

Digest = base64 (sha-256 (utf8bytes(
"2912345698765CAPTURED1.23EUR1.230visa112233888555123qwerty456")))

Digest value is 9HsWSgls51pM/VN+UP1kZ+0H82o4Ra6lItFCndQaRhM=
which is equal to the value of variable 13 (digest) and so merchant is certain that the Bank's response is authentic.

6. Searching transactions with xml queries

To search transaction details without logging to Back Office, an xml request can be sent and Alpha e-Commerce will respond with all relevant information.
To get this feature please inform us to proceed in activation.

Example xml request:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<VPOS xmlns="http://www.modirum.com/schemas/vposxmlapi"
xmlns:ns2="http://www.w3.org/2000/09/xmldsig#">
        <Message version="2.0" messageId="M1497440211835" timeStamp="2017-06-
14T14:36:51.835+03:00">
                <StatusRequest>
                        <Authentication>
                                <Mid>0000001</Mid>
                        </Authentication>
                        <TransactionInfo>
                                <OrderId>1497439008107</OrderId>
                                <TxId>925832951</TxId>
                        </TransactionInfo>
                </StatusRequest>
        </Message>
        <Digest>Dia9jqm+I5pTwG+RtsxyWbDIq+yxl9cRbZvSpELOpOc=</Digest>
</VPOS>
```

The variables you must send are: StatusRequest element, TransactionInfo and Txid.

Example xml response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<VPOS xmlns="http://www.modirum.com/schemas/vposxmlapi"
xmlns:ns2="http://www.w3.org/2000/09/xmldsig#">
        <Message version="2.0" messageId="M1497440211835" timeStamp="2017-06-
14T14:36:51.949+03:00">
                <StatusResponse>
                        <TransactionDetails>
                                <OrderAmount>1.25</OrderAmount>
                                <Currency>EUR</Currency>
                                <PaymentTotal>1.25</PaymentTotal>
                                <Status>REFUSED</Status>
                                <TxId>925832951</TxId>
                                <PaymentRef>    </PaymentRef>
                                <RiskScore>10</RiskScore>
                                <Description>Refused, REFUSED response code 03</Description>
                                <TxType>PAYMENT</TxType>
                                <TxDate>2017-06-14T14:16:47.245+03:00</TxDate>
                                <TxStarted>2017-06-14T14:16:47.204+03:00</TxStarted>
                                <TxCompleted>2017-06-14T14:16:47.375+03:00</TxCompleted>
                                <PaymentMethod>visa</PaymentMethod>
                                <Attribute name="MERCHANT NO">0000001</Attribute>
                                <Attribute name="USER IP">10.3.5.253</Attribute>
                                <Attribute name="CHANNEL">XML API</Attribute>
                                <Attribute name="SETTLEMENT STATUS">NA</Attribute>
                                <Attribute name="BATCH NO">175</Attribute>
                                <Attribute name="ISO response code">03</Attribute>
```

```
                                <Attribute name="ORDER DESCRIPTION" />
                                <Attribute name="CARD MASK PAN">4016#####0002</Attribute>
                                <Attribute name="ECOM-FLG">7</Attribute>
                                <Attribute name="BONUS PARTICIPATION">No</Attribute>
                        </TransactionDetails>
                </StatusResponse>
        </Message>
        <Digest>6lHf0GO/PgCw+nvgY6Y1J2quKs+6TBiATzudG/VV568=</Digest>
</VPOS>
```

Since xml api digest's calculation is different than in redirection channel, below you will find calculation's instructions.

6.1 Digest Calculation

Validity testing of the details of a transaction is an additional way of preventing a fraud transaction to be authorized by the system. If someone wants to interfere with a valid transaction, he will attempt to corrupt the transaction details and send to the system fraudulent data. Alpha e-Commerce service can hinder anyone from corrupting the incoming transactions by making a validity testing in the transaction details.

The key to a secure exchange of data between the merchant and Alpha e-Commerce service is the variable "Digest". "Digest" variable is sent by the merchant's checkout page along with the other transaction variables.

The Digest validity process is as follows:

1) The Bank assigns a specific password, the Shared Secret Key, for every merchant. Only the Bank and the Merchant know this password.
2) In every transaction, the merchant's checkout page calculates and sends the "Digest" variable along with the rest variables. "Digest" variable cannot be calculated without the Shared Secret Key password.
3) Alpha e-Commerce service checks all the variables sent by the merchant's checkout page and with the use of the shared secret key password verifies the validity of the transaction. If the transaction is considered valid, then the customer is redirected to the payment page. If not, the transaction is cancelled.

