

EuPlatesc.ro Gateway

Standard Merchant interface

**Merchant integration in the electronic commerce – EuPlatesc.ro Gateway
based on 3DSecure Standard**

(Visa and MasterCard)

Overview

This manual is intended for use by programmers responsible for the merchant payment module interface with the gateway. It describes the interface that merchant systems use to process credit card based e-commerce transactions using the standard HTTP forms posting method.

This manual covers the following payment aspects:

1. General transaction processing
2. Recurring transaction processing
3. Installment transaction processing
4. Discount/loyalty transaction processing
5. Processing via webservice
6. SMS payments processing
7. Email orders
8. Management messages
9. Authorisation sequence diagram
10. Recurring payment sequence diagram
11. Payment processing sequence diagram
12. SOAP webservices description
13. Java, PHP, C# code example.

1 General transaction processing

Message Structure - Authorisation Request

The following fields set will be posted to EuPlatesc.ro platform through the HTTP POST method. The set of fields are divided into 2 sections: fields included in digital signature of the message (fp_hash) and fields not included in the digital signature of the message (fp_hash).

Table 1. Fields generated by merchant system and included into the fp_hash

Field name	Format	Length	Description
amount	Decimal	1-12	Order total amount in float format with decimal point (thousand separator not allowed). Ex: 1234.56
curr	String	03	Order currency: 3-character currency code (RON, USD, EUR)
invoice_id	Numeric	6-27	Merchant order ID
order_desc	String	1-50	Order description
merch_id	String	8-50	Merchant ID assigned by EuPlatesc.ro
timestamp	YYYYM MDDHH MMSS	14	Merchant transaction timestamp in GMT: YYYYMMDDHHMMSS.
nonce	String	16-64	Merchant nonce. Must be filled with unpredictable random bytes in hexadecimal format
fp_hash	String	1-256	Merchant MAC in hexadecimal form.

Table 2. Fields generated by merchant system and NOT included into the fp_hash

Field name	Format	Length	Description
Billing details			
fname	String	1-255	Client first name
lname	String	1-255	Client last name
company	String	1-255	optional
add	String	1-255	Client street
city	String	1-255	Client city
state	String	1-255	Client state - optional
zip	String	1-25	Client postal code
country	String	1-55	Client country
phone	String	1-225	Client phone
fax	String	1-25	Client fax - optional
email	String	1-65	Client email
Shipping Details – if there are different that billing details			
sfname	String	1-255	Client first name
slname	String	1-255	Client last name
scompany	String	1-255	optional
sadd	String	1-255	Client street
scity	String	1-255	Client city
sstate	String	1-255	Client state - optional
szip	String	1-25	Client postal code
scountry	String	1-55	Client country
sphone	String	1-25	Client phone
sfax	String	1-25	Client fax - optional
semail	String	1-55	Client email
Extra information sent by the merchant to the gateway			
ExtraData	String	0-10240	Additional information sent by the merchant to the gateway. This data will be posted back to the merchant during silent_reply.

Table 3. EuPlatesc.ro response fields set

Response can be sent back to the merchant in 3 ways:

- replay via client`s browser
- silent reply via server to server method
- both methods defined above

Field name	Format	Length	Description
amount	Numeric	1-12	Echo from the request
curr	String	03	Echo from the request
invoice_id	Numeric	6-32	Echo from the request
ep_id	String	1-50	Gateway unique id for each transaction.
merch_id	String	8-50	Echo from the request
action	Numeric	1	If 0 – transaction approved else transaction failed.
message	String	1-50	Response code text message.
approval	String	06	Client bank`s approval code. Can be empty if not provided by gateway.
timestamp	YYYYM MDDHH MMSS	14	Merchant transaction timestamp in GMT: YYYYMMDDHHMMSS.
nonce	String	1-64	Merchant nonce. Must be filled with 8-32 unpredictable random bytes in hexadecimal format
fp_hash	String	1-256	Merchant MAC in hexadecimal form.
Extra information sent by the merchant to the gateway			
ExtraData	String	0-10240	Additional information sent by the merchant to the gateway. This data is posted back to the merchant during silent_reply.

2 Recurring transaction processing

An additional field must be sent to the gateway, in order to process recurring transactions:

```
<input name="recurent" type="hidden" />
```

Recurring transactions are splitted into two parts:

- initial transaction – *recurent* value is “Base”
- subsequent recurring transactions - *recurent* value is “Recurent”. In case of ”Recurent” message, the merchant must send also:

```
<input type="text" NAME="baseEPID" VALUE="" />
```

“baseEPID” value is “ep_id” of the initial “Base” transaction.

All the other fields from the general message structure must be sent for successfully transaction processing.

In order to process recurring transactions, euplatesc.ro system must be configured in advance.

3 Installment transaction processing

An additional field must be sent to the gateway, in order to process installment transactions:

```
<input name="ExtraData[rate]" type="hidden" value="banca-3" />
```

Where:

- *banca* - the issuing bank used for installments. Allowed values are:
 - rzb – Raiffeisen Bank
 - bcr – Romanian Commercial Bank
 - apb – Alpha Bank
 - btrl – Transilvania Bank
- 3 – installments allowed

If the number of installment is not sent, euplatesc.ro payment interface will display a dropdown list, with allowed values. In this way, the client will be able to choose the number of installments directly on the payment interface.

In order to process installment transactions, euplatesc.ro system must be configured in advance.

4 Discount/loyalty transaction processing

General request message structure is used in order to process this type of transactions.

For merchants that are included into the loyalty processing scheme, the `silent_reply` will contain the following data also:

```
$extradata['applied_discount_info'] = array(  
    'org_amount'      => , // original amount received form merchant  
    'discounted_ammount' => , // the discount amount substracted from original amount  
    'discount'        => , // the applied discount (percent)  
    'discount_message' => , // description of applied rule  
);
```

Euplatesc.ro system must be configured in advance to process this transactions.

5 Processing via webservice

6 SMS payments processing

All the fields from the general message structure must be sent for successfully transaction processing.

For SMS payment processing, euplatesc.ro system must be configured in advance.

