maib ECOMMERCE (ECOMM)

INTEGRATION GUIDE

Developer's manual

Table of contents

1. API and modules for CMS.

For quick integration with **maib** ECOMMERCE, you can use our API and ready-made solutions that can be found in the public domain. At the moment there are several modules for CMS (WordPress, Opencart, Magento, Drupal, CS-CART, Bitrix…) and several more are in development.

maib Github repository (API and modules):

https://github.com/maibank

WordPress (WooCommerce):

https://wordpress.org/plugins/wc-moldovaagroindbank/

Bitrix:

http://marketplace.1c-bitrix.ru/solutions/richcode.maib/

We are constantly working on new ready-made solutions so that our customers (merchants) can quickly and easily integrate with **maib** ECOMM.

2. Common functionality.

1. The client selects a product and is ready to pay for the purchase. By clicking the ***Checkout***, control is transferred to the merchant's solution – website/app (hereinafter called **Merchant**).

2. The merchant registers (HTTP POST request) a transaction in **maib** ECOMMERCE server (hereinafter called **ECOMM**) and receives a transaction identifier in response (transaction_id). The request is sent over an encrypted channel (SSL) to the endpoint - *Merchant Handler URL.*

Merchant Handler URL - endpoint provided by **maib**, is used to get the transaction ID.

3. The Merchant with the specified transaction ID (transaction_id) redirects the Client to the payment page (*Client Handler URL*) of ECOMM to enter card data.

Client Handler URL - endpoint provided by **maib**, is used to redirect the Client to the payment page to enter card data.

4. Once the card data is entered, the following operations are performed, depending on whether the merchant supports 3D Secure:

- Merchant supports 3D Secure: After entering the card data, the client is authenticated within 3D Secure. The result of the authentication is sent to the ECOMM.

- Merchant does not support 3D Secure: In this case, a transaction is executed.

5. The transaction takes place if authentication is successfully completed.

6. Client is readdressed back to the merchant return page – *CALLBACK URL*.

CALLBACK URL – url provided by the Merchant and registered on the ECOMM side, the client will be redirected to this address after authentication and transaction completion (regardless of the result).

7. The merchant, in the presence of a transaction ID, receives information about the result of the transaction from ECOMM (transaction is finished or not).

8. In the event of a DMS transaction, an additional transaction is necessary to effect the payment by client. Such a transaction is typically carried out after product delivery to a customer;

9. If required, the merchant can request the ECOMM Server to reverse the transaction (reversal report);

10. At regular intervals (once a day), the merchant sends a request to the ECOMM Server to close the business day;

Note:

By default, a standard payment page template is installed on the ECOMM's side. The design of the payment page can be changed by Merchant (See 12.2). It is also possible to use the payment page in different languages (See 12.3).

2.1 Google Pay™

Google Pay is the fast, simple way that allows you to make card payments without entering card details for each payment. The card data is safely stored by Google. This payment method is available for all devices (mobile phones and computers), regardless of the operating system and web browser.

With this connection method, you do not need to perform additional integrations. The Google Pay button will be displayed on **maib** e-commerce payment page. **maib** e-commerce supports by deafult both authentication methods:

- PAN_ONLY - payment cards is stored on file with the user's Google Account.
- CRYPTOGRAM_3DS – payment cards is stored as Android device tokens.

!!! All merchants must adhere to the Google Pay APIs Acceptable Use Policy and accept the terms defined in the Google Pay API Terms of Service.

2.2 Apple Pay™

Apple Pay is the fast, simple way that allows you to make card payments without entering card details for each payment. The card data is safely stored by Apple. This payment method is available only for iOS and MAC devices.

With this connection method, you do not need to perform additional integrations. The Apple Pay button will be displayed on **maib** e-commerce payment page.

3. Preparing for maib ECOMM integration.

To send a request from the Merchant to ECOMM with all the necessary parameters, the HTTP **POST** method is used. The authentication process occurs due to the SSL certificate provided by the **maib**. The validity period of the certificate is limited (5 years).

The Merchant can also view and manage transactions through a personal account (**Merchant Portal**) using any browser of the current version.