The above validation process is hidden from the customer.

6.1.1. Calculation of the Digest in XML message

To create a digest in XML message you need:

1) **XML element <Message>**

- You have to change <Message> in normal form. The method you have to use is the http://www.w3.org/TR/2001/REC-xml-c14n-20010315.
- The encoding must be UTF-8.

2) SHARED SECRET KEY

SHARED SECRET KEY is the key between company and bank to secure variables transmition and to lock the data send to banks system. SHARED SECRET KEY is given by the bank and must be added at the end of the created string.

3) Functions base64 και sha-256

- base64: Encoding method that converts binary data into ASCII text and vice versa
- sha-256: One-way algorithm used to create digital signatures

The digest XML message steps are the following:

6.1.1.1 Create original XML

6.1.1.2 Select the content in element <Message>

6.1.1.3 Add the shared secret key to the end of the <Message> element and create a single String

6.1.1.4 Convert to UTF8

6.1.1.5 Encrypt the string and generate the value in the digest variable

6.1.1.6 Enter the digest value in the <digest> element of the original XML

Specifically:

1. XML creation.

All the Recuired and some Optional variables must be used. Company can choose which optional variables will send to xml query.

Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<VPOS xmlns="http://www.modirum.com/schemas/vposxmlapi" xmlns:ns2="http://www.w3.org/2000/09/xmldsig#">
   <Message version="2.0" messageId="M1563450665076" timeStamp="2019-07-18T14:51:05.076+03:00">
      <SaleRequest>
         <Authentication>
            <Mid>3692581</Mid>
         </Authentication>
         <OrderInfo>
            <OrderId>1563450631711</OrderId>
            <OrderDesc></OrderDesc>
            <OrderAmount>1.25</OrderAmount>
            <Currency>EUR</Currency>
            <PayerEmail></PayerEmail>
         </OrderInfo>
         <PaymentInfo>
            <PayMethod>visa</PayMethod>
            <CardPan>4509034736221000</CardPan>
            <CardExpDate>2206</CardExpDate>
            <CardCvv2>756</CardCvv2>
```

```
            <CardHolderName>John Smith</CardHolderName>
          </PaymentInfo>
        </SaleRequest>
    </Message>
    <Digest>cw4S5/iQ8WgAGekzsD9YKYd5ikXarha8413CfhFjQIE=
=</Digest>
</VPOS>
```

2. Select the content του περιεχομένου in the element <Message>.

The element <Message> must be selected.

NOTICE: You must add the namespace in the element

Example:

```
<Message xmlns="http://www.modirum.com/schemas/vposxmlapi"
xmlns:ns2="http://www.w3.org/2000/09/xmldsig#" messageId="M1563450665076" timeStamp="2019-
07-18T14:51:05.076+03:00" version="2.0">
        <SaleRequest>
                <Authentication>
                        <Mid>3692581</Mid>
                </Authentication>
                <OrderInfo>
                        <OrderId>1563450631711</OrderId>
                        <OrderDesc/>
                        <OrderAmount>1.25</OrderAmount>
                        <Currency>EUR</Currency>
                        <PayerEmail/><PayerEmail/>
                </OrderInfo>
                <PaymentInfo>
                        <PayMethod>visa</PayMethod>
                        <CardPan>4509034736221000</CardPan>
                        <CardExpDate>2206</CardExpDate>
                        <CardCvv2>756</CardCvv2>
                        <CardHolderName>John Smith</CardHolderName>
                </PaymentInfo>
        </SaleRequest>
</Message>
```

3. Canonicalize the element <Message>

Example:

