Technical Documentation API

Money Transfer Hub provides possibility to process Person-2-Person and Person-2-Merchant transactions in various forms. Please check details in the below documentation.

> This documentation contains the methods for **mobile-server** integration. The methods included in the documentation are intended for Customers creating their own mobile SDK.

> The Customer creating the SDK must also remember about the integration with the MobileDC component

> Documentation for the **server-to-server** integration is available here but is **deprecated.**

Receiver types which can be used to set Receiver.Type

Based on ReceiverType user can fill different field in Receiver object in requests.

ReceiverType	Description
BARE_CARD_NUMBER	Bare card number in **Receiver.card** field
FRIEND_ID	Should pass FriendId in **Receiver.Card** field
WALLET_CARD_ID	Should pass DataCoreCardId to **Receiver.Card** field and UserDataCoreCardId to **Receiver.userId** field
EMPTY	Means that the receiver have the same card data like sender. This type may be useful on Determine Currency

JWE

Peer To Peer Transaction Service supports encryption of requests and responses as standard JSON Web Encryption (JWE) per RFC 7516.

Recommended to read the JWE standard: RFC 7516.

Methods that support request encryption in the JWE standard are tagged in the documentation with the header: *Content-Type:application/x-jwe-encryption-body+json*. If the response is to be encrypted with the JWE standard then the header must be added: *X-Encryption-Public-Key* with the public key.

Processing requests and responses can be divided into 4 options listed below:

1. Base request → Base response - the following headers should be provided to pass this case:
   - *Content-Type: application/json*
2. Base request → Encrypted response - the following headers should be provided to pass this case:
   - *Content-Type: application/json*
3. Encrypted request → Base response - the following headers should be provided to pass this case:
   - *Content-Type: application/x-jwe-encryption-body+json*
4. Encrypted request → Encrypted response - the following headers should be provided to pass this case:
   - *Content-Type: application/x-jwe-encryption-body+json*

Overview

JWE represents encrypted content using JSON data structures and Base64 encoding. The representation consists of three parts: a JWE Header, a encrypted payload, and a signature. The three parts are serialized to UTF-8 bytes, then encoded using base64url encoding. The JWE's header, payload, and signature are concatenated with periods (.).

JWE typically takes the following form:

```
{Base64 encoded header}.{Base64 encoded payload}.{Base64 encoded signature}
```

JWE header contains:

Type	Value	Constraints	Description

alg	RSA-OAEP-256	Required	Identifies the cryptographic algorithm used to secure the JWE Encrypted Key. Supported algorithms: **RSA-OAEP-256**, **RSA-OAEP-384**, **RSA-OAEP-512**. Recommend value: **RSA-OAEP-256**.
enc	A256GCM	Required	Identifies the cryptographic algorithm used to secure the payload. Supported algorithms: **A128GCM**, **A192GCM**, **A256GCM**, **A128CBC-HS256**, **A192CBC-HS384**, **A256CBC-HS512**. Recommend value: **A256GCM**.
typ	JOSE	Optional	Identifies the type of encrypted payload. Recommend value: **JOSE**.
iat	1637929226	Optional	Identifies the time of generation of the JWT token. Supported date format: unix time in UTC. In the case of *iat*send, the validity of JWE is validated. Recommend send the header due to the increase in the security level.
kid	5638742a5094327fcd7a5945d06a45a9d83e9006	Optional	Identifies the public key of use to encrypt payload. Supported format: SHA-1 value of the public key. In the case of *kid* send, the validity of public key is validated, so we can inform the client that the public key has changed.

Payload Encryption

Every encrypted request should include JWE token. The jwe token should be passed in the field: *value*.

In case of problems with the implementation of JWE, please contact the administrator.

To prepare the encrypted payload:

The steps may differ depending on the libraries used.

1. Get the public key using the method: [???](#Get publicKey). The public key is encoded with Base64.
2. Decode the public key.
3. Then create a correct object to be encrypted.
4. Encrypt the created object with the public key.
5. Create JWE header compatible with: JWE Header
6. Make a request on the method that supports JWE. Set the JWE token in the field: *value*. Methods supporting JWE use the following header: *Content-Type:application/x-jwe-encryption-body+json*.

Payload Decryption

To prepare the decrypted payload:

The steps may differ depending on the libraries used.

