



VPOS Integration manual

1. Introduction

iCheckOutX is a WEB front-end service which simplifies web-shop integration with iPayGate by utilizing web technology and standard functionalities.

iCheckOutX provides:

- Standardized interface with iPayGate
- 3D-Secure 2.2 integrated support
- Administration interface for review, completion and cancellation of orders (transactions)
- Integrated into wider security infrastructure
- Effective appearance integration into web site design
- Simplified integration of web shops with iPayGate system and eCommerce processing infrastructure. Serves customer with payment details page
- Web shop only implements shopping basket

1.1. Supported transactions types

iCheckOutX solution supports multiple transaction types and payment channels.

Supported transaction types include:

- Purchase
- Pre-authorization and Completion
- Refund
- Reversal / Cancellation - for transactions made on current business day.
- One click purchase - using the stored Cardholder values.

1.2. iCheckOutX System Architecture

The architecture of typical high-availability iCheckOutX / Internet Payment Gateway (IPGW) solution is described in the following image. The functionalities provided by the system are divided into categories:

- iCheckOutX front-end
- iCheckOutAdminX GUI (for merchant and transaction administration) integrated into the Merchant portal ('Kereskedői portál')
- Internet Payment Gateway backend

iCheckOutX front-end

Request from web shops (V-POS) are processed by the frontend - iCheckOutX system. iCheckOutX system validates all incoming requests, verifies the received web shop information, checks if merchant is active and allowed to perform the specific transaction type. If request validation is successful, iCheckOutX continues with



3D secure validation. The 3D secure validation is done utilizing the 3D Server that communicates with Directory server of the networks.

If required, iCheckOutX will initiate additional security validations as defined in 3D Secure standard. If cardholder authentication is required, iCheckOutX will redirect the cardholder to issuer API to enter the required information. After these steps are done, the transaction is authorized through the IPGW system.

Internet Payment Gateway – IPGW

Internet payment gateway solution is based on server solution that is configured to support card-not-present transaction types. The IPGW solution supports different transactions types: **purchase, preauthorization and completion, refund, reversal.**

IPGW can be used to securely store cardholder data that can be used for initiating recurring transactions or enabling cardholders to initiate payment without (re)entering card information (One-click payment).

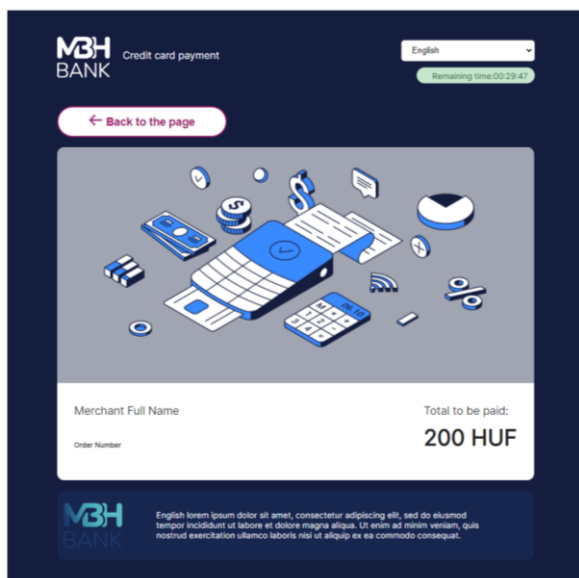
Sensitive cardholder data on IPGW are stored encrypted with key protected inside Hardware Security Module (HSM) device. IPGW supports IBM CCA 4767 and Tales 9000 and 10K HSM devices.

1.3. Integration procedure

1. Registration of the merchant in the banking systems using the information provided by the merchant.
2. Providing access to the Merchant Portal ('Kereskedői portál') system in the sandbox:
Secret key must be exchanged between iCheckoutX and Merchant, which is used to sign all requests between merchant and iCheckoutX system.
The Bank will send a registration letter to the email address of the user provided by the Merchant, by clicking on the link in the letter, once registered, the user will have access to the test secret key in the Merchant Portal ('Kereskedői portál'), which the user can retrieve as indicated in Annex V.
3. The Merchant will receive a welcome letter containing the contact details of the documentation required for the integration, as well as the Merchant and Terminal details in the sandbox, as the Merchant and Terminal ID must be included in the transaction request.
4. Webshop development: The technical steps of the integration are described here in chapter 2 (page 4).
5. Testing: once the sandbox is ready and the transactions can be successfully initiated, the tester must execute the tests in the test case package and verify their success in cooperation with the bank's service provider. The test cases to be executed and the parameters to verify their success are set out in Annex II.
6. Provide access to banking systems and merchant data in a production environment:
After the completion of the testing (all relevant test cases are checked by the bank's service provider and feedback is provided to the Merchant) and the verification of the Merchant's website/webshop, the bank will provide access to the Merchant's specified users in the Merchant Portal (production environment), the process is the same as described in point 2.
Access to the Security Key and Merchant/Terminal data required for go live is provided by the bank in the Merchant Portal (production environment), as the prerequisite for go live, for the first successful transaction request to be accepted, is that the transaction request contains the Security Key and Merchant/Terminal data valid in the production environment.

1.4. iCheckoutX Message Flow - Transaction flow

The iCheckOutX transaction flow begins with merchant web shop sending the initial purchase request described in integration chapter. After validating the request, iCheckOutX will display the landing page for Cardholder data input. The design of the landing page can be customized for each merchant by defining merchants CSS. If no Merchant CSS is provided then default landing page will be displayed. On the payment landing page, the Customer can enter the Cardholder information (Name, Surname, Address...) than the Card data (PAN, CVV, expiry date).



Customer information

Please enter your personal information

Last name *

First name *

Postal code *

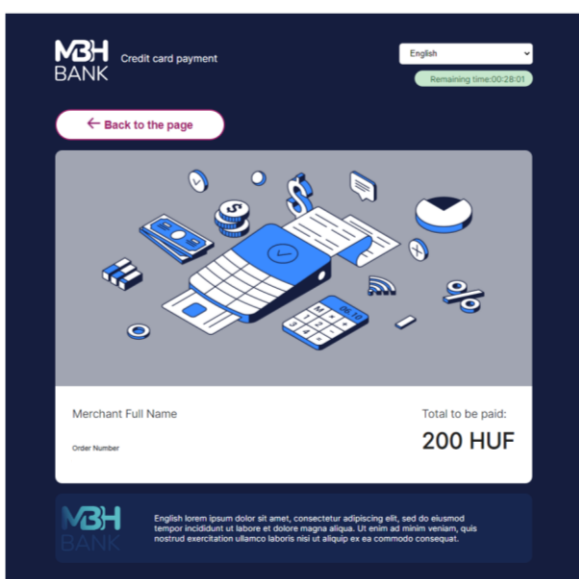
City *

Country *

Address (Name, type, and number of public place, floor, door *

Email address *

[Continue](#)



