Paynet Api.e-com

Document Type:

Technical specification

Version:

ver. 0.5 Eng / 08.05.19

ver. **0.5** rus / **08.05.19**

Change history

Date	Version	Description
08.05.2019	0.5	Added the ability to transfer customer data to fill questionnaires Added the ability to specify what the client should pay The data signing block has been expanded (products, payment instrument, customer data).

Introduction

Paynet API - Paynet API is a software interface for interacting with Paynet system, which provides its customers with the opportunity to start an e-wallet, make payments and transfers, as well as take part in loyalty programs.

Requests are made via the HTTPS (HyperText Transfer Protocol Secure) protocol, which is an extension of the HTTP protocol that supports encryption. Data transmitted over the HTTPS protocol is "packed" into the TLS cryptographic protocol.
Requests are made to the address:

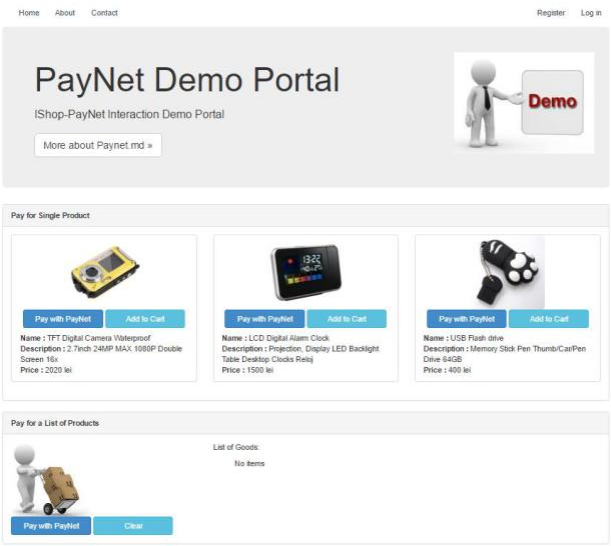
for testing: **https://test-provided_api_host/api/Payments**
for full interaction: https://provided_api_host/api/Payments

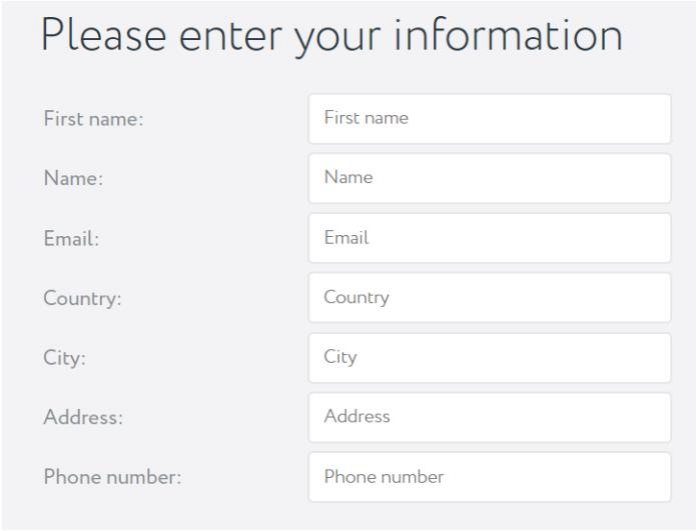
Paynet API consists of a set of services, broken down according to the functionality they implement.
Paynet API.e-com provides the possibility of legal person organize on their websites acceptance of payments for goods or services by electronic money of the client through the Paynet system.
In order to be able to organize on their websites acceptance of payments for goods or services by electronic money of the client through the Paynet system, legal person must become a Paynet Partner and select a scenario in which he wants to integrate with Paynet. To become a Paynet Partner, you must go through the registration of a merchant with Paynet.
As a result, the Partner will receive a merchant and user ID to access his account.

The client will be able to pay with electronic money directly in the payment form of the issuer. A partner, when transferring a client to a payment form, has the opportunity to specify the client's redirection addresses in case of successful payment or refusal to pay the payment..