The cryptographic algorithm used to secure the payload is: **A256GCM**, while to secure the encrypted JWE key: **RSA-OAEP-256**.

1. For the response to be encrypted you need to send *public key* in the header: *X-Encryption-Public-Key*. The header value must be encoded *Base64*.
2. After receiving the response, you should get the JWE token from the field: *value*.
3. Decrypt the JWE token from the field: *value* with the private key.

Public key format to be encoded in Base64.

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0IDAQAB...
-----END PUBLIC KEY-----
```

P2P

Every single method should contains Authorization and Mobile-Product headers.

Active Accounts

Method used to find users with valid mc card type (not expired, strong verified). Response will contain phone numbers with user and card identifiers. Users without accepted TOS or without valid MC card will not be returned in response. If user has multiple cards that match criteria response

will contain only user’s default card id.

Request

Request headers

Request body with header: *X-Encryption-Public-Key*

Type	Value	Constraints	Description
Authorization	Mobile bG9naW46YWNrbWU=	Required	Device token with "Mobile " prefix
Product-Name	TestProduct	Required	Application product name
Content-Type	application/x-jwe-encryption-body+json	Optional	Header must be present if the request body is encrypted using the JWE standard.
X-Encryption-Public-Key		Optional	Header must be present if the response body is to be encrypted using the JWE standard. Public key must be encoded Base64.

Request fields

Response

Error response - ERROR_VALIDATION.

```
HTTP/1.1 400 BAD REQUEST
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "traceId": "{{traceId}}",
```

```
    "errorStatus": "ERROR_VALIDATION",
    "message": "Some fields are invalid",
    "data": [
        {
            "field": "{{field_name_from_request}}",
            "message": "{{message}}"
        }
    ]
}
```

Error response - ERROR_BAD_TOKEN.

```
HTTP/1.1 400 BAD REQUEST
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "traceId": "{{traceId}}",
    "errorStatus": "ERROR_BAD_TOKEN"
}
```

Error response - PRODUCT_NOT_FOUND.

```
HTTP/1.1 404 NOT FOUND
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "traceId": "{{traceId}}",
    "errorStatus": "PRODUCT_NOT_FOUND",
```

```
    "message": "Product by name {{product_name}} not found."
}
```

Error response - INTERNAL_SERVER_ERROR.

```
HTTP/1.1 500 INTERNAL SERVER ERROR
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "traceId": "{{traceId}}",
    "errorStatus": "INTERNAL_SERVER_ERROR"
}
```

Response fields

Errors

Encrypted response fields when sent header: *X-Encryption-Public-Key*

Http Status	Error Status	Description
400 - Bad Request	ERROR_VALIDATION	Some fields are invalid
400 - Bad Request	ERROR_BAD_TOKEN	Invalid authorization token
400 - Bad Request	CRYPTOGRAPHY_ERROR	Error decoding public key has sent in header: *X-Encryption-Public-Key*
400 - Bad Request	CRYPTOGRAPHY_ERROR	Error on decrypting request
400 - Bad Request	CRYPTOGRAPHY_ERROR	Error on encrypting response
400 - Bad Request	CRYPTOGRAPHY_ERROR	JWE encryption Key is invalid
400 - Bad Request	CRYPTOGRAPHY_ERROR	JWE payload is expired
400 - Bad Request	INVALID_PHONE_NUMBERS	Phone numbers has incorrect format
404 - Not Found	PRODUCT_NOT_FOUND	Product not found based on sent header: *Product-Name*

500 - Internal Server Error	INTERNAL_SERVER_ERROR	Internal application error

Examples

Determine currency

Request body with header: *X-Encryption-Public-Key*.

Method is used to determine currencies applied for given sender and receiver cards.

Request

Receiver.receiverType = WALLET_CARD_ID.

```
POST /mobile-api/determine-currency HTTP/1.1
Content-Type: application/json
Authorization: Mobile bG9nzW46YWNrbWU=
Product-Name:
Content-Type: application/json
Content-Length: 56

{
    "sender": {
        "cardId": "219754"
    },
    "receiver": {
        "card": ["2","1","4","4","9","2"],
        "userId": "1223",
        "receiverType": "WALLET_CARD_ID"
    }
}
```

Receiver.receiverType = FRIEND_ID.