Payment Data

Please enter your card information

Credit card number *

Expiry Month *

Expiry Year *

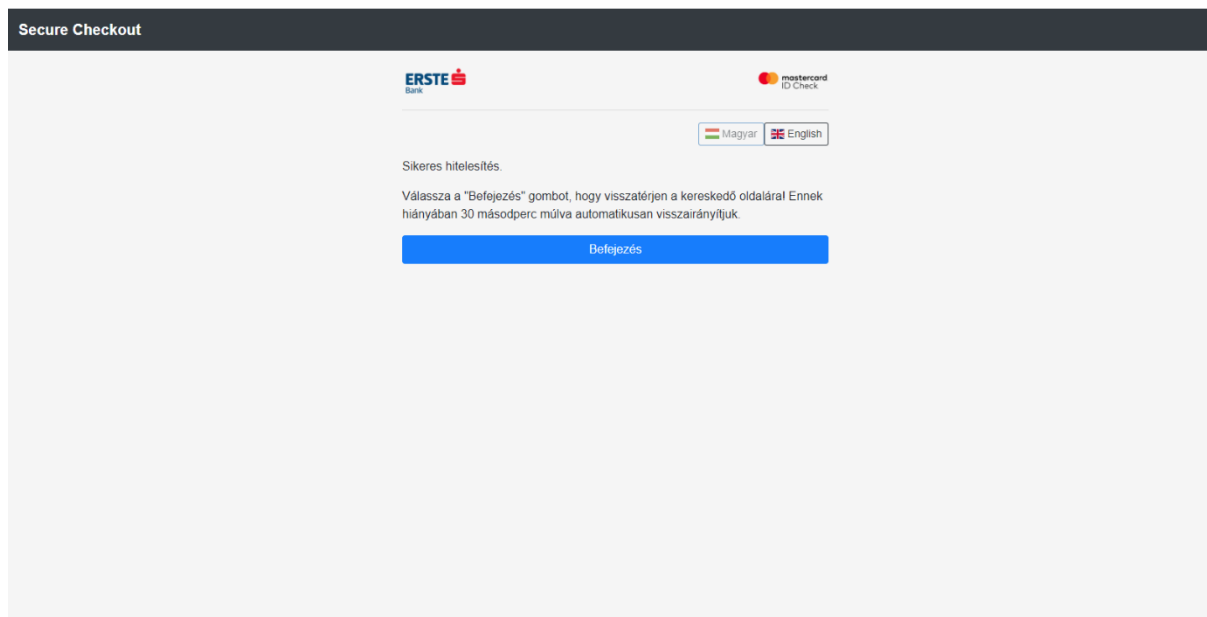
CVC code *

[Previous step](#) [Pay - 200.00 HUF](#)

Payment Landing page

After the cardholder data is entered, iCheckOutX will automatically perform 3D Secure validation for the card. If no 3D Secure validation is needed for cards (card is not enrolled in 3DS or is marked as frictionless) iCheckOutX will continue to Authorization.

If Cardholder Authentication is required iCheckOutX will display Issuers page and wait for customer to continue the authentication process.



After the 3D Secure process is done the transaction is authorized through IPGW system with appropriate acquirer. This process is transparent to the Merchant web shop.

In the final step the iCheckOutX redirects customer back to the original Web shop URL with the transaction result. If the transaction is successful (is Approved) authorization code and other data is returned to the merchants Web shop. Response message details are defined [here](#).

The merchant must display the result and details of the transaction. The bank may also send a confirmation message (by email) to the merchant and the cardholder on the result and details of the transaction, but this service does not exempt the merchant from displaying the requested information on its website.

2. Integration with iCheckOutX system

This chapter describes all integration details. It defines message types and data elements for all iCheckOutX messages.

To simplify the integration process, a predefined HTML template is provided that can be used to test the iCheckOutX integration. The scripts assumes the role of a Web shop and can be used to initiate different transaction requests towards iCheckOutX.

Integration script can be downloaded from this [link](#).

iCheckOutX service integration with Internet shop system is done through HTML POST method. POST data sent in form must be UTF-8 encoded. Example:

```
<form enctype="application/x-www-form-urlencoded; charset=UTF-8">
```

2.1. iCheckOutX Messages

This section defines the messages defined for iCheckOutX system. Message exchange between merchant shop and iCheckOutX service is done using HTTP protocol. Web shop prepares HTML form which is sent using HTTP POST to iCheckOutX endpoint. An HTTP POST request should be sent to URL:

<https://securepay.mbhbank.hu/iCheckOutX/v1/checkout/confirm.xhtml>

The URL address used for testing is given in Annex II.

2.2. iCheckOutX Request Parameters

Parameter Name	Parameter Description	Value Format	Value Length
account_id	One-Click Payments Account ID	AN	36
additional_compl_data1	Additional Completion Data	AN	36
card_cvd	Payment Card CVV	N	4
card_expdate	Payment Card Expiration Date	N	4
card_pan	Payment Card Number	N	20
customer_address	Customer Street	AN	200
customer_city	Customer City	AN	50
customer_country	Customer Country	AN	30
customer_email	Customer Email	AN	30
customer_lang	Customer Language	AN	2
customer_name	Customer First Name	AN	50
customer_phone	Customer Phone	AN	30
customer_surname	Customer Last Name	AN	50

customer_zip	Customer ZIP code	AN	8
discounted_amount	Unused field	AN	13
discounted_card_types	Unused field	AN	200
discounted_max_installments	Unused field	N	2
fixed_card_type	Fixed (unchangeable) card type	AN	10
fixed_installments	Fixed (unchangeable) number of installments	AN	5
merchant_approve_url	Merchant Dynamic Approve URL	AN	128
merchant_decline_url	Merchant Dynamic Decline URL	AN	128
merchant_id	Merchant ID (MID)	AN	16
order_delivery_date	Order Delivery Date	AN	11
order_number	Unique Order Number	AN	50
payment_method	Payment method that will be used. If not present, all methods will be allowed.	AN	16
purchase_amount	Purchase Amount	AN	13
purchase_currency	Purchase Currency	N	3
purchase_description	Purchase Description (for example: 'Test transaction')	AN	200
purchase_installments	Purchase Installments	N	2
recurring_order_number	Original Order Number for Recurring Payments	AN	50
request_hash	Request Hash	AN	128
request_type	Request Type	AN	15

terminal_id	Terminal ID (TID)	AN	32
trantype	Transaction Type	AN	20

Definition of Request Parameters Presence

Presence is defined as:

- M, mandatory
- O, optional
- C, contidional (i.e. account_id must be present fore One-Click transactions)
- "-", not relevant

Parameter Name	Preauthorization	Authorization	Completion	Reversal	Refund
account_id	C	C	-	-	-
additional_compl_data1	-	-	O	-	-
card_cvd	O	O	-	-	-
card_expdate	O	O	-	-	-
card_pan	O	O	-	-	-
customer_address*	O*	O*	-	-	-
customer_city*	O*	O*	-	-	-
customer_country*	O*	O*	-	-	-
customer_email*	O*	O*	-	-	-

Parameter Name	Preauthorization	Authorization	Completion	Reversal	Refund
customer_lang	O	O	-	-	-
customer_name*	O*	O*	-	-	-
customer_phone*	O*	O*	-	-	-
customer_surname*	O*	O*	-	-	-
customer_zip*	O*	O*	-	-	-
discounted_amount	O	O	-	-	-
discounted_card_types	O	O	-	-	-
discounted_max_installments	O	O	-	-	-
fixed_card_type	O	O	-	-	-
fixed_installments	O	O	-	-	-
merchant_approve_url	O	O	O	O	O
merchant_decline_url	O	O	O	O	O
merchant_id	M	M	M	M	M
order_delivery_date	O	O	-	-	-

Parameter Name	Preauthorization	Authorization	Completion	Reversal	Refund
order_number	M	M	M	M	M
payment_method	O	O	-	-	-
purchase_amount	M	M	M	M	M
purchase_currency	M	M	M	M	M
purchase_description	O	O	-	-	-
purchase_installments	O	O	-	-	-
recurring_order_number	C	C	-	-	-
request_hash	M	M	M	M	M
request_type	transaction	transaction	completion	reversal	refund
terminal_id	M	M	M	M	M
trantype	preauth	auth	completion	reversal	refund

Details on parameter presence for One-Click Payments are listed in [One-Click Payments chapter](#).

*The Bank collect the indicated data for each transaction, and it is therefore recommended that these data are collected and transmitted so that the cardholder does not have to enter them on the payment landing page. The provision of this information on the payment landing page is therefore mandatory and the payment experience is enhanced if the information is submitted in the payment request, so 'pre-filling' these fields.

At the merchant's request, it is also possible to hide these fields on the payment landing page, if the data fields are always filled in, in which case the cardholder will not be able to modify these fields on the payment landing page.