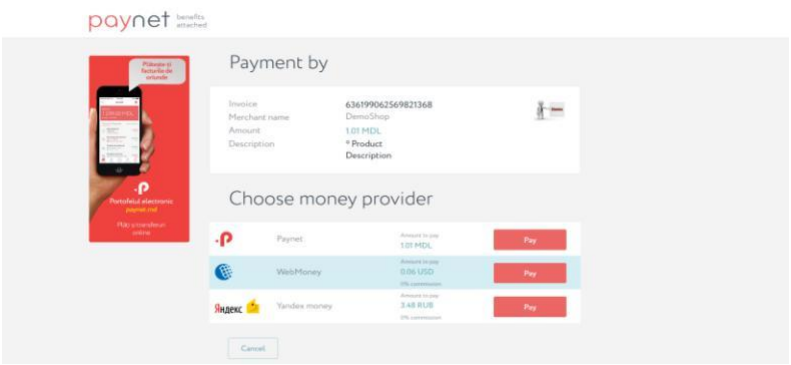
Scenario for accepting payments for goods and specified services on the partner's pages

1. The client is located on the partner's website and wishes to pay for the basket of goods / services of the partner, using the Paynet e-wallet or other agreed electronic money payment tools.

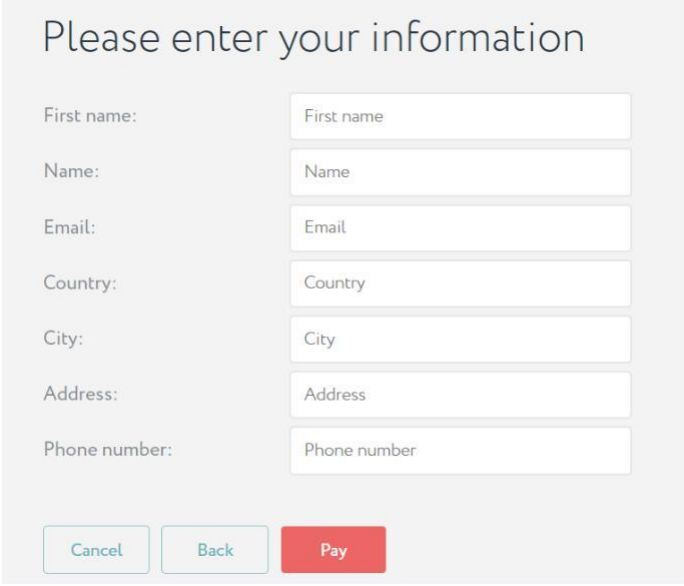




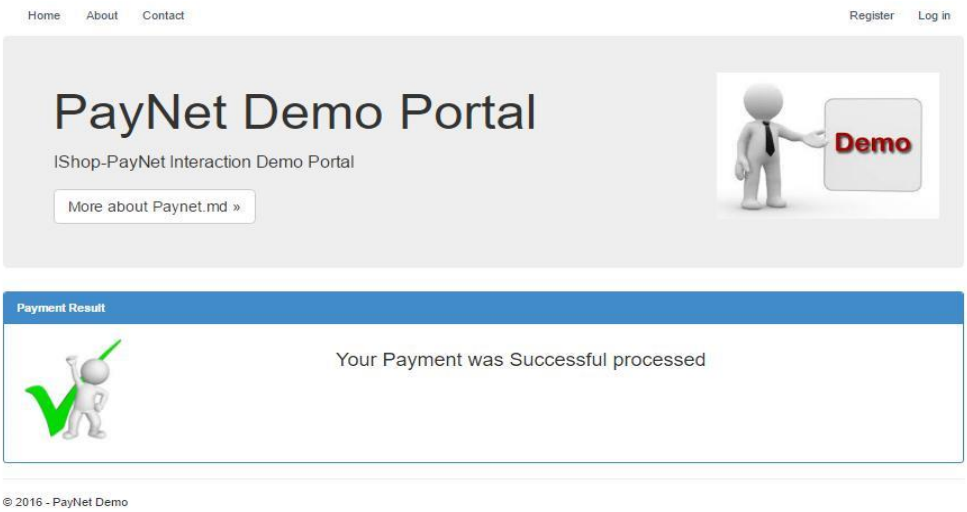
2. The partner can collect all the necessary information from the client (address contact general information about the client, the desired payment instrument) required in the payment process through the Paynet system (PSP),
3. The partner system sends to Paynet data about the payment (product (s) / service (s), customer questionnaire, payment method) and transfers the customer to Paynet
4. If the Partner does not transmit data about the payment method, the client will be asked to choose the payment method on the Paynet page.