```
POST /mobile-api/determine-currency HTTP/1.1
Content-Type: application/json
Authorization: Mobile bG9nzW46YWNrbWU=
```

```
Product-Name: TestProduct
Content-Length: 56

{
    "sender": {
        "cardId": "219754"
    },
    "receiver": {
        "userId": "21",
        "receiverType": "FRIEND_ID"
    }
}
```

Receiver.receiverType = EMPTY.

```
POST /mobile-api/determine-currency HTTP/1.1
Content-Type: application/json
Authorization: Mobile bG9nzW46YWNrbWU=
Product-Name: TestProduct
Content-Length: 56

{
    "sender": {
        "cardId": "219754"
    },
    "receiver": {
        "receiverType": "EMPTY"
    }
}
```

Receiver.receiverType = BARE_CARD_NUMBER.

```
POST /mobile-api/determine-currency HTTP/1.1
Content-Type: application/json
Authorization: Mobile bG9nzW46YWNrbWU=
Product-Name: TestProduct
Content-Length: 56

{
    "sender": {
```

```
        "cardId": "219754"
    },
    "receiver": {
        "card": ["2","2","2","1","0","0","4","0","7","2","1","9","0","1","8","5"],
        "receiverType": "BARE_CARD_NUMBER"
    }
}
```

Request headers

Request body with header: X-Encryption-Public-Key

Type	Value	Constraints	Description
Authorization	Mobile bG9naW46YWNrbWU=	Required	Device token with "Mobile " prefix
Product-Name	TestProduct	Required	Application product name
Content-Type	application/x-jwe-encryption-body+json	Optional	Header must be present if the request body is encrypted using the JWE standard.
X-Encryption-Public-Key		Optional	Header must be present if the response body is to be encrypted using the JWE standard. Public key must be encoded Base64.

Request fields

Response

Error response - ERROR_VALIDATION.

```
HTTP/1.1 400 BAD REQUEST
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
```

```
{
    "traceId": "{{traceId}}",
    "errorStatus": "ERROR_VALIDATION",
    "message": "Some fields are invalid",
    "data": [
        {
            "field": "{{field_name_from_request}}",
            "message": "{{message}}"
        }
    ]
}
```

Error response - ERROR_BAD_TOKEN.

```
HTTP/1.1 400 BAD REQUEST
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "traceId": "{{traceId}}",
    "errorStatus": "ERROR_BAD_TOKEN"
}
```

Error response - PRODUCT_NOT_FOUND.

```
HTTP/1.1 404 NOT FOUND
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
```

```
{
    "traceId": "{{traceId}}",
    "errorStatus": "PRODUCT_NOT_FOUND",
    "message": "Product by name {{product_name}} not found."
}
```

Error response - INTERNAL_SERVER_ERROR.

```
HTTP/1.1 500 INTERNAL SERVER ERROR
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "traceId": "{{traceId}}",
    "errorStatus": "INTERNAL_SERVER_ERROR"
}
```

Response fields

Errors

Encrypted response fields when sent header: *X-Encryption-Public-Key*

Http Status	Error Status	Description
400 - Bad Request	ERROR_VALIDATION	Some fields are invalid
400 - Bad Request	ERROR_BAD_TOKEN	Invalid authorization token
400 - Bad Request	CRYPTOGRAPHY_ERROR	Error decoding public key has sent in header: *X-Encryption-Public-Key*
400 - Bad Request	CRYPTOGRAPHY_ERROR	Error on decrypting request
400 - Bad Request	CRYPTOGRAPHY_ERROR	Error on encrypting response
400 - Bad Request	CRYPTOGRAPHY_ERROR	JWE encryption Key is invalid
400 - Bad Request	CRYPTOGRAPHY_ERROR	JWE payload is expired

400 - Bad Request	ERROR_SENDER_CARD_NOT_ACTIVE	Sender card is not active
400 - Bad Request	ERROR_RECEIVER_CARD_NOT_ACTIVE	Receiver card is not active
400 - Bad Request	UNKNOWN_ERROR	Unknown error
404 - Not Found	PRODUCT_NOT_FOUND	Product not found based on sent header: *Product-Name*
404 - Not Found	CANT_FIND_CARD	Not found card
404 - Not Found	FRIEND_NOT_EXISTS	Not found friend
500 - Internal Server Error	INTERNAL_SERVER_ERROR	Internal application error
500 - Internal Server Error	ERROR_ON_GETTING_DEFAULT_CARD	Error on getting card for friend
500 - Internal Server Error	FENIGE_ERROR	Fenige error

Examples

Currency Rate

Request body with header: *X-Encryption-Public-Key*.

Method is used for determine currency rate for revaluation from funding to payment (lowerRate) and payment to funding (higherRate).
Notice that `lowerRate` is used to transaction processing.

Api Send-money allows users to select the direction of revaluation by providing specify type value in send-money request.
1 - User by selecting type = SENDER defines amount of funding in given currency. This amount is collected from sender card in selected currency.
2 - User by selecting type = RECEIVER defines amount of payment in given currency.
This amount is transferred to receiver card in selected currency.In case there’s need revaluation from one currency to another, system uses lowerRate for situation 1 and higherRate for situation 2

Request

Request headers

Request body with header: *X-Encryption-Public-Key*

Type	Value	Constraints	Description

Authorization	Mobile bG9naW46YWNrbWU=	Required	Device token with "Mobile " prefix
Product-Name	TestProduct	Required	Application product name
X-Encryption-Public-Key		Optional	Header must be present if the response body is to be encrypted using the JWE standard. Public key must be encoded Base64.

Response

Error response - ERROR_BAD_TOKEN.