2.3. iCheckOutX Response Parameters

Parameter Name	Parameter Description	Value Format	Value Length
account_id	One-Click Payments Account ID	AN	36
acquirer	Order Acquirer	AN	16
card_expdate	Payment Card Expiration Date	N	4
card_type	Payment Card Type	AN	64
customer_address	Customer Street	AN	200
customer_city	Customer City	AN	50
customer_country	Customer Country	AN	30
customer_email	Customer Email	AN	30
customer_lang	Customer Language	AN	2
customer_name	Customer First Name	AN	50
customer_surname	Customer Last Name	AN	50
customer_tel	Customer Phone	AN	30
customer_zip	Customer ZIP code	AN	8
discounted_amount	Discounted Amount	AN	13
masked_pan	Masked Payment Card Number	AN	20
merchant_id	Merchant ID (MID)	AN	16

order_date	Order Date	AN	11
order_delivery_date	Order Delivery Date	AN	11
order_number	Unique Order Number	AN	50
payment_method	Payment method that will be used. If not present, all methods will be allowed.	AN	16
purchase_amount	Purchase Amount	AN	13
purchase_currency	Purchase Currency	N	3
purchase_description	Purchase Description (for example: 'Test transaction')	AN	200
purchase_installments	Purchase Installments	N	2
request_type	Request Type	AN	15
response_appcode	Authorization Approval Code	AN	6
response_hash	Response Hash	AN	128
response_message	Authorization Response Message	AN	200
response_result	Response Result	N	3
response_systan	Transaction System Trace Audit Number	AN	6
transaction_type	Transaction Type	AN	20

2.4. Parameter details

2.4.1. request_type - A kérelem típusa

Type: **alphanumeric** Mandatory: **Yes**

The request_type field defines the type of request sent to iCheckOutX system. The type can one of the following values:

Request type	Description
transaction	Indicates a request for financial transaction. Check trantype field for additional details.
register	In case of a payment with saved card data (one-click transaction), the operation that initiates the saving of card data and the creation of a unique identifier.
update	In case of payment with saved card data (one-click transaction), an operation to update/modify data.
get	In case of payment with saved card data (one-click transaction), retrieval of card data to the webshop based on a unique identifier, the card number always appears as masked data.
use	In case of a payment with saved card details (one-click transaction), initiating a transaction with a unique identifier.
delete	In case of a payment with saved card details (one-click transaction), delete unique identifier.
recurring	Indicates a request for an automated financial transaction.
checkstatus	Used for checking the status of an order previously processed by the system

2.4.2. trantype - Transaction Type

Type: **alphanumeric** Mandatory: **Yes**

The trantype field defines the type of type of Financial transaction request sent to iCheckOutX system.

Valid values	Description
auth	Purchase request. Transaction is done in one step, no completion is required.
preauth	Preauthorization request

Valid values	Description
completion	Completion of previous Preauthorization request. Preauth must be Approved.
reversal	Transaction cancellation request for transactions that have not yet been settled (successfully normal transaction on current business day or uncompletion pre-authorisation)
refund	Refund of previously done Purchase or Completion transactions. Max amount limited to the amount of original Purchase or Completion.

2.4.3 merch_id – Merchant ID (MID)

Type: **alphanumeric** Mandatory: **Yes**

The merch_id field indicates the merchant ID (MID), which is defined by the Bank.

It should be paired with [terminal_id](#) field to uniquely identify the merchant and terminal from which the transaction request originates.

2.4.4. terminal_id - Terminal ID (TID)

Type: **alphanumeric** Mandatory: **Yes**

The terminal_id field indicates the terminal ID (TID), which to be defined and transferred by the Bank during integration.

It should be paired with [merch_id](#) field to uniquely identify the merchant and terminal from which the transaction request originates.

2.4.5. order_number – Order Number

Type: **alphanumeric** Mandatory: **Yes**

The order_number field indicates the unique order identifier, which is defined/generated by the merchant.

Merchant should make sure the order_number value is unique for the originating terminal_id.

2.4.6. merchant_approve_url / merchant_decline_url – Merchant Dynamic Callback URLs for Approved and Declined Transactions

Type: **alphanumeric** Mandatory: **No**

The merchant_approve_url and merchant_decline_url parameters can be used in a transaction request to define on which URL the webshop expects the final response.

These URLs are commonly set permanently under merchant settings in iCheckOutX Admin GUI and do not have to be sent with each request. Using `merchant_approve_url` and `merchant_decline_url` in a transaction request overrides the default merchant settings for that transaction.

Examples:

```
merchant_approve_url=https://host-domain.com/iCheckOutX/v1/testMerchant/approved
merchant_decline_url=https://host-domain.com/iCheckOutX/v1/testMerchant/declined
```

2.4.7. `purchase_installments` - Purchase Installments

Type: **numeric** Mandatory: **No**

A field used to split order payment into installments.

Can be either 0 or ≥ 2 , while the maximum value is defined by merchant transaction rules. Merchant can also enforce different installment rules depending on the purchase amount.

Value of 0 indicates a one-time payment.

In a merchant response, this field represents the final number of installments chosen by the customer.

2.4.8. `recurring_order_number` - Recurring Order Number

Type: **alphanumeric** Mandatory: **Yes** (only in cases of subsequent 'recurring' requests)

A field used for recurring payments, `recurring_order_number` indicates the original order to which the new order is referred.

2.4.9. `payment_method` - Payment Method

Type: **alphanumeric** Mandatory: **No**

When `payment_method` is sent as a request parameter, it defines a single payment method that is allowed by merchant and will be used for that particular payment.

Possible values are: card (Credit Card Payment)

2.4.10. `request_hash` - Request security hash

Type: **alphanumeric** Mandatory: **Yes**

Request hash is calculated using the SHA512 algorithm and the string that is used for SHA512 encryption is constructed as follows:

1. Convert all **parameter names** to **lowercase**. Example: "*Customer_Name=John*" is translated to lowercase "*customer_name=John*" The value (in this example John) is not converted.
2. Sort all parameters alphabetically.
3. In hash calculation input only include the parameters with non empty values.
4. Concatenate all parameters separated with "&". Example:
account_id=123&customer_address=Customer Address&customer_city=New York...
5. Add [secret key=XXXXXX](#) as the last parameter in hash input. (replace XXXXX with unique secret key assigned to merchants).
6. Calculate SHA512 hash for constructed input. Input to SHA512 function **must be encoded as UTF-8**
7. Use the calculated hash as **request_hash** parameter in request.

Possible parameters used for request_hash calculation are:

```
account_id
additional_compl_data1
card_cvd
card_expdate
card_pan
customer_address
customer_city
customer_country
customer_email
customer_lang
customer_name
customer_phone
customer_surname
customer_zip
discounted_amount
discounted_card_types
discounted_max_installments
fixed_card_type
fixed_installments
merchant_approve_url
merchant_decline_url
merchant_id
order_delivery_date
order_number
payment_method
purchase_amount
purchase_currency
purchase_description
purchase_differperiod
purchase_installments
request_hash
request_type
terminal_id
trantype
```

The delimiter that is used to separate the parameters is '&'.

Example of hash calculation:

Input parameters:

```
merchant_id=5656565656
```

```

purchase_amount=1.00
purchase_currency=191
order_number=1ORD_120208_19
request_type=transaction
trantype=auth
submit_type=manual
purchase_installments=0
purchase_description=Test transaction
customer_name=John
customer_surname=Smith
customer_address=498 Gainsway Lane
customer_country=United States
customer_city=New York
customer_zip=10040
account_id=

```

Sorted and filtered list (removed empty account_id). Added secret_key=SecretKey123 at the end.

```

customer_dress=18 Gainsway Lane
customer_city=New York
customer_country=United States
customer_name=John
customer_surname=Smith
customer_zip=10040
merchant_id=5656565656
order_number=1ORD_120208_19
purchase_amount=1.00
purchase_currency=191
purchase_description=Test transaction
purchase_installments=0
request_type=transaction
submit_type=manual
trantype=auth
secret_key=SecretKey123

```

Calculate SHA

```

SHA512(customer_address=498 Gainsway Lane&customer_city=New
York&customer_country=United
States&customer_name=John&customer_surname=Smith&customer_zip=10040&merchant_id=5656565656&
order_number=1ORD_120208_19&purchase_amount=1.00&purchase_currency=191&purchase_description=T
est
transaction&purchase_installments=0&request_type=transaction&submit_type=manual&trantype=auth&secr
et_key=SecretKey123)

```



```
=
a87a67e635e726e591e01cab3e11b4c3ba4522fa4e4cad05124085df9ce7169bd59c4ee12759bd1d7a7d206cb7e
b58b2784690647f0d5b0c30a00019eb7107ec
```

Request to iCheckOutX must have:

```
request_hash=a87a67e635e726e591e01cab3e11b4c3ba4522fa4e4cad05124085df9ce7169bd59c4ee12759bd1
d7a7d206cb7eb58b2784690647f0d5b0c30a00019eb7107ec
```