7 Email orders

8 Management messages

Merchant MAC – Message Authentication Code (fp_hash value)

To authenticate transaction messages on EuPlatesc.ro to/from the merchant link, the merchant system should be able to calculate and verify message authentication codes. The merchant system should be able to redirect transactions through cardholder browser, and to send messages directly to EuPlatesc.ro 3Dsystem. MAC is calculated over all fields generated by the merchant system as defined in corresponding format tables (visible and hidden fields generated by the merchant system) except the MAC field (“fp_hash”) itself.

In order to generate or verify the message authentication field, the merchant system must assemble a MAC source string; all field values from the format tables are prefixed with the decimal field length in ASCII and concatenated in the specified order. The default MAC algorithm is HMAC_MD5.

Payment message example

Suppose that we have a transaction with following fields:

Field	Length	Value
amount	5	100.00
curr	3	RON
invoice_id	7	6233097
order_desc	5	Shoes
merch_id	11	testaccount
timestamp	14	20060826054802
nonce	32	e15800a1f52ab6b42e852a9943a6a72a

MAC source string for this example is:

6100.003RON762330975Shoes11testaccount142006082605480232e15800a1f52ab6b42e852a9943a6a72a

Line breaks are inserted for visibility only.

After the MAC source string is assembled, the merchant system must apply a cryptographic algorithm to generate the message authentication code (HMAC_MD5). The merchant system must implement the encryption algorithm either in hardware or software form and be fully responsible for the secure storage and usage of corresponding cryptographic key.

For our MAC source string example and HMAC_MD5 algorithm with hexadecimal secret key “00112233445566778899AABBCCDDEEFF”, the result MAC (“fp_hash”) field must be equal to:

340f3874744bc5710e6eebe386286a64

Transaction Flow Scenario

1. After selecting goods and services, the cardholder presses ‘Buy’ or an equivalent button and proceeds to a page where he can enter or modify delivery information and the payment method. Payment method information may offer various payment methods, like ‘Pay by credit card’ or a similar option.

This option should not include card number, expiry date, CVC2 or any other card related sensitive information. Because of security risks involved, for the merchant system is mandatory to avoid requesting and storing credit card information on the his servers.

2. If cardholder selects ‘Pay by credit card’ option, merchant system must prepare authorization request fields (form) and redirect the cardholder to an ‘Enter credit card information’ webpage on e-Commerce Gateway system..

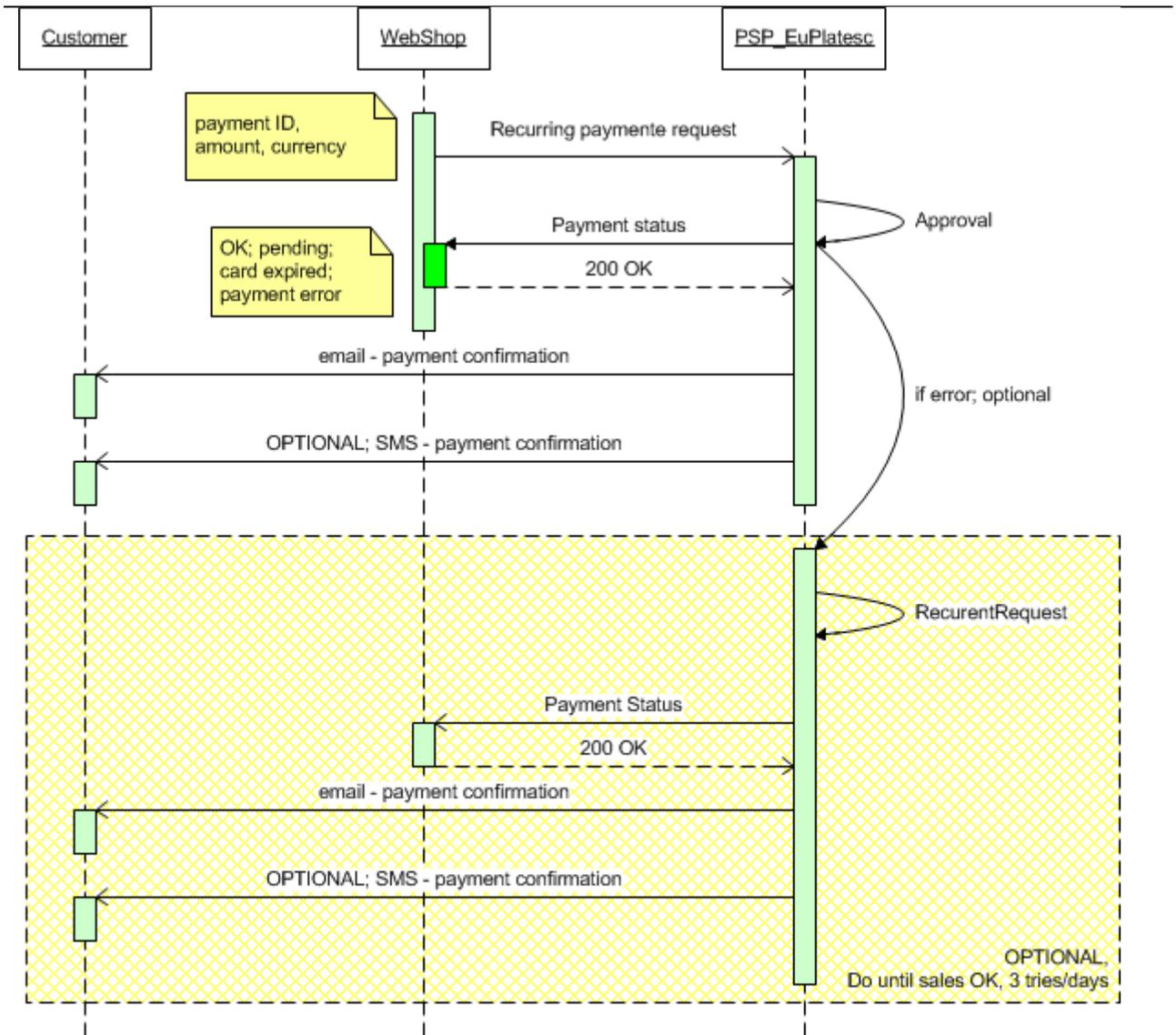
3. After receiving the filled-in form, e-Commerce Gateway validates request information including the message authentication code.