```
<Message
xmlns="http://www.modirum.com/schemas/vposxmlapi" xmlns:ns2="http://www.w3.org/2000/09/xmldsi
g#" messageId="M1563450665076" timeStamp="2019-07-
18T14:51:05.076+03:00" version="2.0"><SaleRequest><Authentication><Mid>3692581</Mid></Auth
entication><OrderInfo><OrderId>1563450631711</OrderId><OrderDesc></OrderDesc><OrderAmou
nt>1.25</OrderAmount><Currency>EUR</Currency><PayerEmail></PayerEmail></OrderInfo><Paym
entInfo><PayMethod>visa</PayMethod><CardPan>4509034736221000</CardPan><CardExpDate>2
206</CardExpDate><CardCvv2>756</CardCvv2><CardHolderName>John
Smith</CardHolderName></PaymentInfo></SaleRequest></Message>
```

4. Add shared secret key at the end of the element <Message> and create the concatenated string

Example:

<Message xmlns="http://www.modirum.com/schemas/vposxmlapi" xmlns:ns2="http://www.w3.org/2000/09/xmldsig#" messageId="M1563450665076" timeStamp="2019-07-18T14:51:05.076+03:00" version="2.0"><SaleRequest><Authentication><Mid>3692581</Mid></Authentication><OrderInfo><OrderId>1563450631711</OrderId><OrderDesc></OrderDesc><OrderAmount>1.25</OrderAmount><Currency>EUR</Currency><PayerEmail></PayerEmail></OrderInfo><PaymentInfo><PayMethod>visa</PayMethod><CardPan>4509034736221000</CardPan><CardExpDate>2206</CardExpDate><CardCvv2>756</CardCvv2><CardHolderName>John Smith</CardHolderName></PaymentInfo></SaleRequest></Message>**SecRetDigest**

5. Encoding using UTF8

The string must be encoded using UTF – 8 char encoding. This can be achieved by using the functions provided by the language that implements the solution (eg utf8_encode for PHP or Encoding.Convert for .NET)..

6. String encryption and digest creation

Final step for digest creation is made by using base64 and sha-256 functions. The functions order is the following:

a. base64 (sha-256 (utf8bytes (string*)))
   * string = XML message + sharedsecret

Digest = Base64(SHA256((utf8bytes(canonicalize(Message))+utf8bytes(sharedSecret))

Example:

<Message xmlns="http://www.modirum.com/schemas/vposxmlapi" xmlns:ns2="http://www.w3.org/2000/09/xmldsig#" messageId="M1563450665076" timeStamp="2019-07-18T14:51:05.076+03:00" version="2.0"><SaleRequest><Authentication><Mid>3692581</Mid></Authentication><OrderInfo><OrderId>1563450631711</OrderId><OrderDesc></OrderDesc><OrderAmount>1.25</OrderAmount><Currency>EUR</Currency><PayerEmail></PayerEmail></OrderInfo><PaymentInfo><PayMethod>visa</PayMethod><CardPan>4509034736221000</CardPan><CardExpDate>2206</CardExpDate><CardCvv2>756</CardCvv2><CardHolderName>John Smith</CardHolderName></PaymentInfo></SaleRequest></Message>**SecRetDigest**

Digest= cw4S5/iQ8WgAGekzsD9YKYd5ikXarha8413CfhFjQIE=

7. Add digest in the element <Digest> and send it

Example:

<?xml version="1.0" encoding="UTF-8" standalone="yes"?><VPOS xmlns="http://www.modirum.com/schemas"><Message xmlns="http://www.modirum.com/schemas" lang="en" messageId="1370004820649" version="1.0"><SaleRequest><Authentication><Mid>0000001</Mid></Authentication><OrderInfo><OrderId>1369981694782</OrderId><OrderDesc></OrderDesc><OrderAmount>1.25</OrderAmount><Currency>EUR</Currency><PayerEmail>andri.kruus@modirum.com</PayerEmail></OrderInfo><PaymentInfo><PayMethod>visa</PayMethod><CardPan></CardPan><CardExpDate>1406</CardExpDate><CardCvv2>756</CardCvv2></PaymentInfo></SaleRequest></Message>SecRetDigest

<Digest>cw4S5/iQ8WgAGekzsD9YKYd5ikXarha8413CfhFjQIE=</Digest></VPOS>>

6.2 Check digest in XML Response

For security reasons, company must check the response in XML Response. The check is done with the Digest variable.

In response message the digest is included with all the rest variables. Company must create digest with variables in response message, to check and compare with the digest they send in the initial message.

NOTICE: the digest sent from the company to the bank will be different than the digest from the bank in response message. So you must not compare these two digests..

6.2.1. Check digest in XML Response

Company must prepare their system to get all the bellow variables.

Example

Response with XML

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<VPOS xmlns="http://www.modirum.com/schemas/vposxmlapi"
xmlns:ns2="http://www.w3.org/2000/09/xmldsig#">
        <Message version="2.0" messageId="M1563450665076" timeStamp="2019-07-
18T14:51:05.117+03:00">
                <SaleResponse>
                        <OrderId>1576665869855</OrderId>
                        <OrderAmount>1.25</OrderAmount>
                        <Currency>EUR</Currency>
                        <PaymentTotal>1.25</PaymentTotal>
                        <Status>CAPTURED</Status>
                        <TxId>928723031</TxId>
                        <PaymentRef>116083</PaymentRef>
                        <Description>OK, CAPTURED response code 00</Description>
                </SaleResponse>
        </Message>
        <Digest>G8PaO+q3iebId2tkDnsuzvI4iP5LKn1MfaCUBOVJs60=</Digest>
</VPOS
```