2.4.11. response_hash - Response security hash

Type: **alphanumeric** Mandatory: **Yes**

Response hash is calculated using the SHA512 algorithm and the string that is used for SHA512 encryption is constructed as follows:

1. Convert all received **parameter names** to **lowercase**. Example: “*Customer_Name=John*” is translated to lowercase “*customer_name=John*” The value (in this example John) is not converted.
2. Omit *response_hash* parameter from hash calculation.
3. Sort all parameters alphabetically.
4. In hash calculation input only include the parameters with non-empty values.
5. Concatenate all parameters separated with “&”. Example: *acquirer=123&customer_email=example@mail.com&customer_lang=hr...*
6. Add [secret_key=XXXXXX](#) as the last parameter in hash input. (replace XXXXX with unique secret key assigned to merchants).
7. Calculate SHA512 hash for constructed input. Input to SHA512 function **must be encoded as UTF-8**
8. Check if calculated hash matches the **response_hash** parameter value in response

Possible parameters used for response_hash calculation are:

```
account_id
acquirer
card_expdate
card_type
customer_address
customer_city
customer_country
customer_email
customer_lang
customer_name
customer_surname
customer_tel
customer_zip
```

```
discount_amount  
masked_pan  
merchant_id  
oneclick_result  
order_date  
order_number  
payment_method  
purchase_amount  
purchase_currency  
purchase_description  
purchase_installments  
request_type  
response_appcode  
response_hash  
response_message  
response_result  
response_systan  
transaction_id  
transaction_type
```

The delimiter that is used to separate the parameters is '&'.

Example of hash calculation:

Merchant received parameters:

```
acquirer=123  
card_type=Visa  
customer_address=Test Address  
customer_city=Test City1  
customer_country=Hrvatska  
customer_email=test@email.com  
customer_lang=hr  
customer_name=TestName  
customer_surname=TestSurname  
customer_tel=+38512345678  
customer_zip=10000  
order_date=2021-09-14 11:54:10.004  
order_number=1ORD_121149_1567  
payment_method=  
purchase_amount=15.00  
purchase_currency=191  
purchase_description=Test order  
purchase_installments=0  
request_type=transaction
```

```
response_appcode=781722
response_hash=7c434756a1fc689f651334ed88603c9c8664b384bacdab82265e2f0cc9827309b184eb200cc0315
5977c0590a655b90f0b3594cf9edcbfc19c500f05dd1fe06f
response_message=APPROVED
response_result=00
response_systan=002977
transaction_id=
transaction_type=auth
account_id=
oneclick_result=
```

Sorted and filtered list (removed empty account_id). Added secret_key parameter at the end.

```

acquirer=123
card_type=Visa
customer_address=Test Address
customer_city=Test City1
customer_country=Hrvatska
customer_email=test@email.com
customer_lang=hr
customer_name=TestName
customer_surname=TestSurname
customer_tel=+38512345678
customer_zip=10000
discount_amount=0.00
order_date=2021-09-14 11:54:10.004
order_number=1ORD_121149_1567
purchase_amount=15.00
purchase_currency=191
purchase_description=Test order
purchase_installments=0
request_type=transaction
response_appcode=781722
response_message=APPROVED
response_result=00
response_systan=002977
transaction_type=auth
secret_key=XXX

```

calculate SHA

```

SHA512(acquirer=123&card_type=Visa&customer_address=Test Address&customer_city=Test
City1&customer_country=Hrvatska&customer_email=test@email.com&customer_lang=hr&customer_name=T
estName&customer_surname=TestSurname&customer_tel=+38512345678&customer_zip=10000&discount_a
mount=0.00&order_date=2021-09-14
11:54:10.004&order_number=1ORD_121149_1567&purchase_amount=15.00&purchase_currency=191&purch
ase_description=Test
order&purchase_installments=0&request_type=transaction&response_appcode=781722&response_message=
APPROVED&response_result=00&response_systan=002977&transaction_type=auth&secret_key=???)
=
7c434756a1fc689f651334ed88603c9c8664b384bacdab82265e2f0cc9827309b184eb200cc03155977c0590a655
b90f0b3594cf9edcbfc19c500f05dd1fe06f

```

Response hash value from iCheckOutX must be:

```
response_hash=7c434756a1fc689f651334ed88603c9c8664b384bacdab82265e2f0cc9827309b184eb200cc03155977c0590a655b90f0b3594cf9edcbfc19c500f05dd1fe06f
```

2.4.12. response_result – Response Result

Type: **alphanumeric**

Field contains response code received from the authorization host. Required only in responses. Response code is the only value that determines whether the request is approved or declined.

Response code	Response code description
000	Approved/Accepted
100	Declined
101	Expired Card
104	Restricted Card
109	Invalid merchant
111	Card not on file
115	Requested function not supported
121	Insufficient funds
400	Reversal accepted
909	Technical error – unable to process request
912	Host link down/ Server not available
930	Transaction not found
931	Transaction voided/reversed

The value of the 'response result' field should be used to provide the customer with a provide to the transaction result in the webshop:

Successful transaction message if 'response result' is '000', for all other values the transaction message is 'Unsuccessful transaction'.

2.4.13. response_message - Response Message

Type: **alphanumeric** Mandatory: **Yes**

The response_message field contains details about order status on the payment system.

The information provided in the 'response_message' field can be used to inform the merchant, the direct display of this message in the webshop is not recommended (in point 2.4.12)

2.4.14. Merchant notification (server-to-server)

Aside from the final response that merchants receive on customer redirect to webshop, there is an additional possibility to receive that same response on a dedicated merchant's endpoint as soon as the transaction has been approved or declined. This can be useful to reduce the potential uncertainty of the transaction status in cases when customers close their web-browser session before redirecting, or lose internet connection etc.

Parameters (as well as response hash) in s2s notification will have the same values as the customer redirect a few moments later.

Parameters will be sent via HTTP POST request in a JSON format.

To verify the notification has been received successfully by the merchant, iCheckOutX should receive HTTP 200 with JSON parameter status=OK. In all other cases, iCheckOutX will try to send the same notification until reaching out maximum number of tries or receiving status=OK.

Merchant notification setting can be triggered using the iCheckOutAdminX interface, which is also where the endpoint URL will be set.

If a webshop does not have a designated s2s response endpoint, it can implement a similar status checking method using [checkstatus requests](#).

2.5.1 Purchase Request

Payment initiation request: **trantype=auth**

```
merchant_id=DEMOMID
purchase_amount=15.00
purchase_currency=191
order_number=1ORD_1211310_4926
request_type=transaction
trantype=auth
purchase_description=Test+order
customer_name=TestName
customer_surname=TestSurname
customer_address=Test+Address
customer_country=Hrvatska
customer_city=Test+City1
customer_zip=10000
customer_lang=hr
customer_email=test@email.com
customer_phone=+38512345678
terminal_id=IN060751
request_hash=01322f2d817a9a3f8bd244f9a77221c3307ac19ae1f1aa72180fd22834f96f0122a07361e9befff272920f1276506a51fac51a7e8c0ee62262ba9c8afb91b823
```

Merchant receives response parameters:

```
account_id=
acquirer=***
card_type=Visa
card_expdate=2404
customer_address=Test Address
customer_city=Test City1
customer_country=Hrvatska
customer_email=test@email.com
customer_lang=hr
customer_name=TestName
customer_surname=TestSurname
```

```
customer_tel=+38512345678
customer_zip=10000
masked_pan=400000*****0000
merchant_id=DEMOMID
oneclick_result=
order_date=2021-10-13 15:38:13.752
order_number=1ORD_1211310_4926
payment_method=
purchase_amount=15.00
purchase_currency=191
purchase_description=Test order
purchase_installments=
request_type=transaction
response_appcode=153819
response_hash=7e5ff52cf33912d78f635b1e4be1969369992e9bb6195025a9a5197465d27aeeafc4af6fb07f7c12
4154bee31af1eacf19cead10efdfaceb61b299cc7d7b329d
response_message=APPROVED
response_result=000
response_systan=003167
transaction_type=auth
```