The resource is available via this link: https://maib.ecommerce.md:11441/ecomm-portal/

Access to the Merchant Portal personal account is provided at the request of the Merchant by e-mail (ecom@maib.md).

When contacting maib (ecom@maib.md) about integration, you will receive an archive that contains everything you need to integrate with ECOMM.

The archive is available here:

https://drive.google.com/file/d/1x6qkGHlxtWp7z7OfqqCaxeBGBDeN4ist/view?usp=sharing

1. Manual.zip
   - Quick guide to integration and testing (en_Guide_test_ecommerce.pdf);
   - Integration guide for developers (en_Ecomm_Doc.pdf);
   - Requirements for online project according to the Rules of International Payment Systems; (en_Requirments_merchant_ecommerce.pdf);
2. SSL certificate for testing 0149583.pfx (certificat_SSL.zip);
3. Payment page template (cardinfo_maib_widget.zip);
4. International payment systems logo (logo_visa_mc.zip);

The test certificate is used to establish an SSL connection with the bank's ECOMM server and to identify the Merchant in the ECOMM system.

From a *pfx* certificate, you can extract the public key and private key using OpenSSL.

Example: (commands may differ depending on the ecommerce solution and the server)

```
openssl pkcs12 -in certificate.pfx -nocerts [-nodes] -out key.pem
openssl pkcs12 -in certificate.pfx -nokeys -out full-chain-certificate.pem

*centos note: curl+nss requires rsa + des3 for private key:
openssl rsa -des3 -in private-key.pem -out pk.pem
```

It is necessary to store the generated keys on the server for use in requests.

The Merchant must provide via e-mail (ecom@maib.md) the following information:

1. IP
2. CALLBACK URL

4. Test procedure.

Testing is a mandatory stage of integration. The testing process is designed to prevent errors in transactions, ensuring compliance with ECOMM requirements.

Data for testing:

Merchant Handler URL: https://maib.ecommerce.md:21440/ecomm/MerchantHandler
Client Handler URL: https://maib.ecommerce.md:21443/ecomm/ClientHandler
SSL-certificate: 0149583.pfx
Test certificate password: Za86DuC$

Commands to test:

1. Registration of transactions through a command **v** (See 5.1).
2. Request the status of a transaction through a command **c** (See 5.2).
3. Reversal of a transaction through a command **r** (See 7).
4. Closing the business day through a command **b** (See 8).

To run tests, use the virtual test card with the following data:

Card number: 5102180060101124
Expire: 06/28
CVV: 760

Test process example (PHP):

https://github.com/DiplNext/maib-ecommerce-php

Note:

Any trans_id generated by the ECOMM server has a lifetime of 10 min. This means that any action taken by the customer after this time will be rejected. So if everything is ok, after the payment is made, the cardholder is redirected to the Callback URL. However, there are cases when the cardholder may not return to this Callback, due to browser, internet, PC, electricity problems.

This means that you must provide the following transactions checks:

a) If the client does not return to the Callback URL after 5 min from the moment of generating trans_id then check the status of transaction, if you receive an intermediate status such PENDING, apply pt. b;

b) Check this status again in another 5 minutes (total 10 minutes), which means you will receive one of the following fine status (TIMEOUT, response code normal [000, 116,], FAILED, DECLINED).

Please note that in the database you must not have any transactions with unidentified or unverified status after the expiration of the 10 minutes of the transactions.

5. SMS transaction (Single Message System).

SMS transaction — this is a single transaction that allows you to transfer money from the Client's account to the Merchant's account. This means that the payment Authorization and its Execution are carried out within the same transaction.

SMS transaction is the easiest to implement and most often meets the requirements of Merchants, so it is the most common type of transaction in ECOMM integrations.

5.1. Registration of an SMS transaction (v).

To register an SMS transaction, use the command **v.**

!!! To complete the registered SMS transaction, it is necessary to execute the transaction status request command (**c**) (See 5.2).