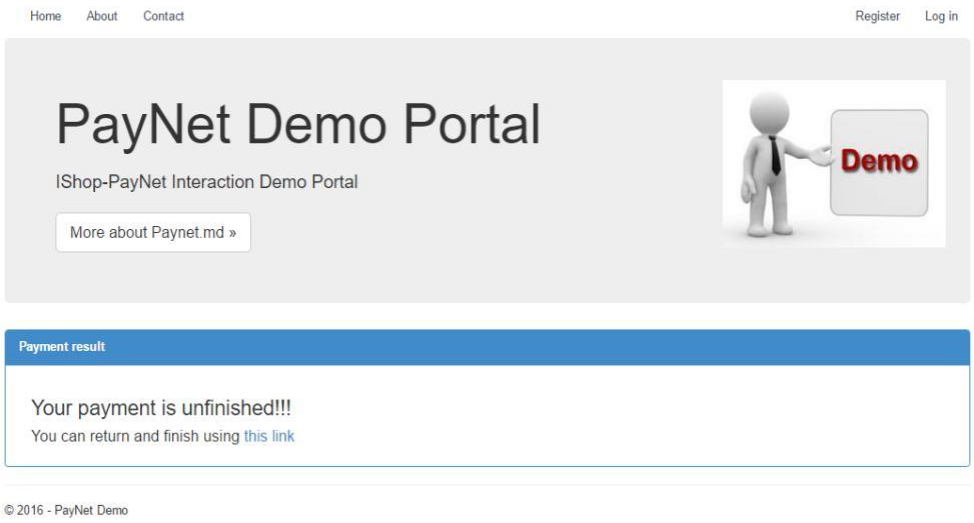


5. If the Partner does not transmit all the necessary information about the client (general contact information about the client), the client will be asked to fill out a questionnaire and continue or refuse to pay.



6. Depending on the chosen payment method, the client will be transferred to the payment form of the issuer of electronic money of the client for further actions (authentication and completion of the purchase).
7. After completing the payment, the customer will be notified of the payment status in the Paynet system and will be offered the opportunity to return to the merchant's shop
8. The client will be transferred to the addresses specified by the partner when the client is transferred to the page Paynet.
9. The Paynet system has the ability to notify the Partner's system via a pre-installed URL (see point o).





10. The Partner's system has the ability to check the status of the payment using the service: **,** Error! Reference source not found.**.**

Client transfer to the payment method selection page

(Send) Transferring Payment Information

Paynet system provides 2 payment data transfer models:

(**server -> server**) In this model, identification and establishment of communication between the Partner's servers and Paynet is required. Signing data is not required. The Partner's system will transmit the payment information by contacting Payments. As a result, Partner will receive a payment ID, through which the Partner will be able to send the client to the Paynet payment form to confirm and pay the payment.

Related data is required for integration on this model:

- Technical user with access to the service **Payments**
- ID Partner in the Paynet systemet

- (**client -> server**) this model information about the payment is transmitted from the client interface, which requires additional data protection. Accordingly, for data protection, it is customary to sign payment information before being sent to the payment form. (look at the signing algorithm)

-Related data is required for integration on this model:

-Secret key involved in signing. Known to both systems.

-ID Partner in Paynet

(Redirect) Client transfer to payment form

In order to redirect the client to the Paynet payment method selection page, a certain set of data should be transferred, depending on the selected payment data transfer model.

General data for the request to the payment form

Method	POST
Host	*provided_portal_host*

Additional data for the request to the payment form

1. Within the model **server -> server**

Url	Acquiring/*GetEcom*

Request body format **JSON:**

```
{
    "operation": PaymentID from Payment Service,
    "LinkUrlCancel”: ”http://url_cancel”,
    "LinkUrlSucces”: ”http://url_succes”,
    "Lang”:”en-US”
    "Signature”:”Hash Provided by Payment Service”
}
```