2.5.2. Preauthorization Request

Payment initiation request: **trantype=preauth**

```
merchant_id=DEMOMID
purchase_amount=2.40
purchase_currency=191
order_number=1ORD_1211310_3939
request_type=transaction
trantype=preauth
purchase_description=Test+order
customer_name=TestName
customer_surname=TestSurname
customer_address=Test+Address
customer_country=Hrvatska
customer_city=Test+City1
customer_zip=10000
customer_lang=hr
customer_email=test@email.com
customer_phone=+38512345678
terminal_id=IN060751
request_hash=3eebb82fdfae91ab864e5a07b2bd7886ee1a4e844a76f1021629af1a36538f8e1c220424dcb54093
9ca36e4e94d39110d234122b32286121507e215cc7d02fd2
```

Merchant receives response parameters:

```

account_id=
acquirer=***
card_type=Visa
card_expdate=2509
customer_address=Test Address
customer_city=Test City1
customer_country=Hrvatska
customer_email=test@email.com
customer_lang=hr
customer_name=TestName
customer_surname=TestSurname
customer_tel=+38512345678
customer_zip=10000
masked_pan=400000*****0000
merchant_id=DEMOMID
oneclick_result=
order_date=2021-10-13 15:42:10.841
order_number=1ORD_1211310_3939
payment_method=
purchase_amount=2.40
purchase_currency=191
purchase_description=Test order
purchase_installments=
request_type=transaction
response_appcode=154216
response_hash=1be6687e1545098d05b81cd35a9f6e61acf9a3e39585ebe5099deef3244f7e9e58e0aba829dbf5
4336bc3843de6869f2562e6e8543a208de97b3c4490be605f8
response_message=APPROVED
response_result=000
response_systan=003168
transaction_type=preauth

```

2.5.3. Completion of the Preauthorized Request

Completion request must refer to the original (preauthorized) transaction using the order_number value:

```

merchant_id=DEMOMID
purchase_amount=2.40
purchase_currency=191
order_number=1ORD_1211310_3939
request_type=transaction
trantype=completion
purchase_description=Test+order
customer_lang=hr

```



```
terminal_id=IN060751
request_hash=1660fab796d5082db795b94756e9cdd449b6495324b9213b19d8b2fde08db78306814fad2adfc77
23e8e54ad60bdb74a8443156dd963bea3287c1251d001d463
```

If a completion request is sent through the webshop, merchant receives response parameters:

```
account_id=
acquirer=***
card_type=
card_expdate=
customer_address=
customer_city=
customer_country=
customer_email=
customer_lang=hr
customer_name=
customer_surname=
customer_tel=
customer_zip=
masked_pan=
merchant_id=DEMOMID
oneclick_result=
order_date=
order_number=1ORD_1211310_3939
payment_method=
purchase_amount=2.40
purchase_currency=191
purchase_description=Test order
purchase_installments=
request_type=transaction
response_appcode=154830
response_hash=c68b3a242449c89f1bb8de9e116cda74f4c47985ccae09998e1efb4d2dba00446d74edd0a50225
c26e1ff5adc18baff7774ac751d9dfe2e061b30f8cfa1ac88a
response_message=APPROVED
response_result=000
response_systan=003168
transaction_type=completion
```

2.5.4. Refund Request

Refund request must also refer to the original transaction using the **order_number** value:

request_type=refund

```
merchant_id=DEMOMID
```

purchase_amount=15.00
purchase_currency=191
order_number=1ORD_1211310_4926
request_type=transaction
trantype=refund
purchase_description=Test+order
customer_name=TestName
customer_surname=TestSurname
customer_address=Test+Address
customer_country=Hrvatska
customer_city=Test+City1
customer_zip=10000
customer_lang=hr
customer_email=test@email.com
customer_phone=+38512345678
terminal_id=IN060751
request_hash=2c3414c7426a446f90b15f1bbc94bc5de2302175edcbb2610bc20cd54b1a08a15e447af03af26503
63a857ecdf8dd393a7f5f10c8025306867ee8f2f19e0c75b
account_id=
acquirer=***
card_type=
card_expdate=
customer_address=Test Address
customer_city=Test City1
customer_country=Hrvatska
customer_email=test@email.com
customer_lang=hr
customer_name=TestName
customer_surname=TestSurname
customer_tel=+38512345678
customer_zip=10000
masked_pan=
merchant_id=DEMOMID
oneclick_result=
order_date=
order_number=1ORD_1211310_4926
payment_method=
purchase_amount=15.00
purchase_currency=191
purchase_description=Test order
purchase_installments=
request_type=transaction
response_appcode=155435
response_hash=ff288cdd3659fe97fb36ef3339a6d7a405d57f9e6f2a16b40186742074cbdaaf9fa9ef3653d8b275
62dae651d8354816a72eae8b9d3d12b56b0e80e102905fca
response_message=APPROVED

```
response_result=000
response_systan=003167
transaction_type=refund
```

2.6. Payment by Link

Instead of initiating payments through the webshop, merchant can also prepare an order on customer demand and provide a payment link that is sent to customer via e-mail. When the customer follows the link, a checkout form will be presented to them with pre-filled order ID, purchase amount and other order details.

To generate payment link for the customer, merchant has to prepare HTTP GET request which will send the payment initiation parameters to iCheckOutX. The link can be presented to customer in text format, as a button, linked image or any other HTML implementation by merchant's choice.

Parameters appended to the payment link's query are the same [request parameters](#) that are used for regular transaction initiation. However, in case of Payment by Link, **HTTP GET** method should be used instead of HTTP POST.

It is important to never include the merchant's [secret key](#) value in the URL, as it is a security parameter that should not be accessible to customer. It is only used for [hash calculation](#), again using the same procedure as in the standard payment requests.

After clicking the link in the e-mail, customer only has to fill out their card data and personal data that is required for continuing the payment.

Example of a fully formed payment link:

```
https://host-
domain/iCheckOutX/v1/checkout/confirm.xhtml?merchant_id=DEMOMID&purchase_amount=15.00&purchas
e_currency=191&order_number=1ORD_122082_4631&request_type=transaction&trantype=auth&purchase_d
escription=Test+order&customer_country=Hrvatska&customer_lang=hr&terminal_id=DEMOTID&request_has
h=2d83f681da0e2afdef5967d5f7e6a1586fed0fb40a0ea98474476dd795e3ba021096b49c3511a20d57de684f59
fbecfb8a173a635dfe354a6b566b2f54f901b0
```

Notice: 'host-domain' should be replaced with the exact system domain.

2.7. Checking Order Status

To check order status at any moment, an HTTP POST request should be made to URL:

<https://securepay.mbhbank.hu/iCheckOutX/v1/checkout/confirm.xhtml>

Notice: 'host-domain' should be replaced with the exact system domain.

If an order with specified order_number exists for the specified merchant, response from iCheckOut will contain the corresponding authorization or preauthorization [response result](#). Transaction is considered approved only if its response_result is equal to "000". Response will also contain other details for queried order such as response_message, order_date, systan and approval_code.