4. Upon authorization, reception gateway prepares and sends a transaction response back to the merchant system. Gateway sends response messages to the merchant system using HTTP POST redirect.

5. After receiving the online transaction response, the merchant system starts delivery of ordered goods and/or services to the cardholder. At this point, the requested amount is blocked on the cardholder account. Merchant should send an e-mail invoice message to the cardholder with order information and delivery time if applicable.

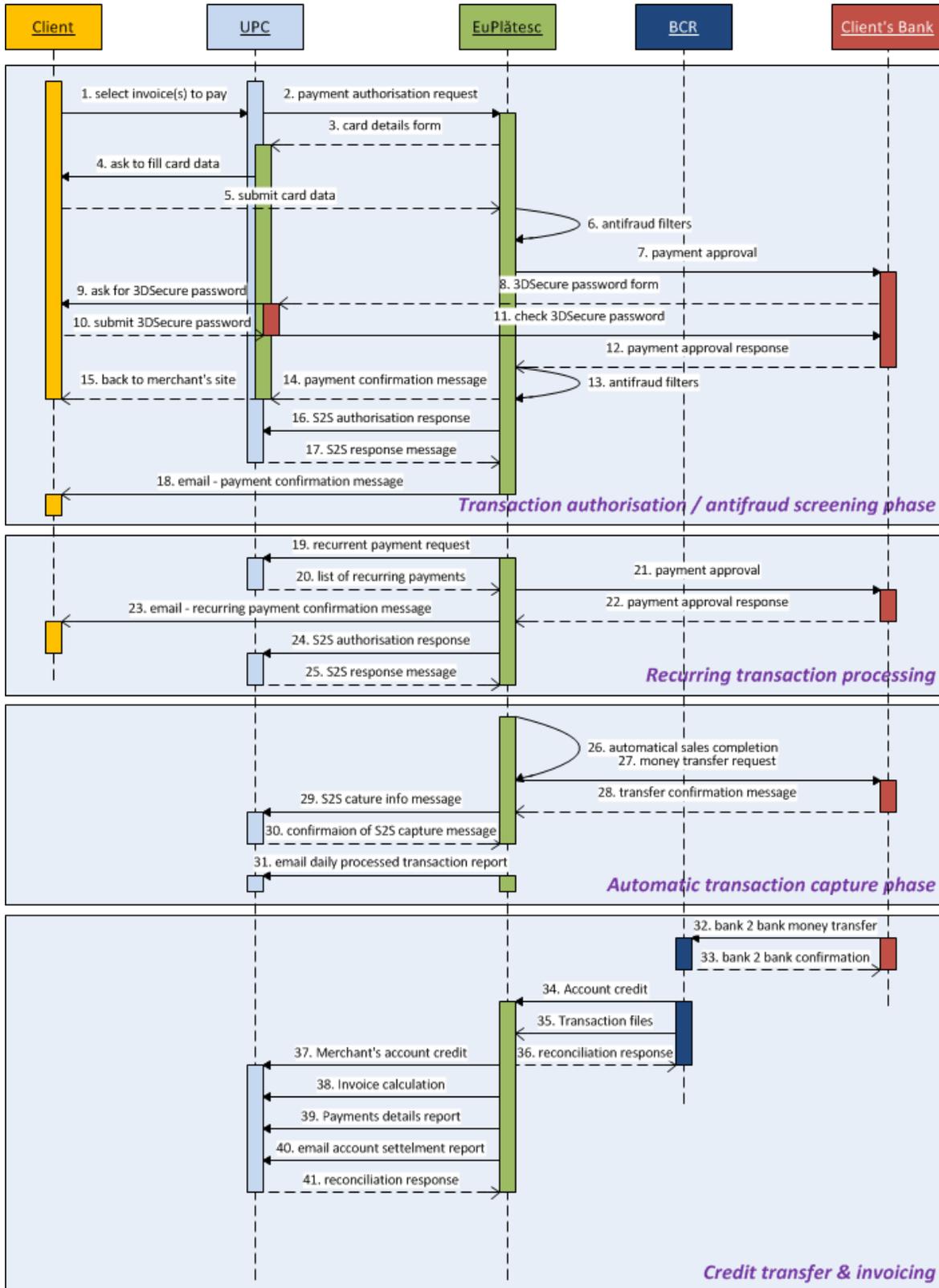
6. When the merchant has the confirmation that the goods/services has been delivered to cardholder, the merchant sends a “Capture” message from that gateway using „Capture tool” available into the gateway.

7. If the merchant is unable to fulfill the cardholder order or if the cardholder cancels the order at a stage allowed by the merchant, the merchant must send a “Reversal” message to cancel the pending or completed transaction, using the specific tool available into the gateway.