Parameter	Type	Number of characters (max)	Description
v	**required**	**1**	**Identifies a request to register a SMS transaction.**
amount	required	12	Transaction amount in fractional units
currency	required	3	Transaction currency code (ISO 4217) MDL – 498 / USD – 840 / EUR – 978
client_ip_addr	required	15	Client's IP address
description	optional	125	Transaction details
language	required	32	Bank payment page language identifier

HTTP POST request:

```
command=v&amount=<amount>&currency=<currency>&client_ip_addr=<ip>&description=<desc>
&language=<language>&msg_type=SMS
```

Result:

```
TRANSACTION_ID: <trans_id>
```

trans_id - transaction ID (28 characters in base64 encoding).

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

5.2. Request for transaction result (-c).

After the client is redirected to the resulting CALLBACK URL page, the transaction status is requested by the command **-c**.

In addition to returning the result (status) of a transaction, this command is also used to complete certain types of transactions (ex. SMS transactions). If this command is not executed, these transactions will not be considered fully completed and will not have the FINISHED status.

If, due to reasons beyond the control of the Bank, the client is not redirected to the CALLBACK URL, it makes sense to request the transaction status only after 10 minutes have elapsed from the moment the transaction was registered (transaction life cycle is 10 minutes).

Parameter	Type	Number of characters (max)	Description
-c	**required**	**1**	**Identifies the transaction result request**
trans_id	required	28	Transaction identifier
client_ip_addr	required	15	Client's IP address

HTTP POST request:

```
command=c&trans_id=<trans_id>&client_ip_addr=<ip>
```

Result:

```
RESULT: <result>
RESULT_PS: <result_ps>
RESULT_CODE: <result_code>
3DSECURE: <3dsecure>
3DSECURE_REASON: <3dsecure_reason>
RRN: <rrn>
APPROVAL_CODE: <app_code>
CARD_NUMBER: <pan>
AAV: <aav>
PAYMENT_ACCOUNT_REFERENCE: <payment_account_reference>
RECC_PMNT_ID: <rcc_pmnt_id>
RECC_PMNT_EXPIRY: <rcc_pmnt_ex>
```

5.3. SMS transaction example.

1. Request for registering an SMS transaction:

```
https://maib.ecommerce.md:21440/ecomm/MerchantHandler?command=v&amount=12300&currency=498&client_ip_addr=109.0.20.30&description=Order#123&language=en&msg_type=SMS
```

Result:

```
TRANSACTION_ID: rEsfhyIk8s9ypxkcS9fj/3C8FqA=
```

2. Redirecting the Client to the address of the payment gateway (ClientHandler):

https://maib.ecommerce.md:21443/ecomm/ClientHandler?trans_id=rEsfhyIk8s9ypxkcS9fj/3C8FqA=

3. Confirmation/determination of the payment status (on the CALLBACK URL page).

```
https://maib.ecommerce.md:21440/ecomm/MerchantHandler?command=c&trans_id=rEsfhyIk8s9ypxkcS9fj/3C8FqA=&client_ip_addr=109.0.20.30
```

Result:

```
RESULT: OK
RESULT_PS: FINISHED
RESULT_CODE: 000
3DSECURE: AUTHENTICATED
RRN: 331711380059
APPROVAL_CODE: 327593
CARD_NUMBER: 5**********2372
```

6. DMS transaction (Dual Message System).

In case of a DMS transaction, debiting from the Client's card occurs in two stages:

1. Authorization – the funds on the Client's card are blocked.
2. Execution of the transaction (financial transaction).

For example, the execution of a transaction upon checking the availability of goods in the warehouse, upon shipment of the goods, after additional checks, etc. The Execution operation is the basis for debiting funds from the Client's account, while the Authorization only implies blocking funds on the Client's account.

!!! If you use DMS when integrating with ECOMM, **Apple Pay** payments will not be available. We recommend using **SMS** transactions.