Parameter *order_lifecycle* can obtain the following values:

Case a) Order is not found

order_lifecycle = "Order not found"

Case b) Order is pending

order_lifecycle = "Received" OR order_lifecycle = "Sent"

Case c) Order is declined

order_lifecycle = "Declined" OR order_lifecycle = "Canceled" OR order_lifecycle = "Session expired"

Case d) Order is approved

order_lifecycle = "Authorized" OR order_lifecycle = "Completed" OR order_lifecycle = "Refunded" OR order_lifecycle = "Reversed" A "host-domain" adatokat természetesen a valóban használatban lévő domain-nel kell helyettesíteni

Ha a megadott kereskedőhöz kapcsolódóan létezik megrendelés/tranzakció a megadott egyedi azonosítóval (order_number), az iCheckOut válaszában megadja a vonatkozó normál tranzakció, vagy előengedélyezéses tranzakció eredményére vonatkozó információt ([response result](#)). A tranzakció abban az esetben tekinthető sikeresnek, ha „response_result” értéke "000" lesz. A válasz tartalmazza a lekérdezés mezősorrendjének megfelelő rendben a következő mezők értékeit is is: „response_message”, „order date”, „systan”, „approval code”

Az „order_lifecycle” paraméter a következő értékeket kaphatja:

- a) A megrendelés nem található
order_lifecycle = "Order not found"
- b) A megrendelés függőben van
order_lifecycle = "Received" OR order_lifecycle = "Sent"
- c) A megrendelést elutasították
order_lifecycle = "Declined" OR order_lifecycle = "Canceled" OR order_lifecycle = "Session expired"
- d) A megrendelést jóváhagyták
order_lifecycle = "Authorized" OR order_lifecycle = "Completed" OR order_lifecycle = "Refunded" OR order_lifecycle = "Reversed"

[Response Hash](#) should be calculated the same way as in standard e-commerce transactions. In the following example, these parameters were included in response hash calculation:

```
card_type=PBZ Visa
order_date=2021-09-29 12:32:38.889711
order_number=1ORD_121299_3108
response_appcode=874161
response_message=APPROVED
response_result=000
response_systan=003055
secret_key=XXXXX
```

checkstatus request example

```
merchant_id=DEMOMID
order_number=1ORD_121299_3108
```

```
request_type=checkstatus
terminal_id=IN060751
request_hash=33aba71b36782c729d89c63d3651799eac1496485b4b03d169b97e6e1becb14b46a5e894b7a5c8
ea5b981a3a112956feccc78c1c3d353134c5c12dc99dfdadc6
```

checkstatus response example

```
<response>
  <card_type>PBZ Visa</card_type>
  <order_date>2021-09-29 12:32:38.889711</order_date>
  <order_number>1ORD_121299_3108</order_number>
  <response_appcode>874161 </response_appcode>
  <response_hash>
c65ee526cf852a47f7b6ec64fafa78969fdcbde96a706acb41f1e90b694edcf1ac91bbb995b956bd122a7a01cb0cc
56293c914979756973e9d6004768d752039
  </response_hash>
  <response_message>APPROVED</response_message>
  <response_result>000</response_result>
  <response_systan>003055</response_systan>
</response>
```

2.8. One click payment

iCheckOut system supports one-click payment type orders and transactions. This functionality is typically used when cardholder wants to save his card data and use it in subsequent orders on the webshop. This is useful as cardholder doesn't need to enter his card data with every order, he just selects on the webshop which card he would like to use, and icheckout does all the rest.

For One-Click payment transactions, an HTTP POST request should be sent to URL:

<https://securepay.mbhbank.hu/iCheckOutX/v1/checkout/confirm.xhtml>

Notice: 'host-domain' should be replaced with the exact system domain.

Where **host-domain/** represents a merchant web domain, which is usually different for each merchant.

One-click payment requests that are sent to icheckout are similar as regular authorization requests, with a few modifications in parameters and request types.

There are 5 different request types that can be used for one-click payment transactions:

1. **Register** - request type that is sent for initial card registration for the webshop. Icheckout uses this request to issue a unique account_id for the card, which can subsequently be used for later payments. Note that subfield account_id must be empty in the request, as it will be assigned by icheckout. Parameter account_id is of type UUID and is written as a sequence of lower-case hexadecimal digits, in several groups separated by hyphens, specifically a group of 8 digits followed by three groups of 4 digits followed by a group of 12 digits, for a total of 32 digits. Registered account will not be active until card number, expiry date and card verification data is

assigned to account. If expiry data and/or card verification data is missing during register account request, update account request must be used to add missing elements before account becomes active.

2. Update - request type that is used to update card information for a specific account_id. This should be used when card number, expiry date and/or card verification data needs to be changed or added to the account. Any number of elements can be updated with one update request. Elements that must remain unchanged must not be sent in update account request.

3. Get - This is instruction for icheckout to get card data from existing account and return it to webshop. Card PAN is always returned masked. Request must contain a valid account_id .

4. Use - This is instruction for icheckout to use existing (stored) account information to retrieve card number, expiry date and card verification data and use it for authorization, preauthorization, completion, reversal or refund. Request must contain a valid account_id.

5. Delete - This is instruction for icheckout to delete existing account, and should be used when account is not required any more. Request must contain a valid account_id.

Register account

The following table contains request parameters that need to be sent for valid account registration.

Presence is defined as:

M - mandatory, must be present in the message.

O - optional, present if information is available.

Parameter name	Usage
trantype	M → only auth or preauth
request_type	M → fixed value ' register '
request_hash	M
purchase_amount	M
purchase_currency	M
purchase_installments	O
order_number	M
merchant_id	M

terminal_id	M
payment_method	O
purchase_description	O
customer_lang	O
customer_name	O
customer_surname	O
customer_address	O
customer_country	O
customer_city	O
customer_zip	O
customer_phone	O
customer_email	O

Register request (without initiating a payment transaction)

Account registration request is sent to icheckout when a customer selects on a webshop page that he wants to save his card data so that he can use the same card for later payments on the webshop. Customer can save any number of cards, and can use any of them after they are properly registered on the icheckout payment system. Card registration request will be used to generate a unique account_id number that is used for all subsequent payments with use account requests. For card registration, an icheckout form will be presented to the customer, in which he must enter all card information (pan, expiry date, cvv...), and when the customer selects "SAVE" button on the form, card will be registered and the transaction will be processed as preauthorization transaction with generated account_id. The transaction is processed as preauthorization to verify the customer account. If the preauthorization is declined, card registration will also be declined. If account_id is generated, it will be returned in response parameters, and that same account_id information can be later used to make subsequent orders with "Use account" request type.

Response:

If the transaction is approved response message will contain account_id response parameter filled with assigned identification code. WEB shop can use this ID code in subsequent requests to identify payment information (card number, expiry date and card verification data). If the account registration is declined, account_id parameter will be empty. Payment gateway will reject request for creating account if such functionality is not activated for merchant.

NOTE: customer card data information is not saved on the webshop domain by any means. Customer enters card data manually, and when account_id is generated by icheckout, it stores encrypted card data that is used for transaction authorization. Also, card PAN is returned masked to the webshop in response parameters, so it can be used as a reference to the customer and to the webshop.

Update account

The following table contains request parameters that need to be sent for valid account update.

Presence is defined as:

M - mandatory, must be present in the message.

O - optional, present if information is available.

C - conditional, must be present only if the parameter needs to be updated.

Parameter name	Usage
trantype	O
request_type	M → fixed value: 'update'
request_hash	M
merchant_id	M
terminal_id	M
account_id	M
card_pan	C
card_expdate	C
card_cvd	C

Request:

For *update account* requests, a customer selects on the webshop which card they would like to modify and according to the selection, webshop sends an update account request to icheckout. A valid account_id must be sent in request so that icheckout can correctly update card data. Upon receiving an update account request, icheckout generates a customer form for card data modification in which a customer enters data that needs to be modified. For example, if only expiry date needs updating, update account request must contain new and valid card_expdate parameter and a valid account_id. Parameters like card_pan and card_cvd must be empty in this case.

Response:

The oneclick_result response parameter will contain update confirmation value. Possible values are:

- 300** = Successful
- 301** = Reserved for future use
- 302** = Unable to update, account ID doesn't exist.
- 303** = Reserved for future use
- 304** = Failed, technical error (retry at later time)

Get account

The following table contains request parameters that need to be sent for valid account info fetch.

Presence is defined as:

M - mandatory, must be present in the message.

O - optional, present if information is available.

Parameter name	Usage
trantype	O
request_type	M → fixed value: 'get'
request_hash	M
merchant_id	M
terminal_id	M
account_id	M

Request:

This type of request is used to fetch card information data based on a valid account_id.

Response:

The oneclick_result response parameter will contain retrieval confirmation value. Possible values are:

- 300** = Successful

301 = Reserved for future use

302 = Unable to update, account ID doesn't exist.

303 = Reserved for future use

304 = Failed, technical error (retry at later time)

Use account

The following table contains request parameters that need to be sent for valid account usage.

Presence is defined as:

M - mandatory, must be present in the message.

O - optional, present if information is available.