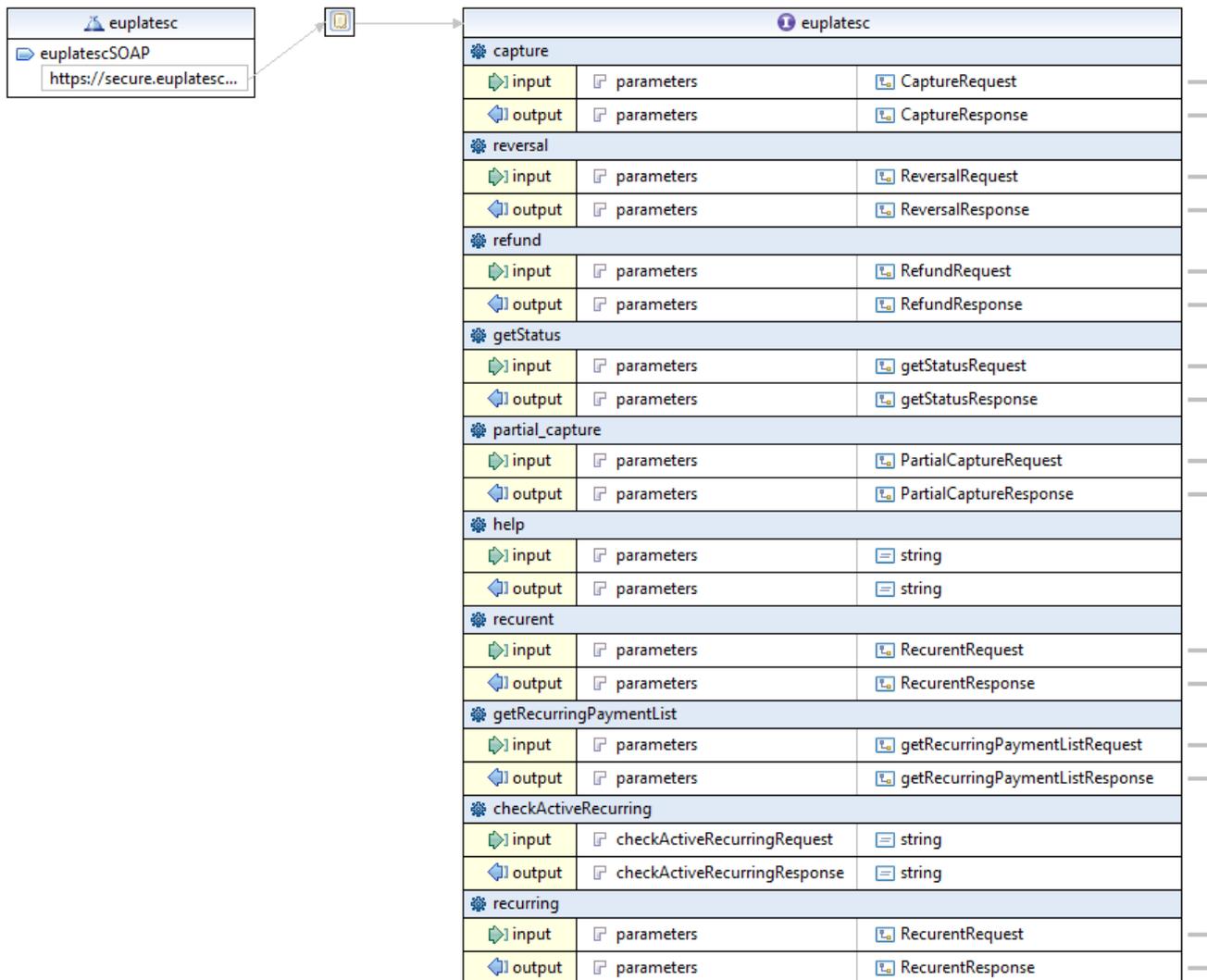
Recurring payment EuPlatesc sequence diagram

Payment processing flow



WebService calls:

The following SOAP webservices are available:



CaptureRequest		
e	ep_id	[1..1] string
e	mid	[1..1] string
e	amount	[1..1] float
e	curr	[1..1] string
...	delivery_awb	[0..1] string
e	delivery_by	[0..1] string
e	timestamp	[1..1] string
e	nonce	[1..1] string
e	fp_hash	[1..1] string

CaptureResponse		
e	ep_id	[1..1] string
e	mid	[1..1] string
e	status	[1..1] int
...	message	[1..1] string
e	timestamp	[1..1] string
e	nonce	[1..1] string
e	fp_hash	[1..1] string

RefundRequest		
e	ep_id	[1..1] string
e	mid	[1..1] string
e	amount	[1..1] float
e	curr	[1..1] string
...	reason	[1..1] string
e	timestamp	[1..1] string
e	nonce	[1..1] string
e	fp_hash	[1..1] string

RefundResponse		
e	ep_id	[1..1] string
e	mid	[1..1] string
e	status	[1..1] int
...	message	[1..1] string
e	timestamp	[1..1] string
e	nonce	[1..1] string
e	fp_hash	[1..1] string

ReversalRequest		
e	ep_id	[1..1] string
e	mid	[1..1] string
e	amount	[1..1] float
...	curr	[1..1] string
e	timestamp	[1..1] string
e	nonce	[1..1] string
e	fp_hash	[1..1] string

ReversalResponse		
e	ep_id	[1..1] string
e	mid	[1..1] string
e	status	[1..1] int
...	message	[1..1] string
e	timestamp	[1..1] string
e	nonce	[1..1] string
e	fp_hash	[1..1] string

getStatusRequest		
e	ep_id	[1..1] string
e	mid	[1..1] string
...	e	what [0..1] int
e	timestamp	[1..1] string
e	nonce	[1..1] string
e	fp_hash	[1..1] string

getStatusResponse		
e	what	[1..1] int
e	ep_id	[1..1] string
e	mid	[1..1] string
e	status	[1..1] int
e	message	[1..1] string
e	transaction_status	[1..1] string
e	transaction_status_msg	[0..1] string
e	transaction_channel	[1..1] string
e	transaction_type	[1..1] string
e	approval_date	[1..1] string
e	capture_date	[1..1] string
e	refund_date	[1..1] string
...	e	original_value [1..1] float
e	curr	[1..1] string
e	refunded_value	[1..1] float
e	discount_amount	[0..1] float
e	effective_captured_value	[1..1] float
e	fee	[0..1] float
e	installments	[0..1] int
e	installment_bank	[0..1] string
e	rrn	[0..1] string
e	cbk	[0..1] string
e	settlement	[0..1] string
e	timestamp	[1..1] string
e	nonce	[1..1] string
e	fp_hash	[1..1] string

PartialCaptureRequest		
e	ep_id	[1..1] string
e	mid	[1..1] string
e	amount	[1..1] float
e	amount_to_refund	[1..1] float
e	curr	[1..1] string
...	e	delivery_awb [0..1] string
e	delivery_by	[0..1] string
e	timestamp	[1..1] string
e	nonce	[1..1] string
e	fp_hash	[1..1] string