2. Within the client **model -> server**

Url	Acquiring/SetEcom

Request body format **JSON:**

```
{
    "ExternalID": 20160622010101,
    "Currency": 498,
    "Merchant": "MerchantCode"
    "SaleAreaCode":"sale_area_code" – can be empty
    "Customer": {
        "Code": "Customer Code",
        "NameFirst": "payer first name",
        "NameLast": "payer last name",
        "PhoneNumber": "customer/payer phone number",
        "email" : "customer/payer email",
        "Country": "customer country",
        "City": "customer city",
        "Address": "Address description"
    },
    "ExpiryDate": "2016-01-01T00:00:00",
    "Services": [{
        "Name": "Service Name 1",
        "Description": "Service Name Decription1"
        "Amount": 100,
        "products":[{
            "Amount"     : 100,
            "Barcode"    : 13243546,
            "Code"       : "PRODUCT1",
            "Description"    : "description of PRODUCT1",
            "GroupId"    : "22",
            "GroupName"  : "a group name of this product",
            "LineNo"     : 1,
            "Name"       : "product 1",
            "UnitPrice" : 100,
            "UnitProduct"    : 1
        }],
    }],
    "LinkUrlCancel”:”http://url_cancel”,
    "LinkUrlSuccess”:”http://url_success”,
    "Lang”:”en-US”,
    "Signature”:”xxxxxxxxxxxxxxxxxxxxxxxxxxxx”,
```

```
    "SignVersion":"v05",
    "MoneyType": {
            "Code" : "PAYNET"
    }
}
```

Authentication and authorization to the Paynet Web service

To access Paynet service and pass authorization, you need to get token authorization. You can get this token by calling the service method described below, specifying the parameters (username and password) received during partner registration.

(Auth) Authentication and token acquisition - and authorization

To call this method, you need a user and password issued by the owner of the Paynet system.

Request

Method	POST
Host	*provided_api_host*
Url	/auth
Content-Type	application/x-www-form-urlencoded
Body	grant_type=password&username=*username*&password=*password*

The answer is in the format JSON:

```
{
    "access_token": "provided_token_for_you",
    "token_type": "bearer",
    "expires_in": number_of_seconds
}
```

Description of the Web service Payments

This service is designed to register, search and obtain information about registered payments in the Paynet system. If successful (HTTP Response Code -> 200), the service method returns a JSON document containing an object or a list of Payments objects (presented below).

Reference field of the Payment object:

Field	Type	Description
PaymentID	long	Paynet Payment ID
Invoice	long	Partner System Payment ID
MerchantCode	string	Partner code in which the payment is made
SaleAreaCode	string	Partner site code in which payment is made
MoneyType	string	The code of the payment instrument from the list of payment instruments established by the contract. As an example, use the code "PAYNET"
Customer – Information about the Client on the side of the partner		
Code	string	Client code on partner system side
Address	string	Customer address

	NameFirst	string	Client name on the partner side
	NameLast	string	Customer’s name on partner’s side
	email	string	Client’s email address on the partner’s side
	PhoneNumber	string	Client's phone on the partner side
	Country	string	Client’s country on partner’s side
	City	string	Client's city on the partner side

Registered	DateTime	Date of payment registration. Value according to standard ISO 8601
Confirmed	DateTime	Date of payment confirmation by the client. Value according to standard ISO 8601
Processed	DateTime	Date of payment by the client. Value according to standard ISO 8601
Canceled	DateTime	Date of payment cancellation. Value according to standard ISO 8601
Currency	int	Currency code, value according to standard ISO 4217
ExpiryDate	DateTime	The term of payment by the client. Value according to standard ISO 8601
Amount	long	Amount to be converted to an integer value. Example: *12.01 => 1201.*
Signature	string	Hash client escort in Paynet
Status	enum	1 – Registered 2 – Customer Verified 3 – Initialized to be Paid 4 – Paid
SignVersion	string	Version API signing. Currently used version **0.5** ("v05")

Services — List of services provided by the partner

Name	string	Name of service
Description	string	Service Description

Products

Amount	long	The cost of the product in the integer format ... Example 12.34 => 1234
Barcode	long	Bar product code
LineNo	int	Product sequence number
GroupId	long	Product group id
Code	string	Code of product
Name	string	Product Name
Description	string	Extended Product Description
Quantity	long	the amount of product in integer format ... Example 12.01 ->1201
UnitPrice	long	The cost of one product unit in integer format ... Example 12.01 -> 1201

		UnitProduct	string	Product Type
	Amount		Int	Amount to be converted to integer value. Example: 12.01 => 1201.

In the case of detecting errors, the service method returns a JSON document that contains an Error object (shown below).