Parameter name	Usage
trantype	M
request_type	M → fixed value: 'use'
request_hash	M
merchant_id	M
terminal_id	M
payment_method	O
account_id	M
purchase_description	O
purchase_amount	M
purchase_currency	M
order_number	M
customer_lang	O
customer_name	M: for auth/preauth; O: otherwise

customer_surname	M: for auth/preauth; O: otherwise
customer_address	M: for auth/preauth; O: otherwise
customer_country	M: for auth/preauth; O: otherwise
customer_city	M: for auth/preauth; O: otherwise
customer_zip	M: for auth/preauth; O: otherwise
customer_phone	M: for auth/preauth; O: otherwise
customer_email	M: for auth/preauth; O: otherwise

Request:

This type of one-click payment request is used to make payments on a webshop using a valid and registered account_id. Transaction types that can be processed are: authorization (purchase), preauthorization, completion, reversal and refund. Card information data doesn't need to be sent in the request. Instead, a valid and registered account_id parameter needs to be used.

Response:

The oneclick_result response parameter will contain confirmation value. Possible values are:

300 = Successful

301 = Operation not permitted

302 = Request rejected, account ID doesn't exist.

303 = Reserved for future use

304 = Failed, technical error

Delete account

The following table contains request parameters that need to be sent for valid account deletion.

Presence is defined as:

M - mandatory, must be present in the message.

O - optional, present if information is available.

Parameter name	Usage
trantype	O
request_type	M → fixed value 'delete'
request_hash	M
merchant_id	M
terminal_id	M
account_id	M

Request:

This type of request is used to unregister a specific cardholder card. After account is successfully unregistered, subsequent payments with Use account requests won't be possible unless a new register request is sent. A valid account_id parameter must be presented in the request.

Válasz:

The oneclick_result response parameter will contain deletion confirmation value. Possible values are:

300 = Successful

301 = Reserved for future use

302 = Unable to delete, account ID doesn't exist.

303 = Reserved for future use

304 = Failed, technical error (retry at later time)

Registration with authorization example**Register request:**

```
trantype=auth
request_type=register
purchase_amount=15.00
purchase_currency=191
order_number=1ORD_1211310_9607
merchant_id=SAMPLEMID
terminal_id=SAMPLETID
purchase_description=Test+order
customer_name=TestName
customer_surname=TestSurname
customer_address=Test+Address
customer_country=Hrvatska
customer_city=Test+City1
customer_zip=10000
customer_lang=hr
customer_email=test@email.com
customer_phone=+38512345678
request_hash=78ec32466a1b3f40533ec6d38304c3c9d4e87e96a71629c7ea8a4f0d68fa78355c79b18392e6030
eb6c0a6d28ea2819357d3f3ea3581fb34738d114cea1d974b
```

On icheckout payment form, customer enters their card data:

```
card_pan=4*****0000
card_expdate=2701
card_cvd=***
```

Register response:

```
account_id=23d92892-2c2e-11ec-bead-f0b913d67fdb
acquirer=***
card_type=Visa
card_expdate=2606
customer_address=Test Address
customer_city=Test City1
customer_country=Hrvatska
customer_email=test@email.com
customer_lang=hr
customer_name=TestName
customer_surname=TestSurname
customer_tel=+38512345678
customer_zip=10000
masked_pan=400000*****0000
```

```

merchant_id=DEMOMID
oneclick_result=
order_date=2021-10-13 16:01:57.316
order_number=1ORD_1211310_9607
payment_method=
purchase_amount=15.00
purchase_currency=191
purchase_description=Test order
purchase_installments=
request_type=register
response_appcode=160204
response_hash=750181435d591d12d0f3fc393c236ce136a1e151bacedf4906ec6aeb8b71b4d7e0ac8342681c98
db6a7e75df2c37c6ed1503c0bfe91dcd8551974d4db9430c3a
response_message=APPROVED
response_result=000
response_systan=003169
transaction_type=auth

```

Since **account_id** parameter is received in response, account registration was successful.

Update account example

In previous card registration example, a newly assigned account ID was received: 842ec9ca-cde9-11eb-8738-7acb13d67fdb which can now be used as an example for an account update request.

Update account request:

```

trantype=auth
request_type=update
merchant_id=SAMPLEMID
terminal_id=SAMPLETID
account_id=842ec9ca-cde9-11eb-8738-7acb13d67fdb
card_expdate=2408
card_cvd=444
request_hash=3f9dba73c8a70818aab720d30076206fd38860bba972ef443e39c7f203950a9928f1693f50f2f7eda
84f6c4da0707413b166a5de723713333cb9ccf6c80d874b

```

Update account response:

```
APPROVED
account_id=
acquirer=
card_type=
card_expdate=2408
customer_address=
customer_city=
customer_country=
customer_email=
customer_lang=
customer_name=
customer_surname=
customer_tel=
customer_zip=
masked_pan=
merchant_id=SAMPLEMID
oneclick_result=300
order_date=
order_number=
payment_method=
purchase_amount=
purchase_currency=
purchase_description=
purchase_installments=
request_type=update
response_appcode=
response_hash=8c24f5fd9546b638482ae44525ea47c0fcc2bdf87c5df84f923b09b67987ed8d71dde7345679064
afe251a20652369b6894bfce7b06f65a9c1916d7aeba76071
response_message=APPROVED
response_result=000
response_systan=
transaction_type=auth
```

Since value **'300'** was received in parameter **oneclick_result**, the account was successfully updated.

Get account example**Get account request:**

```

trantype: auth
request_type: get
merchant_id: SAMPLEMID
terminal_id: SAMPLETID
account_id: 842ec9ca-cde9-11eb-8738-7acb13d67fdb
request_hash:
5ad15ebb79c1cc6edd86c63251217aa5eae496c702fcd9e49bea5a4b61b26877469cf04b74500e8e0573f1659cc9
4fbc4289938b684a036bd3be732f9d22743e

```

Get account response:

```

APPROVED
account_id=842ec9ca-cde9-11eb-8738-7acb13d67fdb
acquirer=
masked_pan: 4000*****00
card_expdata: 2312
customer_address=
customer_city=
customer_country=
customer_email=
customer_lang=
customer_name=
customer_surname=
customer_tel=
customer_zip=
discount_amount=
masked_pan=
merchant_id=DEMOBANK
oneclick_result=300
order_date=
order_number=
payment_method=
purchase_amount=
purchase_currency=
purchase_description=
purchase_installments=
request_type=get
response_appcode=
response_hash=b5717b459b745d469b7189910f5ed90781a912d0f4abeafb9c7e7e87abbd0ffd4a81eb91d32fec
32c1c4d98769abd11aceff31c9c58cd68f7e8e3890f6a3eb25
response_message=APPROVED

```



```
response_result=000
response_systan=
transaction_type=auth
```

The get account request was successful since parameters **masked_pan** and **card_expdate** are received in response.

Use account example

Use account request:

```
trantype: preauth
request_type: use
merchant_id: SAMPLEMID
terminal_id: SAMPLETID
account_id: 842ec9ca-cde9-11eb-8738-7acb13d67fdb
purchase_amount: 1.00
purchase_currency: 191
order_number: ORD_TEST_12345
payment_method:
purchase_description: Test
customer_lang: en
customer_name: TestName
customer_surname: TestSurname
customer_address: TestAddress
customer_country: Hungary
customer_city:
customer_zip:
customer_phone:
customer_email: test@email.com
request_hash:
8ea5147d13e8a1869432280249272eb7cf44ede061918b39c142f7824fadd628784ae5b1a440006bcffa30b5bb55
bc6cbe8a2dc0f1a496cc7b7d16485dd1b0f9
```

Use account response:

```
APPROVED
customer_address: TestAddress
customer_city:
customer_country: Hungary
customer_email: test@email.com
customer_lang: en
customer_name: TestName
customer_surname: TestSurname
customer_tel:
customer_zip:
order_date:
order_number: ORD_TEST_12345
payment_method:
purchase_amount: 1.00
purchase_currency: 191
purchase_description: Test
purchase_installments: 0
request_type: use
response_appcode: 889834
response_message: APPROVED
response_result: 000
transaction_type: preauth
account_id: 842ec9ca-cde9-11eb-8738-7acb13d67fdb
oneclick_result: 300
response_hash=750181435d591d12d0f3fc393c236ce136a1e151bacedf4906ec6aeb8b71b4d7e0ac8342681c98
db6a7e75df2c37c6ed1503c0bfe91dcd8551974d4db9430c3a
```

The preauthorization was approved since **oneclick_result=300** was received.