PartialCaptureResponse		
e	ep_id	[1..1] string
e	mid	[1..1] string
e	status	[1..1] int
...	e	message [1..1] string
e	timestamp	[1..1] string
e	nonce	[1..1] string
e	fp_hash	[1..1] string

RecurentRequest			
e	base_ep_id	[1..1]	string
e	mid	[1..1]	string
e	amount	[1..1]	float
e	curr	[1..1]	string
e	invoice_id	[1..1]	string
e	oredr_desc	[1..1]	string
e	recurent	[1..1]	string
e	timestamp	[1..1]	string
...	e	nonce	[1..1] string
e	fp_hash	[1..1]	string
e	delivery_awb	[0..1]	string
e	delivery_by	[0..1]	string
e	fname	[1..1]	string
e	lname	[1..1]	string
e	email	[1..1]	string
e	phone	[0..1]	string
e	ExtraData	[0..1]	string

RecurentResponse			
e	amount		float
e	curr		string
e	invoice_id		string
e	base_ep_id		string
e	ep_id		string
e	merch_id		string
...	e	status	int
e	message		string
e	approval		string
e	timestamp		string
e	nonce		string
e	fp_hash		string
e	ExtraData		string

getRecurringPaymentListRequest			
e	ep_id	[1..1]	string
e	mid	[1..1]	string
e	get_recurring_type	[1..1]	int
e	timestamp	[1..1]	string
e	nonce	[1..1]	string
e	fp_hash	[1..1]	string

getRecurringPaymentListResponse			
e	ep_id	[1..1]	string
e	mid	[1..1]	string
e	status	[1..1]	int
e	message	[1..1]	string
...	e	recurring_lits	[0..*] (recurring_litsType)
e	timestamp	[1..1]	string
e	nonce	[1..1]	string
e	fp_hash	[1..1]	string

(recurring_litsType)			
e	merchant_name	[1..1]	string
e	merchant_url	[1..1]	string
e	merchant_description	[1..1]	string
e	recurrent_id	[1..1]	string
e	recurrent_frequency	[1..1]	string
e	recurrent_expiration_date	[1..1]	string
e	recurrent_maximum_amount	[1..1]	string
e	recurrent_canceled_on	[1..1]	string
e	recurrent_canceled_reason	[1..1]	string
e	recurrent_masked_card	[1..1]	string
e	payment_ep_id	[1..1]	string
e	payment_order_desc	[1..1]	string
e	payment_invoice_id	[1..1]	string
e	payment_amount	[1..1]	float
e	payment_currency	[1..1]	string
e	payment_action	[1..1]	int
e	payment_message	[1..1]	string
e	payment_timestamp	[1..1]	string

RecurentRequest		
e	base_ep_id	[1..1] string
e	mid	[1..1] string
e	amount	[1..1] float
e	curr	[1..1] string
e	invoice_id	[1..1] string
e	oredr_desc	[1..1] string
e	recurent	[1..1] string
e	timestamp	[1..1] string
...	e	nonce [1..1] string
e	fp_hash	[1..1] string
e	delivery_awb	[0..1] string
e	delivery_by	[0..1] string
e	fname	[1..1] string
e	lname	[1..1] string
e	email	[1..1] string
e	phone	[0..1] string
e	ExtraData	[0..1] string

RecurentResponse		
e	amount	float
e	curr	string
e	invoice_id	string
e	base_ep_id	string
e	ep_id	string
e	merch_id	string
...	e	status int
e	message	string
e	approval	string
e	timestamp	string
e	nonce	string
e	fp_hash	string
e	ExtraData	string

```
/**
 * class CaptureRequest
 *
 * @author Stefan
 *
 * CaptureRequest structure is an array or class that extends DORquest = the structure that represent transaction to be captured.
 * The structure is
 * ep_id <string[40]> <required> is the transaction's unique ID receive from the payment system after a successful approval
 * mid <string[15]> <required> is the merchant terminal. This is defined on opening a merchant account on EuPlatesc system
 * amount <float[10.2]> <required> is the amount of the approved transaction.
 * curr <string[3]> <required> is the currency of the approved amount
 * delivery_awb <string> <optional> if missing set it to NULL, is the delivery document (AWB or invoice number) used in case of chargeback
 * fp_hmac, timestamp, nonce value are overwrite after each call of function addHMAC($key, true)
 * timestamp <yyyymmddhhiss> <required> is the request timestamp set by the merchant system on sending this request. Filled on __constructor if is NULL.
```