Suppose that Shared Secret Key is the: SecRetDigest

In order to check the variables validity from the XML Response, company must calculate again the new digest and compare it with the one sent by the bank.

With the message element from the response message, add the attribute xmlns=http://www.modirum.com/schemas and at the end add the Shared Secret Key:

Use the below functions:

Digest = base64 (sha-256 (utf8bytes (<Message xmlns="http://www.modirum.com/schemas/vposxmlapi" xmlns:ns2="http://www.w3.org/2000/09/xmldsig#" messageId="M1563450665076" timeStamp="2019-07-18T14:51:05.117+03:00"

version="2.0"><SaleResponse><OrderId>1576665869855</OrderId><OrderAmount>1.25</OrderAmount><Currency>EUR</Currency><PaymentTotal>1.25</PaymentTotal><Status>CAPTURED</Status><TxId>928723031</TxId><PaymentRef>116083</PaymentRef><Description>OK, CAPTURED response code 00</Description></SaleResponse></Message>SecRetDigest)))

The value created is same with the one sent by the bank and is the following:
G8PaO+q3iebId2tkDnsuzvI4iP5LKn1MfaCUBOVJs60=

ANNEX

A. CUSTOMER AND MERCHANT E-MAIL TEMPLATES

A.1 MERCHANT E-MAIL

When a transaction is completed successfully, merchant will receive a notification e-mail. The e-mail body text is a combination of plain text and variables (variables can be seen in red font) as shown below:

Subject: *Alpha e-Commerce Service - New Electronic Transaction Confirmation*

You have a new electronic transaction.

Merchant Details

Merchant Name:	*Merchant Name*
Merchant No.:	*Merchant ID*
Merchant URL:	*Merchant URL*

Transaction Details

Transaction Date:	*Date of transaction*
TX ID:	*Transaction ID*
Order ID:	*Order ID*
Payment Reference:	*Payment Reference*
Order Amount:	*Order Amount*
Type:	*Transaction Type*
Payment Method:	*Payment Method*
Installment Period:	*Installment Period*
Recurring Frequency:	*Frequency of recurring payments*
Recurring Sequence:	*Sequency of recurring payment*

Please note, that if this transaction's type is a PRE-AUTHORISATION, you must capture the transaction within 7 calendar days.

Order Details

Order Description:	*Order description*
Tax Id:	*Var1*
Var2:	*Var2*
Var3:	*Var3*
Var4:	*Var4*
Var5:	*Var5*

Customer Details

Cardholder Name:	*Customer Name*
Card Number:	*Masked Pan*

Payer email: *Payer e-mail*
Payer Phone: *Payer Phone*
User IP: *Ip Address*

Regards,
Alpha e-Commerce
ecommercesupport@alpha.gr

A.2 CUSTOMER E-MAIL

When a transaction is completed successfully, customer may receive a notification e-mail (optional). The e-mail body text is a combination of plain text and variables (variables can be seen in red font) as shown below:

Subject: *Electronic Transaction Confirmation for your Order (orderid) at merchant*

Dear customer,

This e-mail verifies that your online transaction at merchant has been processed by Alpha e-Commerce Service.