Delete account example**Delete account request:**

```
trantype: auth
request_type: delete
merchant_id: SAMPLEMID
terminal_id: SAMPLETID
account_id: 842ec9ca-cde9-11eb-8738-7acb13d67fdb
request_hash:
7278b3541776d467b7257dc11037d7b0ce79c08427f841ec225d426af1b3093e6ed0c28205ed4b8651925595dcf
a6cf284da1cb310947dd81d67b3a57619ae0b
```

Delete account response:

```
APPROVED

transaction_type: auth
request_type: delete
merchant_id: SAMPLEMID
terminal_id: SAMPLETID
account_id: 842ec9ca-cde9-11eb-8738-7acb13d67fdb
response_message: APPROVED
response_result: 000
oneclick_result: 300
response_hash=b5717b459b745d469b7189910f5ed90781a912d0f4abeafb9c7e7e87abbd0ffd4a81eb91d32fec
32c1c4d98769abd11aceff31c9c58cd68f7e8e3890f6a3eb25
```

The account was successfully deleted since **oneclick_response=300** was received.

I. Annex – Logo placement policy

Is mandatory to display Card Companies's logos on the webshop. Available as a [separate file](#) on the Bank's website.

II. Annex – Test cases

When testing, transaction request should be sent to the following URL:

<https://icheckoutx-test.empiretransactions.eu/iCheckOutX/v1/icheckout/confirm.xhtml> (no authentication)

The test cases required to test successful integration are listed in the table below.

In the vPOS terminal card acquiring contract, the merchant has indicated whether it wishes to initiate additional types of transactions other than the standard payment transactions.

The successful completion of the test cases included in the test group(s) assigned to the transaction type(s) specified in the contract is a prerequisite for the go live of the vPOS terminal and the delivery of the security key (to production environment).

Transaction type/ vPOS services	1. test group	2. test group	3. test group	4. test group
Purchase	+			
Pre-authorization	+	+		
Recurring	+		+	
One click purchase	+			+

1. Test group

Test name	Transaction to be checked	Comment	Test case successful / unsuccessful
Purchase with Mastercard card	Request type: transaction Tran. type: auth	Minimum expected number of transactions: 10 pcs.	
Purchase with VISA card	Request type: transaction Tran. type: auth	Minimum expected number of transactions: 10 pcs.	
Unsuccessful purchase	Request type: transaction Tran. type: auth Transaction amount: 557 huf (declined) 558 huf (expired card) 559 huf (blocked card) 560 huf (invalid acceptor) 560 huf (insufficient funds)	Test can only be started with Mastercard type card. A minimum of one transaction is required for each amount indicated.	

2. Test group

Test name	Transaction to be checked	Comment	Test case successful / unsuccessful
Pre-authorization with Mastercard card	Request type: transaction Tran. type: pre auth	Minimum expected number of transactions: 10 pcs.	
Pre-authorization with VISA card	Request type: transaction Tran. type: pre auth	Minimum expected number of transactions: 10 pcs.	
Unsuccessful Pre-authorization	Request type: transaction Tran. type: pre auth Transaction amount: 557 huf (declined) 558 huf (expired card) 559 huf (blocked card) 560 huf (invalid acceptor) 560 huf (insufficient funds)	Test can only be started with Mastercard type card. A minimum of one transaction is required for each amount indicated.	

3. Test group

Test name	Transaction to be checked	Comment	Test case successful / unsuccessful
First transaction - Recurring with Mastercard card	Request type: recurring Tran. type: auth	Minimum expected number of transactions: 5 pcs.	
Following transaction - Recurring with Mastercard card	Request type: recurring Tran. type: auth	Minimum of one successful following transaction is expected for each first transaction.	
First transaction - Recurring with VISA card	Request type: recurring Tran. type: auth	Minimum expected number of transactions: 5 pcs.	

Test name	Transaction to be checked	Comment	Test case successful / unsuccessful
Following transaction - Recurring with VISA card	Request type: recurring Tran. type: auth	Minimum of one successful following transaction is expected for each first transaction.	
Unsuccessful recurring	Request type: recurring Tran. type: auth Transaction amount: 557 huf (declined) 558 huf (expired card) 559 huf (blocked card) 560 huf (invalid acceptor) 560 huf (insufficient funds)	Test can only be started with Mastercard type card. A minimum of one transaction is required for each amount indicated.	

4. Test group

Test name	Transaction to be checked	Comment	Test case successful / unsuccessful
One-click payment – purchase and registration with Mastercard card	Request type: register Tran. type: auth or pre auth	Minimum expected number of transactions: 5 pcs.	
One-click payment – Purchase with saved Mastercard payment instrument data (token)	Request type: use Tran. type: auth or pre auth	Minimum expected number of transactions: 5 pcs.	
One-click payment – saved Mastercard payment instrument data modify/update	Request type: update Tran. type: auth or pre auth	Minimum expected number of transactions: 5 pcs.	
One-click payment – saved Mastercard payment instrument data query	Request type: get Tran. type: auth or pre auth	Minimum expected number of transactions: 5 pcs.	
One-click payment – saved Mastercard payment instrument data delete	Request type: delete Tran. type: auth or pre auth	Minimum expected number of transactions: 5 pcs.	

Test name	Transaction to be checked	Comment	Test case successful / unsuccessful
One-click payment – purchase and registration with VISA card	Request type: register Tran. type: auth or pre auth	Minimum expected number of transactions: 5 pcs.	
One-click payment – Purchase with saved VISA payment instrument data	Request type: use Tran. type: auth or pre auth	Minimum expected number of transactions: 5 pcs.	
One-click payment – saved VISA payment instrument data modify/update	Request type: update Tran. type: auth or pre auth	Minimum expected number of transactions: 5 pcs.	
One-click payment – saved VISA payment instrument data query	Request type: get Tran. type: auth or pre auth	Minimum expected number of transactions: 5 pcs.	
One-click payment – saved VISA payment instrument data delete	Request type: delete Tran. type: auth or pre auth	Minimum expected number of transactions: 5 pcs.	
One-click payment – Unsuccessful purchase	Request type: use Tran. type: auth or pre auth Transaction amount: 557 huf (declined) 558 huf (expired card) 559 huf (blocked card) 560 huf (invalid acceptor) 560 huf (insufficient funds)	Test can only be started with Mastercard type card. A minimum of one transaction is required for each amount indicated.	

5. Test group – Optional test – pay by link

The request parameters used for a pay by link request are the same as for a normal transaction request. However, if you use pay by link, you must use the **HTTP GET** method instead of HTTP POST.

Test name	Transaction to be checked	Comment	Test case successful / unsuccessful
-----------	---------------------------	---------	-------------------------------------

Purchase with Mastercard card	Request type: transaction Tran. type: auth		
Purchase with VISA card	Request type: transaction Tran. type: auth		

III. Annex – Cards for testing

Card numbers of the test cards:

- VISA card:
4176660000000100
- Mastercard (Maestro) card:
5457210001000019
5346931300108113
6799990100000159

For the above test cards, any expiration date and CVC2/CVV2 can be used.

- It will always be rejected regardless of the amount:

Card number: 5444 0340 7823 8013

Expiration date: 11/24

CVC2/CVV2: 916

IV. Annex – Scripts/samples for webshop development

Available as a [separate file](#) on the Bank's website.

V. Annex – Secret key on the Merchant Portal ('Kereskedői portál')

After successful login, the merchant can access the secret key in the 'Account settings' menu, which must be included in the payment request.

The security key is available by clicking on the 'Show key' button:

Password modify

Current password
Please, give your current password...

New password
Please, give your new password...

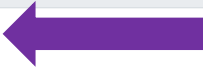
Confirm new password
Please, confirm your new password...

Save

View security key

Security key

Show key



VI. Annex – Technical support

Technical support is provided by the staff of Compuworx Zrt. acting on behalf of the Bank, who must receive confirmation by email of the completion of the test cases described in Annex II. If you have any technical questions regarding the integration and testing, or would like to receive confirmation of successful completion of the tests, please contact us at vpos.support@compuworx.hu