Reference field of the object Error: Error field reference book:

Field	Type	Description
Code	string	Error code
Message	string	Error description

(Send) Sending a payment document to Paynet

This service method is designed to send a payment document to Paynet system. As a result, the Partner receives the ID of the payment document on the Paynet side and has the opportunity to direct the client to the Paynet page to select the issuer where the purchase will be paid.

Directory of fields involved in the service method:

Field	Type	Description
Invoice	long	Partner System Payment ID
Customer – Information about the Client on the side of the partner		
Code	string	Client code on partner system side
Address	string	Customer address
NameFirst	string	Client name on the partner side
NameLast	string	Customer's name on partner's side
email	string	Client's email address on the partner's side
PhoneNumber	string	Client's phone on the partner side
Country	string	Client's country on partner's sid
City	string	Client's city on the partner side
Currency	int	Currency code, value according to standard ISO 4217
ExpiryDate	DateTime	The term of payment by the client. Value according to standard ISO 8601
Services – set of services included in the payment		
Name	string	Name of service
Description	string	Service Description
Products – set of products receivable as a result of payment for services		
Amount	long	The cost of the product in the integer format ... Example 12.34 => 1234

		Barcode	long	Bar product code
		LineNo	int	Product sequence number
		GroupId	long	Product group id
		Code	string	Код продукта
		Name	string	Product Name
		Description	string	Extended Product Description
		Quantity	long	The amount of product in integer format ... Example 12.01 ->1201
		UnitPrice	long	The cost of one product unit in integer format ... Example 12.01 -> 1201
		UnitProduct	string	Product Type
	Amount		Int	Amount to be converted to integer value. Example: 12.01 => 1201.
MerchantCode	string	Partner code in which the payment is made		
SignVersion	string	Version API signing. Currently used version **0.5** (item.: "v05")		
MoneyType	string	The code of the payment instrument from the list of payment instruments established by the contract.		

Request:

Method	POST
Host	*provided_api_host*
Url	/api/Payments
Content-Type	application/json
Header(Authorization)	*bearer provided_token_for_you*

```
{
    "Invoice": 20160622010101,
    "Currency": 498,
    "MerchantCode": "MerchantCode"
    "SaleAreaCode":"sale_area_code" - can be empty
    "Customer": {
        "Code": "Customer Code",
        "NameFirst": "payer first name",
        "NameLast": "payer last name",
        "PhoneNumber": "customer/payer phone number",
        "email" : "customer/payer email",
        "Country": "customer country",
        "City": "customer city",
        "Address": "Address description"
    },
    "ExpiryDate": "2016-01-01T00:00:00",
    "Services": [{
        "Name": "Service Name 1",
        "Description": "Service Name Decription1",
        "products": [{
            "Amount"    : 100,
            "Barcode"   : 13243546,
            "Code"      : "PRODUCT1",
            "Description"   : "description of PRODUCT1",
            "GroupId"   : "22",
```

```
            "GroupName" : "a group name of this product",
            "LineNo"    : 1,
            "Name"      : "product 1",
            "UnitPrice" : 100,
            "UnitProduct"   : 1
        }],
        "amount": 100
    }],
    "SignVersion":"v05",
    "MoneyType": {
             "Code" : "PAYNET"
    }
}
```


The answer is in the format JSON:


```
{
    "PaymentID": 45678901011,
    "Invoice": 20160622010101,
    "Customer": {
        "Code": "Customer Code",
        "Name": "payer full name",
        "Address": "Address description",
        "NameFirst": "payer first name",
        "NameLast": "payer last name",
        "PhoneNumber": "customer/payer phone number",
        "email" : "customer/payer email",
        "Country": "customer country",
        "City": "customer city"
    },
    "MerchantCode": "MerchantCode",
    "SaleAreaCode":"sale_area_code" - can be empty
    "Registered": "2016-01-01T00:00:00",
    "Currency": 498,
    "ExpiryDate": "2016-01-01T00:00:00",
    "Services": [{
        "Name": "Service Name 1",
        "Description": "Service Name Decription1",
        "products": [{
            "Amount"    : 100,
            "Barcode"   : 13243546,
            "Code"      : "PRODUCT1",
            "Description"   : "description of PRODUCT1",
            "GroupId"   : "22",
            "GroupName" : "a group name of this product",
            "LineNo"    : 1,
            "Name"      : "product 1",
            "UnitPrice" : 100,
            "UnitProduct"   : 1
        }],
        "amount": 100
    }],
    "Status": 1
}
```