The details of your transaction are:

Merchant Name:	*Merchant Name*
Merchant URL:	*Merchant URL*
Transaction ID:	*Transaction ID*
Order Amount:	*Order Amount*
Transaction Date:	*Date of transaction*
Card Type:	*Payment Method*
Order ID:	*Order ID*
Approval Code:	*Payment Reference*
No. of Installments:	*Installment Period*
Recurring Frequency:	*Frequency of recurring payment*
No. of Recurring Payments:	*Sequency of recurring payment*

Please note that this confirmation e-mail only indicates that your transaction has been processed successfully by Alpha e-Commerce Service. It does not indicate that your order has been accepted. It is the responsibility of merchant to confirm that your order has been accepted and to deliver any goods or services you have purchased. Please save this confirmation e-mail for future reference.

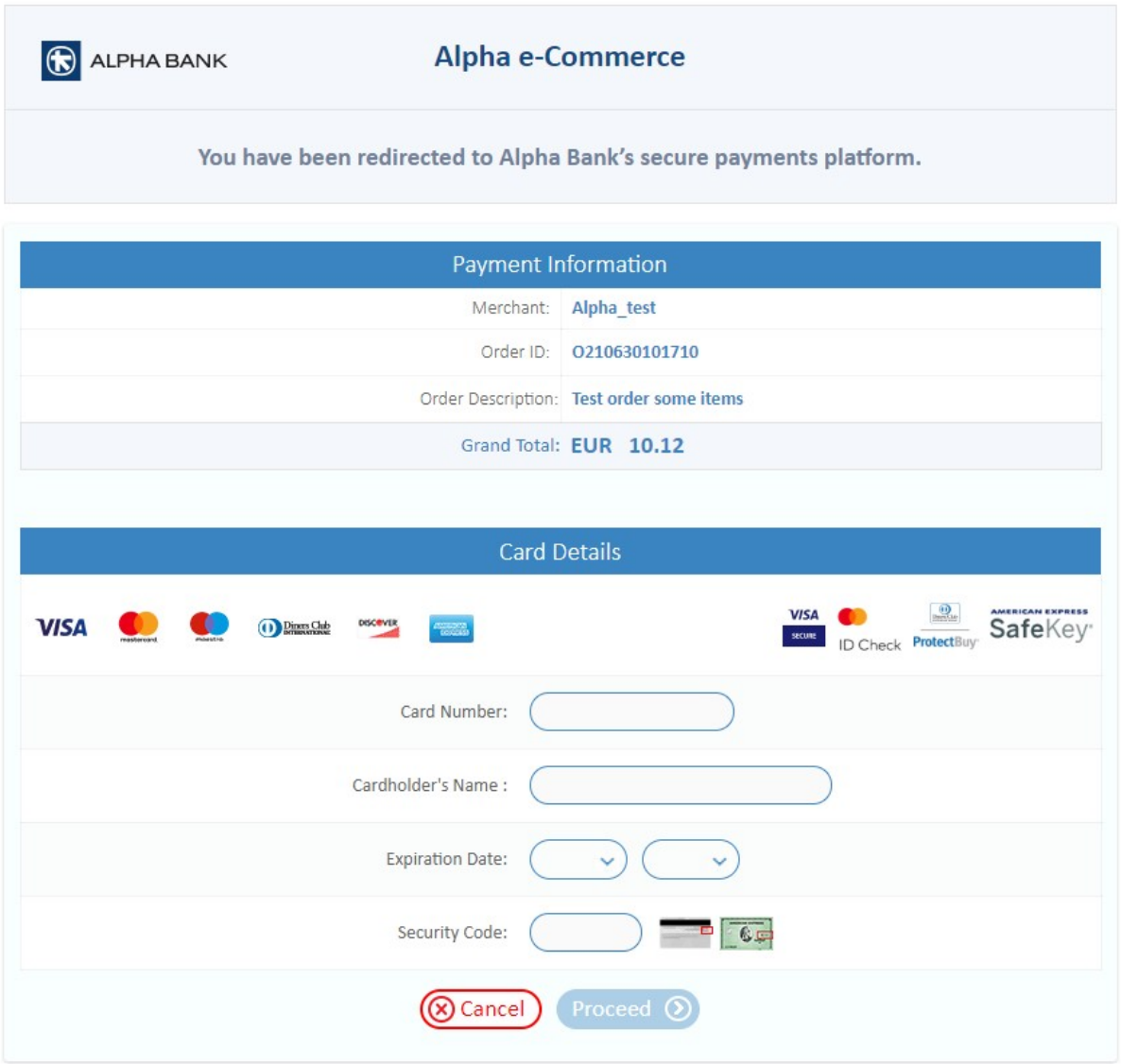
Sincerely,

Alpha e-Commerce Service
Alpha Bank

B. PURCHASE TRANSACTION RECEIPT TEMPLATE

PURCHASER NAME	
ORDER ID	
MERCHANT NAME	
MERCHANT ON-LINE ADDRESS	
DESCRIPTION OF MERCHANT'S e-SERVICES	
TRANSACTION AMOUNT	
TRANSACTION DATE	
CURRENCY	
AUTHORIZATION CODE	
E.C. (Electronic Commerce)*	
CANCELLATION POLICY	