6.1. Registration of a DMS authorization (-a).

Parameter	Type	Number of characters (max)	Description
-a	**required**	**1**	**Identifies a request to authorize a DMS transaction.**
client_ip_addr	required	15	Client's IP address
amount	required	12	Transaction amount in fractional units
currency	optional	3	Transaction currency code (ISO 4217) MDL – 498 / USD – 840 / EUR – 978
description	optional	125	Transaction details
language	required	32	Bank payment page language identifier

HTTP POST request:

```
command=a&amount=<amount>&currency=<currency>&client_ip_addr=<ip>&description=<desc>
&language=<language>&msg_type=DMS
```

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:"

!!! Based on the results of redirecting the Client to the specified CALLBACK URL, the transaction should be confirmed (command=**c**), which will also determine the success of the operation (See 5.2).

6.2. Completion of a DMS transaction (-t).

Parameter	Type	Number of characters (max)	Description
-t	**required**	**1**	**Identifies a request to completion a DMS transaction.**
trans_id	required	28	Transaction identifier
amount	required	12	Transaction amount in fractional units

currency	required	3	Transaction currency code (ISO 4217) MDL – 498 / USD – 840 / EUR – 978
client_ip_addr	required	15	Client's IP address
description	optional	125	Transaction details

HTTP POST request:

```
command=t&trans_id=<trans_id>&amount=<amount>&currency=<currency>&client_ip_addr=<ip>
&description=<desc>&language=<language>&msg_type=DMS
```

Result:

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

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

7. Transaction reversal (-r).

Reversal are initiated when it is necessary to refund funds to the Client (partial or full).

Parameter	Type	Number of chr. (max)	Description
-r	**required**	1	**Identifies a transaction reversal request**
trans_id	required	28	Transaction identifier
amount	required	12	For DMS authorizations only full amount can be reversed, i.e., the reversal and authorization amounts have to match. In other cases, a partial reversal is also available.
suspected_fraud	optional	3	A flag indicating that a transaction is being reversed because of suspected fraud. In such cases, the value of this parameter should be set to "yes". If this parameter is used, only full reversals are allowed.

HTTP POST request:

```
command=r&trans_id=<trans_id>&amount=
```

Result:

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

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

8. Closing of the business day (-b).

The procedure for closing a business day must be initiated once a day. Based on the results of the procedure, the Bank processes the received transactions, then reimburses the Merchant with funds by transferring to the settlement account under the Agreement.

We recommend closing the business day at **23:59:00**.

Parameter	Type	Symbols (max)	Description
-b	**required**	**1**	**Identifies a request to close a business day**

HTTP POST request:

```
command=b
```

Result:

```
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>
```

<result> – close-of-business-day results:

OK - successful close of business day;

FAILED - failed close of business day;

<result_code> – close-of-business-day code returned from Card Suite FO (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 case of an error, the returned string of symbols begins with "error:"

9. Recurring (regular) payments.

Recurring payments are payments that do not require the Client to re-enter the card details.

The Client makes a payment 1 time, agrees to the terms of regular debiting, subsequent debiting occurs without the participation of the Client (as an example, this could be a subscription to a service with a monthly payment). The regular payment is registered at the first payment, and the regular payment template is registered in the Bank's database. The template is assigned a number that the Merchant knows. Upon request for the template number, a regular payment operation is initiated.

Recurring payments can be created in two ways:

1. along with the execution of DMS authorization or SMS transaction for a certain amount (**-z** / **-d**);
2. by executing an account verification operation (**-p**).

9.1. Registration of a recurring payment with authorization and execution of the first payment (-z / -d).

Parameter	Type	Number of characters (max)	Description
-z (for SMS) **or** **-d (for DMS)**	**required**	**1**	**Request for registration of a regular (recurring) payment**
amount	required	12	Transaction amount in fractional units

currency	required	3	Transaction currency code (ISO 4217) MDL – 498 / USD – 840 / EUR – 978
client_ip_addr	required	15	Client's IP address
description	опционально	125	Transaction details
language	required	32	Bank payment page language identifier
biller_client_id	required	49	Merchant's chosen identifier of the regular payment. If not specified, TRANSACTION_ID will be used as the recurring payment ID.
perspayee_expiry	required	4	Validity limit of the regular payment in the format MMYY
perspayee_gen=1	required		Used to generate a new regular (recurring) payment template
perspayee_overwrite=1			Used to edit the recurring payment template (for example, if the payment details are changed by the Client).