* nonce hexadecimal string <filled on instance> is random generated on each call. Represents the salt for fp_hash calculation and is used to identify duplicate messages.
* fp_hash hexadecimal string <filled by calculate_fp_hash method> is calculated using data returned by get_fp_hash_data method and the key provided by the EuPlatesc. It is the control key of the message.
*/

```
/**  
 * class CaptureResponse  
 * @author Stefan  
 *  
 * CaptureResponse structure is an array or class that extends WSResponse = that contain the response of the capture operation  
 * ep_id <string 40> is the transaction's unique ID receive from the payment system after a successful approval  
 * status <int> is the confirmation code  
 * message <string> is the confirmation string message  
 * fp_hmac, timestamp, nonce value are overwrite after each call of function addHMAC($key, true)  
 * timestamp <yyyymmddhiiss> <required> is the response timestamp set by the EuPlatesc system on sending this request. Filled on __constructor if is NULL.  
 * nonce hexadecimal string <filled on instance> is random generated on each call. Represents the salt for fp_hash calculation and is used to identify duplicate messages.  
 * fp_hash hexadecimal string <filled by calculate_fp_hash method> is calculated using data returned by get_fp_hash_data method and the key provided by the EuPlatesc. It is the control key of the message.  
 *  
 * status message  
 * 0 capture: capture pending OK  
 * 101 capture: invalid parameter ep_id: <ep_id>  
 * 102 capture: invalid parameter mid: <mid>  
 * 103 capture: invalid parameter curr: <curr>  
 * 104 capture: invalid parameter timestamp: <timestamp>  
 * 105 capture: invalid parameter amount (AMOUNT > 0): <amount>  
 * 301 capture: capture not possible, transaction isn't approved  
 * 302 capture: capture not possible, transaction expired  
 * 303 capture: capture not possible, transaction data didn't match  
 * 304 capture: transaction already captured  
 * 305 capture: transaction is reversal  
 * 306 capture: capture request already sent  
 * 307 capture: capture not permitted; actual status is <the_status>  
 * 308 capture: capture not permitted, reversal pending  
 * 401 capture: transaction not found  
 * 901 capture: internal error  
 * 999 capture: unknown error  
 */
```

```
/**  
 * class ReversalRequest  
 * @author Stefan  
 *  
 * ReversalRequest structure is an array or class that extends DORquest = the structure that represent transaction to be reversed.  
 * The structure is  
 * ep_id <string[40]> <required> is the transaction's unique ID receive from the payment system after a successful approval  
 */
```



```
* RefundRequest structure is an array or class that extends DOResult = the structure that represent transaction to be refunded.
* The RefundRequest message will flag the transaction to be processed offline by EuPlatesc team.
* The structure is
*   ep_id <string[40]> <required> is the transaction's unique ID receive from the payment system after a successful approval
*   mid <string[15]> <required> is the merchant terminal. This is defined on opening a merchant account on EuPlatesc system
*   amount <float[10.2]> <required> is the amount of the approved transaction.
*   curr <string[3]> <required> is the currency of the approved amount
*   reason <string> <required> the reason of refund
*   fp_hmac, timestamp, nonce value are overwrite after each call of function addHMAC($key, true)
*   timestamp <yyyymmddhhiiss> <required> is the request timestamp set by the merchant system on sending this request. Filled on __constructor if is NULL.
*   nonce   hexadecimal string <filled on instance> is random generated on each call. Represents the salt for fp_hash calculation and is used to identify duplicate
messages.
*   fp_hash   hexadecimal string <filled by calculate_fp_hash method> is calculated using data returned by get_fp_hash_data method and the key provided by the
EuPlatesc. It is the control key of the message.
*/
/**
* class RefundResponse
* @author Stefan
*
* RefundResponse structure is an array or class that extends WSResponse = that contain the response of the refund operation
*   ep_id <string 40> is the transaction's unique ID receives on capture operation call
*   status <int> is the confirmation code
*   message <string> is the confirmation string message
*   fp_hmac, timestamp, nonce value are overwrite after each call of function addHMAC($key, true)
*   timestamp <yyyymmddhhiiss> <required> is the timestamp of this message. Filled on __constructor if is NULL.
*   nonce   hexadecimal string <filled on instance> is random generated on each call. Represents the salt for fp_hash calculation and is used to identify duplicate
messages.
*   fp_hash   hexadecimal string <filled by calculate_fp_hash method> is calculated using data returned by get_fp_hash_data method and the key provided by the
EuPlatesc. It is the control key of the message.
*
*   status   message
*   0       refund: refund pending OK
*   101     refund: invalid parameter ep_id: <ep_id>
*   102     refund: invalid parameter mid: <mid>
*   103     refund: invalid parameter curr: <curr>
*   104     refund: invalid parameter timestamp: <timestamp>
*   105     refund: invalid parameter amount (AMOUNT > 0): <amount>
*   301     refund: refund not possible, transaction isn't approved
*   302     refund: refund not possible, after the permitted timeframe
*   303     refund: refund not possible, transaction data didn't match
*# 304     refund: refund not possible, transaction already captured
*   305     refund: transaction already is refunded/reversal
*   306     refund: refund request already sent
*   401     refund: transaction not found
*   901     refund: internal error
*   999     refund: unknown error
```

```
*/
/**
 * class PartialCaptureRequest
 * @author Stefan
 *
 * PartialCaptureRequest structure is an array or class that extends DOResponse = the structure that represent transaction to be partial captured.
 * The structure is
 *   ep_id <string[40]> <required> is the transaction's unique ID receive from the payment system after a successful approval
 *   mid <string[15]> <required> is the merchant terminal. This is defined on opening a merchant account on EuPlatesc system
 *   amount <float[10.2]> <required> is the original amount of the approved transaction.
 *   amount_to_refund <float[10.2]> <required> is the amount to be given back to the cardholder, float postive value between 0 and amount. 0 = no refund
 *   curr <string[3]> <required> is the currency of the approved amount
 *   delivery_awb <string> <required> if missing set it to NULL, is the delivery document (AWB or invoice number) used in case of chargeback
 *   fp_hmac, timestamp, nonce value are overwrite after each call of function addHMAC($key, true)
 *   timestamp <yyyymmddhhiiss> <required> is the request timestamp set by the merchant system on sending this request. Filled on __constructor if is NULL.
 *   nonce hexadecimal string <filled on instance> is random generated on each call. Represents the salt for fp_hash calculation and is used to identify duplicate