```
HTTP/1.1 400 BAD REQUEST
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "traceId": "{{traceId}}",
    "errorStatus": "ERROR_BAD_TOKEN"
}
```

Error response - PRODUCT_NOT_FOUND.

```
HTTP/1.1 404 NOT FOUND
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "traceId": "{{traceId}}",
```

```
    "errorStatus": "PRODUCT_NOT_FOUND",
    "message": "Product by name {{product_name}} not found."
}
```

Error response - INTERNAL_SERVER_ERROR.

```
HTTP/1.1 500 INTERNAL SERVER ERROR
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "traceId": "{{traceId}}",
    "errorStatus": "INTERNAL_SERVER_ERROR"
}
```

Response fields

Errors

Encrypted response fields when sent header: *X-Encryption-Public-Key*

Http Status	Error Status	Description
400 - Bad Request	ERROR_BAD_TOKEN	Invalid authorization token
400 - Bad Request	CRYPTOGRAPHY_ERROR	Error decoding public key has sent in header: *X-Encryption-Public-Key*
400 - Bad Request	CRYPTOGRAPHY_ERROR	Error on encrypting response
400 - Bad Request	CRYPTOGRAPHY_ERROR	JWE encryption Key is invalid
400 - Bad Request	CRYPTOGRAPHY_ERROR	JWE payload is expired
404 - Not Found	PRODUCT_NOT_FOUND	Product not found based on sent header: *Product-Name*
500 - Internal Server Error	INTERNAL_SERVER_ERROR	Internal application error
500 - Internal Server Error	FENIGE_ERROR	Fenige error

Examples

Calculate commission

Request body with header: *X-Encryption-Public-Key*.

This method is used to receive information about the commission that will be charged for the transaction. Additional description:

- If value the field: "reconciliationType" is "PLUS", the commission during the transaction will be added to the amount sent (the amount charged from the sender will be increased by a commission).
- If value the field: "reconciliationType" is "MINUS", then the commission during the transaction will be deducted from the amount received (the amount that will be received by the receiver will be reduced by the commission).
- If value the field: "reconciliationType" is "DEPOSITED", the commission during the transaction will neither be subtracted nor added (the amount to be received by the receiver is the same as the amount sent).

In addition, the user may specify in the field: type two values **SENDER** or **RECEIVER**.
After selecting the value: **SENDER**, the transaction will be sent in the amount indicated in the field: amount. Whereas after choosing the value: **RECEIVER**, the transaction will be received in the amount indicated in the field: amount. The method allows user to calculate commissions for the currencies that have been entered.

Request

Receiver.receiverType = WALLET_CARD_ID.

```
POST /mobile-api/calculate-commission HTTP/1.1
Content-Type: application/json
Authorization: Mobile bG9nzW46YWNrbWU=
Product-Name: TestProduct
Content-Length: 101

{
    "amount": 200078,
    "type": "RECEIVER",
    "sender":{
        "cardId":"219834",
```

```
        "currency":"PLN"
    },
    "receiver":{
        "userId": 2345,
        "card": ["2","2","1","2","4","5"],
        "currency": "PLN",
        "receiverType": "WALLET_CARD_ID"
    }
}
```

Receiver.receiverType = FRIEND_ID.