HTTP POST request (SMS):

```
command=z&amount=<amount>&currency=<currency>&client_ip_addr=<ip>&desc=<desc>&language=<language>&msg_type=SMS&biller_client_id=<recc_pmnt_id>&perspayee_expiry=<expiry>&perspayee_gen=1
```

HTTP POST request (DMS):

```
command=d&amount=<amount>&currency=<currency>&client_ip_addr=<ip>&desc=<desc>&language=<language>&msg_type=DMS&biller_client_id=<recc_pmnt_id>&perspayee_expiry=<expiry>&perspayee_gen=1
```

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:".

!!! SMS transactions (**-z**) must be completed using the "**-c**" command (See 5.2) and the DMS authorization (**-d**) must be completed by executing the transaction with the "**-t**" command (See 6.2).

9.2. Registration of a regular payment with authorization, but without the first payment (-p).

Parameter	Type	Number of char. (max)	Description
-p	**required**	**1**	**Request for registration of a regular (recurring) payment, without amount**
currency	required	3	Transaction currency code (ISO 4217) MDL – 498 / USD – 840 / EUR – 978
client_ip_addr	required	15	Client's IP address
description	optional	125	Transaction details
language	required	32	Bank payment page language identifier
biller_client_id	required	49	Merchant's chosen identifier of the regular payment. If not specified, TRANSACTION_ID will be used as the recurring payment ID.
perspayee_expiry	required	4	Validity limit of the regular payment in the format MMYY
perspayee_gen=1	required		Used to generate a new regular (recurring) payment template

HTTP POST request:

```
command=p&currency=<currency>&client_ip_addr=<ip>&description=<desc>&language=<language>&msg_type=AUTH&biller_client_id=<recc_pmnt_id>&perspayee_expiry=<expiry>&perspayee_gen=1
```

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:".

!!! Based on the results of redirecting the Client to the specified CALLBACK URL, the transaction should be confirmed (command=**c**), which will also determine the success of the operation (See 5.2).

9.3. Making a payment based on the created Recurring payment (-e).

This command (**-e**) allows you to make a payment based on the template of a previously created Recurrent Payment. In this case, the payment will be completed and will have the status FINISHED, so there is no need to explicitly complete the created payment (without using the "**-c**" or "**-t**" commands).

Parameter	Тип поля	Количество символов	Description
-e	**required**	**1**	**Request to make a payment off a previously registered regular (recurring) payment.**
amount	required	12	Transaction amount in fractional units
currency	required	3	Transaction currency code (ISO 4217) MDL – 498 / USD – 840 / EUR – 978
client_ip_addr	required	15	Client's IP address
description	optional	125	Transaction details
biller_client_id	required	49	Identifier of the recurring payment.

HTTP POST request:

```
command=e&amount=<amount>&currency=<currency>&client_ip_addr=<ip>&description=<desc>&
biller_client_id=<recc_pmnt_id>
```

Result:

```
TRANSACTION_ID: <trans_id>
RESULT: <result>
RESULT_CODE:<result_code>
RRN:<rrn>
APPROVAL_CODE:<appr_code>
```

9.4. Deleting a Recurring payment (-x).

Parameter	Type	Number of characters	Description
-x	**required**	**1**	**Request to delete a previously created Recurring payment**
biller_client_id	required	49	Identifier of the created Recurring payment

HTTP POST request:

```
command=x&biller_client_id=<recc_pmnt_id>
```

Result:

```
RESULT: <result>
```

<result> – will have the value OK in case of a successfully deleted Recurring payment.