 * messages.
 *   fp_hash hexadecimal string <filled by calculate_fp_hash method> is calculated using data returned by get_fp_hash_data method and the key provided by the
 * EuPlatesc. It is the control key of the message.
 *
 * The captured transaction is (amount - amount_to_refund)
 * !! This operation is implemented as a partial refund
 *
 * For this type of operatin the merchant will be charged additionally as:
 * - assume that the transaction fee is fee%
 * - the charged fee = (amount + (amount - amount_to_refund)) * fee% = (2 * amount + amount_to_refund) * fee%
 */
/**
 * class PartialCaptureResponse
 * @author Stefan
 *
 * PartialCaptureResponse structure is an array or class that extends WSResponse = that contain the response of the partial captured operation
 *   ep_id <string 40> is the transaction's unique ID receives on capture operation call
 *   status <int> is the confirmation code
 *   message <string> is the confirmation string message
 *   fp_hmac, timestamp, nonce value are overwrite after each call of function addHMAC($key, true)
 *   timestamp <yyyymmddhhiiss> <required> is the response timestamp set by the EuPlatesc system on sending this request. Filled on __constructor if is NULL.
 *   nonce hexadecimal string <filled on instance> is random generated on each call. Represents the salt for fp_hash calculation and is used to identify duplicate
 * messages.
 *   fp_hash hexadecimal string <filled by calculate_fp_hash method> is calculated using data returned by get_fp_hash_data method and the key provided by the
 * EuPlatesc. It is the control key of the message.
 *
 * status message
 * 0 partial_capture: partial_capture pending OK
 * 101 partial_capture: invalid parameter ep_id: <ep_id>
 * 102 partial_capture: invalid parameter mid: <mid>
```

```
* 103 partial_capture: invalid parameter amount (AMOUNT > 0): <amount>
* 104 partial_capture: invalid parameter timestamp: <timestamp>
* 105 partial_capture: invalid parameter amount_to_refund [0; ORG_AMOUNT]: <amoun_to_refund>
* 106 partial_capture: invalid parameter curr: <curr>
* 301 partial_capture: reversal not possible, transaction isn't approved
* 302 partial_capture: reversal not possible, transaction expired
* 303 partial_capture: reversal not possible, transaction data didn't match
* 304 partial_capture: reversal not possible, transaction already captured
* 305 partial_capture: transaction already is reversal/partial_capture
* 306 partial_capture: partial_capture request already sent
* 401 partial_capture: transaction not found
* 901 partial_capture: internal error
* 999 partial_capture: unknown error
*/
```

```
/**
 * class RecurentRequest
 * RecurentRequest is a class that extends DORquest = used for call a recurrent payment.
 * call function addHMAC($key) before pass this structure to EuPlatesc WebServices in order to set the fp_hmac, timestamp, nonce value.
 * call function addHMAC($key) before each call EuPlatesc WebServices even the data is identical.
 *
 * fp_hmac, timestamp, nonce value are overwrite after each call of function addHMAC($key, true)
 * timestamp <yyyymmddhhiiss> <required> is the request timestamp set by the merchant system on sending this request. Filled on __constructor if is NULL.
 * nonce hexadecimal string <filled on instance> is random generated on each call. Represents the salt for fp_hash calculation and is used to identify duplicate
 messages.
 * fp_hash hexadecimal string <filled by calculate_fp_hash method> is calculated using data returned by get_fp_hash_data method and the key provided by the EuPlatesc.
 It is the control key of the message.
 *
 * @author Stefan
 *
 */
```

```
/**
 * class RecurentResponse
 * RecurentResponse is a class that extends StdClass used for call a recurrent payment.
 * This is the structure of the a Recurent EuPlatesc WebService call
 * @author Stefan
 *
 * ep_id <string 40> is the transaction's unique ID receives on capture operation call
 * status <int> is the confirmation code
 * message <string> is the confirmation string message
 * fp_hmac, timestamp, nonce value are overwrite after each call of function addHMAC($key, true)
 * timestamp <yyyymmddhhiiss> <required> is the response timestamp set by the EuPlatesc system on sending this request. Filled on __constructor if is NULL.
 * nonce hexadecimal string <filled on instance> is random generated on each call. Represents the salt for fp_hash calculation and is used to identify duplicate
 messages.
 * fp_hash hexadecimal string <filled by calculate_fp_hash method> is calculated using data returned by get_fp_hash_data method and the key provided by the
 EuPlatesc. It is the control key of the message.
 *
 */
```

```
* $status message
* 0 recurent: recurent pending OK
* 101 recurent: invalid parameter ep_id: <ep_id>
* 102 recurent: invalid parameter mid: <mid>
* 103 recurent: invalid parameter amount (AMOUNT > 0): <amount>
* 104 recurent: invalid parameter curr: <curr>
* 105 recurent: invalid parameter timestamp: <timestamp>
* 106 recurent: invalid parameter oredr_desc: <oredr_desc>
* 107 recurent: invalid parameter recurent: <recurent>
* 108 recurent: invalid parameter email: <email>
* 109 recurent: invalid parameter invoice_id: <invoice_id>
* 301
* 302
* 303
* 304
* 305
* 306
* 401
* 901
* 999
*/
```

```
/**
 * class StatusRequest
 * Ask about a transaction status
 *
 * @author stefan
 *
 * ep_id, is the transaction's unique ID receive from the payment system after a successful approval
 * mid, Merchant ID
 * what, optional - what status you what to get, actual only EPS_GET_ALL_STATUS is implemented
 * timestamp <yyyymmddhiiss> <required> is the request timestamp set by the merchant system on sending this request. Filled on __constructor if is NULL.
 * nonce hexadecimal string <filled on instance> is random generated on each call. Represents the salt for fp_hash calculation and is used to identify duplicate
 messages.
 * fp_hash hexadecimal string <filled by calculate_fp_hash method> is calculated using data returned by get_fp_hash_data method and the key provided by the EuPlatesc.
 It is the control key of the message.
 */
```

```
/**
 * class StatusResponse
 * provide the transaction status details
 *
 * @author stefan
 *
 * transaction_status // Transaction status as integer
 * transaction_status_msg // Transaction status as text
 * approval // The approval code of. Is set only if the transaction is apprved
 * rrn // The RRN code of the tranzaction
```

```
* cbk // If the transaction has charge back
* approval_date // the date of the approval request
* capture_date // date when the transaction was captured, if is captured
* refund_date // date when the transaction was refunded, if is refunded
* original_value // the approval value
* curr // currency
* refunded_value // the amount that was refund (<= approval value)
* effective_captured_value // the net amount (original_value - refunded_value)
* installments // number of approved installments
* installment_bank // instalemnts on bank
* discount_amount // if the transactuona has discount
* transaction_channel // transaction channel: CARD, SMS, OP