Error Code Options:

HTTP Response Code	Detailed Code system errors Paynet	General description of the error	Actions
401- Unauthorized request	3	User not found	It is proposed to go again authenticate and get new token
	3080		

	14		Enter the appropriate data

400- Validation error	25	Invalid parameter set either their meanings	
500- Paynet server error	82	The error received in the process init operation	Use the Search method to try to find payment and check its status
	2		
	11		
	4		
	73		
201- Resource created	10	Payment document already registered	Service- Search method check availability of a registered payment
	202		
404- Resource was not found	68	The client does not participate in the program loyalty	Make sure everything is correct.

(Get) Receive information about a registered payment

This service method is designed to obtain information about the payment. It can be used in case of communication problems at the time of payment or using information about the operation on the side of the partner's system.

Request:

Method	Get
Host	*provided_api_host*
Url	/api/Payments/*id*
Content-Type	application/json
Header(Authorization)	*bearer provided_token_for_you*

The answer is in the format JSON:

```
{
    "PaymentID": 45678901011,
    "Invoice": 20160622010101,
    "MerchantCode": "MerchantCode",
    "SaleAreaCode":"sale_area_code" - can be empty
    "Customer": {
        "Code": "Customer Code",
        "Address": "Address description",
        "NameFirst": "payer first name",
        "NameLast": "payer last name",
        "PhoneNumber": "customer/payer phone number",
        "email" : "customer/payer email",
        "Country": "customer country",
        "City": "customer city"
    },
    "Registered": "2016-01-01T00:00:00",
    "Confirmed": "2016-01-01T00:00:00",
    "Processed": "2016-01-01T00:00:00",
    "Canceled": "2016-01-01T00:00:00",
    "Currency": 498,
    "ExpiryDate": "2016-01-01T00:00:00",
    "Services": [{
        "Name": "Service Name 1",
        "Description": "Service Name Decription1",
        "products": [{
            "Amount"    : 100,
            "Barcode"   : 13243546,
```


```
            "Code"       : "PRODUCT1",
            "Description"    : "description of PRODUCT1",
            "GroupId"    : "22",
            "GroupName" : "a group name of this product",
            "LineNo"     : 1,
            "Name"       : "product 1",
            "UnitPrice" : 100,
            "UnitProduct"    : 1
        }]
        "amount": 100
    }],
    "Status": 4
}
```


Error Code Options:

HTTP Response Code	Detailed Code mistakes paynet systems	General description of the error	Actions
401 - Unauthorized request	3	User not found	It is proposed to authenticate again and get a new token
	3080		
400 - Validation error	14	Invalid parameter set or their meanings	Enter the appropriate data
	25		
500 - Paynet server error	82	Error received in the search process	Try calling this method again.
	2		
	11		
	73		
404 - Resource was not found	64	No payment found	Use the advanced search method Search. Recheck Payment ID

(Search) Search for payments by specified criteria

This service method is designed to search for payments by the specified criteria. Available criteria are listed below. It can be used in case of communication problems at the time of registration or use of information on payment on the side of the partner's system.

Request:

Method	GET
Host	*provided_api_host*
Url	/api/Payments?*QueryString*
Content-Type	application/json
Header(Authorization)	*bearer provided_token_for_you*

Критерии поиска:

Параметр	Тип	Описание
Invoice	long	payment identifier on the external system side
from	string	Start date of the period in which the payment was made. Value according to standard y ISO 8601
to	string	The end date of the period in which the payment was made. Value according to standard ISO 8601