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

10. Description of the meaning of ECOMM responses to requests.

RESULT	Transaction result status
OK	the transaction is successfully completed
FAILED	the transaction has failed
CREATED	the transaction is just registered in the system
PENDING	the transaction is not completed yet
DECLINED	the transaction is declined by Ecomm, because ECI is in the blocked ECI list (Ecomm Server side configuration)
REVERSED	the transaction is reversed
AUTOREVERSED	the transaction is reversed by autoreversal
TIMEOUT	the transaction was timed out
RESULT_PS	**Transaction result, Payment Server interpretation**
ACTIVE	Registered and not yet completed payment
FINISHED	Successfully completed payment
CANCELLED	Cancelled payment
RETURNED	Returned payment
RESULT_CODE	**Transaction result code returned from Card Suite FO (3 digits).** In case of a Recurring payment, some result codes (attribute "result_code") have additional meanings: 108 – Merchant communication with cardholder has to be done; 114 – It is possible to try to repeat the transaction next time; 180 – Cardholder ended cooperation. Recurring payment has been deleted; 2xx – Recurring payment has been deleted.
3DSECURE	**3D Secure status**
RRN	**Retrieval reference number returned from Card Suite FO**
APPROVAL_CODE	**Approval Code returned from Card Suite FO (max 6 characters)**

CARD_NUMBER	Masked card number
TRANSACTION_ID	Transaction identifier (28 characters base64)
Applicable for recurring (regular) payment	
RECC_PMNT_ID	Recurring payment (if available) identification in Payment Server
RECC_PMNT_EXPIRY	Recurring payment (if available) expiry date in Payment Server in the form MMYY;
MRCH_TRANSACTION_ID	Merchant Transaction Identifier (if available) for the payment, shown if it was sent as an additional parameter with name "mrch_transaction_id" upon payment registration.

11. ECOMM system errors (RESULT_CODE=1001).

Error	Description
error: unregistered merchant. IP: 12.34.56.78	The Merchant's IP address is not registered in the ECOMM System. It is necessary to send a corresponding request to the Bank (ecom@maib.md) in order to add a new IP address to the list of allowed ones on the Bank's side.
error: unable to process transaction request	Loss of communication with ECOMM.
error: wrong transaction id	An invalid value for transaction id parameter was specified in the request to ECOMM.
error: no transaction id	The request in ECOMM does not specify the value of the transaction id parameter.
error: failed to get payment status	The symbol "+" in the value of the trans ID parameter is incorrectly processed on the Merchant's side.
error: total refunds amount already exceeds original amount	The total amount to be returned to the Client exceeds the amount of the original transaction.
error: digest failed	Two possible options: - The validity period of the card specified by the Client is earlier than today's date. - The number of characters in the "description" field exceeds the allowed.
error: transaction not found	The specified trans_id was not found. One of the possible reasons: The symbol "+" in the value of the trans ID parameter is incorrectly processed on the Merchant's side.
error = 'transaction already reversed'	Reversal has already been made before.

12. Additional information.

12.1. Using in the query additional attributes.

Additional properties can be used to provide more details for the ECOMM. Property application depends on the solution and is described in the documentation thereof. For example, one of the applications is flight ticket itinerary details.

Example:

command=v&amount=<amount>&currency=<currency>&client_ip_addr=<ip>&description=<desc>&language=<language>&msg_type=SMS&**<additional_property_name1>=<value1>&<additional_property_name2>=<value2>**

12.2. Readdressing the client.

The client can be readdressed for card data entry to the bank-specified URL by applying the 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 "/" that must be replaced with web-friendly series (for example, "=" with "%3D") before it is sent. Additional parameters can be transferred during readdressing, which will be returned back to the merchant as the client is being readdressed to the merchant page.

+	/	!	“	#	%	&	‘	*	,	:	;	<	=	>	?
%2B	%2F	21%	22%	23%	25%	26%	27%	%2a	%2c	%3a	%3b	%3c	%3d	%3e	%3f

[	]	^	`	{	|	}	space
%5b	%5d	%5e	60%	%7b	%7c	%7d	20%

!!! In the case when the trans_id parameter is converted incorrectly, the client will receive an error: ***error: failed to get payment status***.