* transaction_type // recurring flag
* settlement // the EuPlatesc invoice
* what // the what value receive in StatusRequest message
* fee // unused
* ep_id // string, is the transaction's unique ID receive from the payment system after a successful approval
* mid // string Merchant ID (terminal ID)
* status // int, error code 0 => OK != 0 => an error occurred
* message // string, error code message
* timestamp // <yyyymmddhhiiss> <required> is the response timestamp set by the EuPlatesc system on sending this request. Filled on __constructor if is
NULL.
* nonce // hexadecimal string <filled on instance> is random generated on each call. Represents the salt for fp_hash calculation and is used to
identify duplicate messages.
* fp_hash // hexadecimal string <filled by calculate_fp_hash method> is calculated using data returned by get_fp_hash_data method and the key provided by
the EuPlatesc. It is the control key of the message.
*/
/**
* class getRecurringPaymentListRequest
*
* @author Stefan
*
* getRecurringPaymentListRequest structure is an array or class that extends EuPlatescMessage = the structure that represent the mesage request.
* The structure is
* ep_id <string[40]> <required> is the transaction's unique ID receive from the payment system after a successful approval
* mid <string[15]> <required> is the merchant terminal. This is defined on opening a merchant account on EuPlatesc system
* fp_hmac, timestamp, nonce value are overwrite after each call of function addHMAC($key, true)
* timestamp <yyyymmddhhiiss> <required> is the request timestamp set by the merchant system on sending this request. Filled on __constructor if is NULL.
* nonce hexadecimal string <filled on instance> is random generated on each call. Represents the salt for fp_hash calculation and is used to identify duplicate
messages.
* fp_hash hexadecimal string <filled by calculate_fp_hash method> is calculated using data returned by get_fp_hash_data method and the key provided by the EuPlatesc.
It is the control key of the message.
*/
/**
* class getRecurringPaymentListResponse
* getRecurringPaymentListResponse is a class that extends StdClass used for get a list of last 10 recurrent payment.
```

```
* This is the structure of the a Recurent EuPlatesc Webservice call
* @author Stefan
*
*     ep_id <string 40> is the transaction's unique ID receives on capture operation call
*     status <int> is the confirmation code
*     message <string> is the confirmation string message
*     recurring_lits <array of RecurringPaymentList>
*     fp_hmac, timestamp, nonce value are overwrite after each call of function addHMAC($key, true)
*     timestamp <yyyymmddhhiiss> <required> is the response timestamp set by the EuPlatesc system on sending this request. Filled on __constructor if is NULL.
*     nonce hexadecimal string <filled on instance> is random generated on each call. Represents the salt for fp_hash calculation and is used to identify duplicate
messages.
*     fp_hash hexadecimal string <filled by calculate_fp_hash method> is calculated using data returned by get_fp_hash_data method and the key provided by the
EuPlatesc. It is the control key of the message.
*
* $status message
* 0 getRecurringPaymentListResponse: recurent pending OK
* 101 getRecurringPaymentListResponse: invalid parameter ep_id: <ep_id>
* 102 getRecurringPaymentListResponse: invalid parameter mid: <mid>
* 105 getRecurringPaymentListResponse: invalid parameter timestamp: <timestamp>
* 301
* 302
* 303
* 304
* 305
* 306
* 401
* 901
* 902 getRecurringPaymentListResponse: invalid parameter RecurringLits [must be an array of RecurringPaymentList].
* 999
*/
/**
 * class EuPlatescMessage
 *
 * implements the basic functions of the messages that is used to communicate with EuPlatesc (www.euplatesc.ro) webservices
 *
 * call function addHMAC($key) before pass this structure to EuPlatesc WebServices in order to set the fp_hmac, timestamp, nonce value.
 * call function addHMAC($key) before each call EuPlatesc WebServices even the data is identical.
 *
 * @author stefan
 *
 * The structure is
 * fp_hmac, timestamp, nonce value are overwrite after each call of function addHMAC($key, true)
 * timestamp <yyyymmddhhiiss> <required> is the request/response timestamp set by the merchant/EuPlatesc system on sending this request. Filled on __constructor if is
NULL.
 * nonce hexadecimal string <filled on instance> is random generated on each call. Represents the salt for fp_hash calculation and is used to identify duplicate
messages.
```

```
*      fp_hash  hexadecimal string <filled by calculate_fp_hash method> is calculated using data returned by get_fp_hash_data method and the key provided by the
EuPlatesc. It is the control key of the message.
*
*/
/* class DORquest
*
* implements the base class for all Webservice requests (www.euplatesc.ro)
*
* @author Stefan
*
*      ep_id <string[40]> <required> is the transaction's unique ID receive from the payment system after a successful approval
*      mid <string[15]> <required> is the merchant terminal. This is defined on opening a merchant account on EuPlatesc system
*      amount <float[10.2]> <required> is the amount of the approved transaction.
*      curr <string[3]> <required> is the currency of the approved amount
*      fp_hmac, timestamp, nonce value are overwrite after each call of function addHMAC($key, true)
*      timestamp <yyyymmddhhiiss> <required> is the request timestamp set by the merchant system on sending this request. Filled on __constructor if is NULL.
*      nonce  hexadecimal string <filled on instance> is random generated on each call. Represents the salt for fp_hash calculation and is used to identify duplicate
messages.
*      fp_hash  hexadecimal string <filled by calculate_fp_hash method> is calculated using data returned by get_fp_hash_data method and the key provided by the
EuPlatesc. It is the control key of the message.
*/
/**
* class WSResponse
* implements the base class for all Webservice response (www.euplatesc.ro)
*
* @author Stefan
*
*      ep_id <string[40]> <required> is the transaction's unique ID receive from the payment system after a successful approval
*      mid <string[15]> <required> is the merchant terminal. This is defined on opening a merchant account on EuPlatesc system
*      status <int> <required> the error code. 0 == OK (no error), != 0 == ERROR (an error occurred)
*      message <string> <required> the error description
*      fp_hmac, timestamp, nonce value are overwrite after each call of function addHMAC($key, true)
*      timestamp <yyyymmddhhiiss> <required> is the response timestamp set by the EuPlatesc system on sending this request. Filled on __constructor if is NULL.
*      nonce  hexadecimal string <filled on instance> is random generated on each call. Represents the salt for fp_hash calculation and is used to identify duplicate
messages.
*      fp_hash  hexadecimal string <filled by calculate_fp_hash method> is calculated using data returned by get_fp_hash_data method and the key provided by the
EuPlatesc. It is the control key of the message.
*/
```