The answer is in the format JSON:


```
[{
    "PaymentID": 45678901011,
    "Invoice": 20160622010101,
    "MerchantCode": "MerchantCode",
    "SaleArea":{
          "Code": "SaleAreaCode"
    },
    "Customer": {
        "Code": "Customer Code",
        "Name": "Customer full name",
        "Address": "Address description",
        "NameFirst": "payer first name",
        "NameLast": "payer last name",
        "PhoneNumber": "customer/payer phone number",
        "email" : "customer/payer email",
        "Country": "customer country",
        "City": "customer city"
    },
    "Registered": "2016-01-01T00:00:00",
    "Confirmed": "2016-01-01T00:00:00",
    "Processed": "2016-01-01T00:00:00",
    "Canceled": "2016-01-01T00:00:00",
    "Currency": 498,
    "ExpiryDate": "2016-01-01T00:00:00",
    "Services": [{
        "Name": "Service Name 1",
        "Description": "Service Name Decription1",
        "products": [{
            "Amount"     : 100,
            "Barcode"    : 13243546,
            "Code"       : "PRODUCT1",
            "Description"    : "description of PRODUCT1",
            "GroupId"    : "22",
            "GroupName" : "a group name of this product",
            "LineNo"     : 1,
            "Name"       : "product 1",
            "UnitPrice" : 100,
            "UnitProduct"    : 1
        }]
        "amount": 100
    }],
    "Status": 4
}]
```


Варианты кодов ошибок:

HTTP Response Code	Detailed Code system errors Paynet	General description of the error	Actions
401- Unauthorized request	3	User not found	It is proposed to authenticate again. and get a new token
	3080		
400 - Validation error	14	Invalid parameter set or their meanings	Укажите соответствующие данные
	25		
500 - Paynet server error	82	Ошибка, полученная в процессе поиска	Try calling the service again. Method
	2		
	11		
	73		
404 - Resource was not found	64	No payment found	Recheck search criteria

Paynet system notifications

The Paynet system is able to notify the Partner's system about the status of the payment. To do this, you must send the address of the service of the Partner's system to which notifications will be sent.

The Paynet system signs the notification with a secret key known to both systems.

Directory of notification fields:

Field	Type	Description
Eventid	long	Notification ID
EventType	string	PAID
EventDate	DateTime	Notification date. Value according to standard ISO 8601
Payment – payment information		

ID	long	Paynet Payment ID
ExternalID	long	Partner System Payment ID
Merchant	string	Код партнёра, в рамках которого совершается платёж
Customer	string	Partner code in which the payment is made
StatusDate	DateTime	Date of transfer to specified payment status. Value according to ISO 8601
Amount	long	Amount to be converted to an integer value. Example: 12.01 => 1201.

(Paid Notification) Notification of the Partner's system about the completed payment transaction

This service method is designed to send the status of the payment document to the Partner's system.

As a result, the Partner receives general information about the payment document with its status and can make a decision stipulated by the Partner's system.

Notification data is signed by a secret key (SecretKey). The secret key is agreed between the Partner's system and the Paynet system. And should not be transferred to the 3rd party.

Notification will be sent after the Issuer sends information about successful payment to the client.

In the event of communication problems, the Paynet system will eventually try to contact the Partner's system 3 times and, in the event of a "failure," will complete the notification attempts. To obtain information about the payments made, special service methods have been developed in the Payments service.

The Paynet system believes that the logic of processing duplicate notifications is built on the side of the Partner's system. The unique key is the payment ID.

"Payment.ID" you

Directory of fields involved in the service method:

Method	POST
Host	*provided_api_host*
Url	*MerchantSystemUrl*
Content-Type	application/json

Header(Authorization)	*bearer provided_token_for_you*
Header(Hash)	*HashValue*

Request:

```
{
    "Eventid": 20160622010101,
    "EventType": "Paid",
    "EventDate": "2016-01-01T00:00:00",
    "Payment": {
        "ID": 1234567,
        "ExternalID": 7676766,
        "Merchant": "123123",
        "Customer": "37369XXX111",
        "StatusDate": "2016-01-01T00:00:00",
        "Amount": 123
    }
}
```

Answer

As a successful response, Paynet expects http status equal to 200. What is considered successful. In any other status, Paynet will try to apply with a notification up to 3 times, in case of no success, it will complete its attempts to send this notification.

```
{
    "Eventid": 20160622010101,
    "EventType": "Paid",
    "EventDate": "2016-01-01T00:00:00",
    "Payment": {
        "ID": 1234567,
        "ExternalID": 7676766,
        "Merchant": "123123",
        "Customer": "37369XXX111",
        "StatusDate": "2016-01-01T00:00:00",
        "Amount": 123
    }
    “ResultCode”:”SUCCESS”,
    “ResultMessage”:”Success Message”,
}
```

Signature generation

To guarantee the integrity of the data and verify their authenticity, it was decided to protect the data with a signature.