12.3. HTML template cardinfo.html.

Card data is entered into a dynamically generated HTML template whose design can be adapted to the needs of a merchant. A sample template can be found in "example/cardinfo.html". The Ecomm Server recognizes the following tags in the template:

Tag	Description
%%javascript%%	Replaced with JavaScript for verification of input fields.
%%googlepay_js%%	Replaced with JavaScript for Google Pay. Add this tag in head.
<script src="%%googlepay_web_sdk_url%%" onLoad="googlePay.onGooglePayLoaded()"></script>	Replaced with JavaScript for Google Pay. Add this script in footer.
%%applepay_js%% %%applepay_css%%	Replaced with JavaScript and CSS for Apple Pay. Add this tags in head.
	Google Pay button
	Apple Pay button

%%formdef%%	Replaced with <form id=”cardentry” action=<url> method="post"> <input type="hidden" name="trans_id" value="<trans_id>" readonly> <input type="hidden" name="count" value="X" readonly> where "X" is a form sending a trial number
%%trans_id%%	<input type="hidden" name="trans_id" value=<trans_id>" readonly> - can be used to get transaction id if the "%%formdef%%" tag is not used
%%cardname%%	<input type="text" name="cardname" size="19" maxlength="100">
%%cardnr%%	<input type="text" name="cardnr" size="19" maxlength="19">
%%expmonth%%	<input type="text" name="validMONTH" size="2" maxlength="2">
%%expmonth_select%%	<select name=”validMONTH”><option value=”01”>01</option>...</select>
%%expyear%%	<input type="text" name="validYEAR" size="2" maxlength="2">
%%expyear_select%%	<select name=”validYEAR”><option value=”08”>08</option>...</select>
%%cvc2%%	<input type="text" name="cvc2" size="4" maxlength="4">
%%cvc2_password%%	<input type="password" name="cvc2" size="4" maxlength="4">
%%amount%%	Transaction amount
%%ccyalpha%%	Transaction currency
%%description%%	Transaction details transferred by merchant to the Ecomm
%%logo_url%%	Replaced with a merchant specific logo url.
%%save_card%%	Replaced with a checkbox, which if ticked will create a new Recurring payment with saved card data. Will appear only when a Recurring payment is registered with properties "ask_save_card_data=true"
%%merchantid%%	Merchant ID

12.4. Support for different languages (localization).

To implement a user interface in a language other than the default language, it is necessary to prepare appropriate HTML templates in that language and place them on the ECOMM Server. The agreed language identifier should be sent in the transaction registration message along with description parameters and other attributes, and is used on the server side to find the appropriate HTML template. The description parameter can be substituted by an empty string "" if not needed. If the template for the language specified in the query is not found, the default template is used. Error reporting is localized through files

locale.properties. A file with this name must be present in every template directory whose error reporting is to be localized.

By default, the following error reports apply:

```
invalid_cardnr=Invalid card number
invalid_cardnr_length=Card number length should be in range 13 - 19
invalid_cardname=Card name length should be in range 1 - 100
invalid_exp_year=Expiration year should be in range 01 - 99
invalid_exp_month=Expiration month should be in range 01 - 12
invalid_cvc2=Invalid CVC2 value
```

12.5. Localization for error messages.

To localize JavaScript error messages, in the card entry window the following object with messages in a required language should be defined in the template "cardinfo.html" before the tag "%%javascript%%":

```
<script language="JavaScript">
strLocale = new Object();
strLocale_loc.enter_cvc = 'Enter CVC2/CVV2 or CID code!';
strLocale_loc.enter_cid = 'Enter Amex CID code!';
strLocale_loc.enter_cvc2 = 'Enter CVC2/CVV2 code!';
strLocale_loc.enter_name = 'Enter name, surname!';
strLocale_loc.enter_cardnr = 'Enter card number!';
strLocale_loc.enter_expiry = 'Enter card expiry date!';
</script>
```

If "strLocale" is not defined, then messages will be in English same as in this example. To localize error messages, you can also use a localization file called **locale.properties**, which contains a description of the necessary translations. By default, the following client-side error messages apply:

```
enter_cardnr=Enter card number
enter_expiry=Enter card expiry date
enter_cardname=Enter name, surname
enter_cvc=Enter CVC2/CVV2 or CID code
enter_cid=Enter Amex CID code
enter_cvc2=Enter CVC2/CVV2 code
```