C. BANK'S DEFAULT PAYMENT PAGE



D. ERROR MESSAGES IN PAYMENT PAGE

During test or production transactions payment page may display various error messages. The following list includes the most frequent of them:

Amount must be decimal number
Amount must be positive decimal number
Amount required
Authentication failed, please retry
Capture amount is greater than available to capture
Capture amount is greater than payment total
Capture amount is required
Capture amount must be greater than 0
Card authentication failed. You may retry.
Card holder name is required
Card number does not match card/payment type selected
Card Pan invalid
Card payment failed. You may retry or select other payment method.
Channel {1} not enabled for merchant {0}
CVV Required for this card type
Entered card data not valid
Error in {0} payment method implementation, request creation failed
Error payment method {0} is not allowed for merchant
Error payment method {0} is not yet implemented
Error Reason: {0}
Error: Transaction type {0} invalid
External payment canceled. You may retry or select other payment method.
External payment failed. You may retry or select other payment method.
Failure to process payment message
Installment offset exceed maximum allowed value {0}
Installment offset exceed merchant maximum allowed value {0}
Installment periods exeed maximum allowed value {0}
Installment periods exeed merchant maximum allowed value {0}
Installments not enabled for merchant
Installments not enabled in system
Invalid (not supported) payment method {0}
Invalid amount {0}
Invalid CVV
Invalid expiration date
Invalid input field {0} length {1} max length allowed {2}
Invalid Installment offset {1} or period {0}
Invalid merchant id "{0}"
Invalid or unsupported currency {0}
Invalid order desciption {0}
Invalid order id {0}

Invalid processing flow. Use application controls only!
Invalid recurring frequencry {0} or end date {1}
Invalid request, transaction not created
Invalid shipping weight {0}
Merchant {0} is not enabled
Merchant {0} not found in system
Merchant missing
Merchant will not allow this transaction, please select other payment method or cancel
Pan not matching original
Payment method must be selected
Please report this error with id to email address below, if it happens again.
Recurring end date {1} is not in format {0}
Recurring end date exceeds maximum {0} days allowed for merchant
Recurring end date exceeds recurring max period {0}
Recurring frequency exceeds recurring merchant max period {0}
Recurring not enabled for merchant {0}
Recurring not enabled in system
Recurring parameter frequency {0} or end date {1} is invalid
Request data is not valid
Request validation failed (at merchant)
Security error, invalid digest.
Security violation
Service provider is not active, transaction can not performed
Sorry, this card can not accpeted for installment payments, use another card
Sorry, this card can not accpeted for recurring payments, use another card
Sorry, this card can not accpted currently, please use another
System error id: {0}
System error, digest calculation error
Transaction missing or expired
Transaction operation notes may not be empty
Transaction operation reference may not be empty
Warning: Shipping service error, shipping unavailable
Your card is not recognized by system
Your session has been expired!

E. DECLINED TRANSACTIONS MESSAGES

During test or production transactions, Alpha e-Commerce Merchant BackOffice Tool may display various messages. The following list includes the most frequent of them:

Error Code	Message
00	APPROVED OR COMPLETED SUCCESSFULLY
01	REFER TO CARD ISSUER
02	REFER TO SPECIAL CONDITIONS FOR CARD ISSUER
03	INVALID MERCHANT
04	PICK-UP
05	DO NOT HONOR
06	ERROR
08	HONOR WITH IDENTIFICATION
11	APPROVED (VIP)
12	INVALID TRANSACTION
13	INVALID AMOUNT
14	INVALID CARD NUMBER (NO SUCH NUMBER)
15	NO SUCH ISSUER
30	FORMAT ERROR
31	BANK NOT SUPPORTED BY SWITCH
33	EXPIRED CARD
36	RESTRICTED CARD
38	ALLOWABLE PIN TRIES EXCEEDED
41	LOST CARD
43	STOLEN CARD, PICK UP
51	NOT SUFFICIENT FUND
54	EXPIRED CARD
55	INCORRECT PERSONAL IDENTIFICATION NUMBER
56	NO RECORD FOUND
57	TRANSACTION NOT PERMITTED TO CARDHOLDER
61	EXCEEDS WITHDRAWAL AMOUNT LIMIT
62	RESTRICTED CARD
65	EXCEEDS WITHDRAWAL FREQUENCY LIMIT
68	RESPONSE RECEIVED TOO LATE
75	ALLOWABLE NUMBER OF PIN TRIES EXCEEDED
76	APPROVED COUNTRY CLUB
77	APPROVED PENDING IDENTIFICATION (SIGN PAPER DRAFT)
78	APPROVED BLIND
79	APPROVED ADMINISTRATIVE TRANSACTION
80	APPROVED NATIONAL NEG HIT OK
81	APPROVED COMMERCIAL
82	RESERVED FOR PRIVATE USE
83	NO ACCOUNTS
84	NO PBF
85	PBF UPDATE ERROR
86	INVALID AUTHORIZATION TYPE
87	BAD TRACK DATA

88	PTLF ERROR
89	INVALID ROUTE SERVICE
94	DUPLICATE TRANSACTION
N0	UNABLE TO AUTHORIZE
N1	INVALID PAN LENGTH
N2	PREAUTHORIZATION FULL
N3	MAXIMUM ONLINE REFUND REACHED
N4	MAXIMUM OFFLINE REFUND REACHED
N5	MAXIMUM CREDIT PER REFUND REACHED
N6	MAXIMUM REFUND CREDIT REACHED
N7	CUSTOMER SELECTED NEGATIVE FILE REASON
N8	OVER FLOOR LIMIT
N9	MAXIMUM NUMBER OF REFUND CREDIT
O1	FILE PROBLEM
O2	ADVANCE LESS THAN MINIMUM
O3	DELINQUENT
O4	OVER LIMIT TABLE
O5	PIN REQUIRED
O6	MOD 10 CHECK
O7	FORCE POST
O8	BAD PBF
O9	NEG FILE PROBLEM
P0	CAF PROBLEM
P1	OVER DAILY LIMIT
P2	CAPF NOT FOUND
P3	ADVANCE LESS THAN MINIMUM
P4	NUMBER TIMES USED
P5	DELINQUENT
P6	OVER LIMIT TABLE
P7	ADVANCE LESS THAN MINIMUM
P8	ADMINISTRATIVE CARD NEEDED
P9	ENTER LESSER AMOUNT
Q0	INVALID TRANSACTION DATE
Q1	INVALID EXPIRATION DATE
Q2	INVALID TRANSACTION CODE
Q3	ADVANCE LESS THAN MINIMUM
Q4	NUMBER TIMES USED
Q5	DELINQUENT
Q6	OVER LIMIT TABLE
Q7	AMOUNT OVER MAXIMUM
Q8	ADMINISTRATIVE CARD NOT
Q9	ADMINISTRATIVE CARD NOT
R0	APPROVED ADMINISTRATIVE
R1	APPROVED ADMINISTRATIVE
R2	APPROVED ADMINISTRATIVE
R3	CHARGEBACK, CUSTOMER FILE
R4	CHARGEBACK, CUSTOMER FILE
R5	CHARGEBACK, INCORRECT
R6	CHARGEBACK, INCORRECT
R7	ADMINISTRATIVE TRANSACTIONS

R8	CARD ON NATIONAL NEGATIVE FILE
S4	PTLF FULL
S5	CHARGEBACK APPROVED
S6	CHARGEBACK APPROVED
S7	CHARGEBACK ACCEPTED
S8	ADMN FILE PROBLEM
S9	UNABLE TO VALIDATE PIN; SECURITY MODULE IS DOWN
T1	INVALID CREDIT CARD ADVANCE INCREMENT
T2	INVALID TRANSACTION DATE
T3	CARD NOT SUPPORTED
T4	AMOUNT OVER MAXIMUM
T5	CAF STATUS = 0 OR 9
T6	BAD UAF
T7	CASH BACK EXCEEDS DAILY
T8	INVALID ACCOUNT