MD5 encryption method is selected for signature generation, followed by Base64 encoding.

```
signature = ToBase64String(MD5(PreparedString + secretKey))
```

Sample Signature Code (C#):

```
var paramsDictionary = new SortedDictionary<string, string>();

paramsDictionary.Add("EVENTDATE", "???????????");
paramsDictionary.Add("EVENTID", "???????????");
paramsDictionary.Add("EVENTTYPE", "???????????");
paramsDictionary.Add("PAYMENT.AMOUNT", "???");  Integer value In pennies            100 = 1 lei
paramsDictionary.Add("PAYMENT.CUSTOMER", "?????");
paramsDictionary.Add("PAYMENT.EXTERNALID", "???????????");
paramsDictionary.Add("PAYMENT.ID", "???????????");
paramsDictionary.Add("PAYMENT.MERCHANT", "???????????");
paramsDictionary.Add("PAYMENT.STATUSDATE", "???????????");

var signatureData = new StringBuilder();

foreach (string key in paramsDictionary.Keys)
{
       signatureData.Append(paramsDictionary[key]);
}

string secretKey = "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"; //The secret key known to both
systems, but not transmitted in the clear as the system interacts. And not passed to the 3rd
party

string message = signatureData + secretKey;
Byte[] bytes = Encoding.GetEncoding(1251).GetBytes(message);
Byte[] hash = new MD5CryptoServiceProvider().ComputeHash(bytes);

string signature = Convert.ToBase64String(hash);
```

(SignNotification) An example of forming a string for signing notification data

Notification Object :

```
{
    "Eventid": 20160622010101,
    "EventType": "Paid",
    "EventDate": "2016-01-01T00:00:00",
    "Payment": {
        "ID": 1234567,
        "ExternalID": 7676766,
        "Merchant": "123123",
        "Customer": "37369XXX111",
        "StatusDate": "2016-01-01T00:00:00",
        "Amount": 123
    }
}
```

The string for signing is formed as follows:

```
PreparedString = EventDate + Eventid + EventType + Payment.Amount + Payment.Customer +
Payment.ExternalID + Payment.ID + Payment.Merchant + Payment.StatusDate
```

(!) The sequence is strictly defined. By the principle of sorted field reference in alphabetical order.

(SignPayment) Example of the signing payment data

Payment object

```
{
    "ExternalID": 20160622010101,
    "Currency": 498,
    "Merchant": "MerchantCode",
    "SaleAreaCode":"sale_area_code" - can be empty
    "Customer": {
        "Code": "CUSTOMERCODE",
        "NameFirst": "payer first name",
        "NameLast": "payer last name",
        "PhoneNumber": "customer/payer phone number",
        "email" : "payer@paynet.md",
        "Country": "customer country",
        "City": "customer city",
        "Address": "Address description"
    },
    "ExpiryDate": "2016-01-01T00:00:00",
    "Services": [{
            "Name": "Service Name 1",
            "Description": "Service Name Decription1"
            "amount": 100,
            "Products": [{
                "Amount"    : 100,
                "Barcode"   : 13243546,
                "Code"      : "PRODUCT1",
```

```
                "Description"    : "description of PRODUCT1",
                "GroupId"    : "22",
                "GroupName"  : "a group name of this product",
                "LineNo"     : 1,
                "Name"       : "product 1",
                "UnitPrice"  : 100,
                "UnitProduct"    : 1
            }]
    }],
    "MoneyType": {
            "Code" : "PAYNET"
    }
}
```

The string for signing is formed as follows::

```
PreparedString = Currency + Customer.Address + Customer.City +Customer.Code +
Customer.Country + Customer.email + Customer.NameFirst + Customer.NameLast +
Customer.PhoneNumber+ ExpiryDate + ExternalID + Merchant + MoneyType.Code
foreach(var service in Services)
{
    PreparedString = PreparedString + service.Amount + service.Description
+ service.Name
    foreach(var product in Products)
    {
        PreparedString = PreparedString + product.Amount + product.Barcode
+ product.Code
                + product.Description + product.GroupId + product.GroupName
                + product.LineNo + product.Name + product.UnitPrice + prod-
uct.UnitProduct
    }
}
```