```
POST /mobile-api/calculate-commission HTTP/1.1
Content-Type: application/json
Authorization: Mobile bG9nzW46YWNrbWU=
Product-Name: TestProduct
Content-Length: 101

{
    "amount": 200078,
    "type": "RECEIVER",
    "sender":{
        "cardId":"219834",
        "currency":"PLN"
    },
    "receiver":{
        "userId": 2345,
        "currency": "PLN",
        "receiverType": "FRIEND_ID"
    }
}
```

Receiver.receiverType = BARE_CARD_NUMBER.

```
POST /mobile-api/calculate-commission HTTP/1.1
Content-Type: application/json
Authorization: Mobile bG9nzW46YWNrbWU=
Product-Name: TestProduct
Content-Length: 101
```

```
{
    "amount": 200078,
    "type": "RECEIVER",
    "sender":{
        "cardId":"219834",
        "currency":"PLN"
    },
    "receiver":{
        "card": ["5","4","9","5","9","8","4","1","7","9","0","8","2","6","4","5"],
        "currency": "PLN",
        "receiverType": "BARE_CARD_NUMBER"
    }
}
```

Request headers

Request body with header: *X-Encryption-Public-Key*

Type	Value	Constraints	Description
Authorization	Mobile bG9naW46YWNrbWU=	Required	Device token with "Mobile " prefix
Product-Name	TestProduct	Required	Application product name
Content-Type	application/x-jwe-encryption-body+json	Optional	Header must be present if the request body is encrypted using the JWE standard.
X-Encryption-Public-Key		Optional	Header must be present if the response body is to be encrypted using the JWE standard. Public key must be encoded Base64.

Request fields

Response

Error response - ERROR_VALIDATION.

```
HTTP/1.1 400 BAD REQUEST
Content-Type: application/json;charset=UTF-8
```

```
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "traceId": "{{traceId}}",
    "errorStatus": "ERROR_VALIDATION",
    "message": "Some fields are invalid",
    "data": [
        {
            "field": "{{field_name_from_request}}",
            "message": "{{message}}"
        }
    ]
}
```

Error response - ERROR_BAD_TOKEN.

```
HTTP/1.1 400 BAD REQUEST
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "traceId": "{{traceId}}",
    "errorStatus": "ERROR_BAD_TOKEN"
}
```

Error response - PRODUCT_NOT_FOUND.

```
HTTP/1.1 404 NOT FOUND
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
```

```
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "traceId": "{{traceId}}",
    "errorStatus": "PRODUCT_NOT_FOUND",
    "message": "Product by name {{product_name}} not found."
}
```

Error response - INTERNAL_SERVER_ERROR.

```
HTTP/1.1 500 INTERNAL SERVER ERROR
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "traceId": "{{traceId}}",
    "errorStatus": "INTERNAL_SERVER_ERROR"
}
```

Response fields

Errors

Encrypted response fields when sent header: *X-Encryption-Public-Key*

Http Status	Error Status	Description
400 - Bad Request	ERROR_VALIDATION	Some fields are invalid
400 - Bad Request	ERROR_BAD_TOKEN	Invalid authorization token

400 - Bad Request	CRYPTOGRAPHY_ERROR	Error decoding public key has sent in header: *X-Encryption-Public-Key*
400 - Bad Request	CRYPTOGRAPHY_ERROR	Error on decrypting request
400 - Bad Request	CRYPTOGRAPHY_ERROR	Error on encrypting response
400 - Bad Request	CRYPTOGRAPHY_ERROR	JWE encryption Key is invalid
400 - Bad Request	CRYPTOGRAPHY_ERROR	JWE payload is expired
400 - Bad Request	ERROR_WHILE_GETTING_COUNTRY_CODE	Could not get card country code
400 - Bad Request	ERROR_WHILE_GETTING_SENDER_COUNTRY_CODE	Could not get card country code for sender
400 - Bad Request	ERROR_WHILE_GETTING_RECEIVER_COUNTRY_CODE	Could not get card country code for receiver
400 - Bad Request	ERROR_SENDER_CARD_NOT_ACTIVE	Sender card is not active
400 - Bad Request	ERROR_RECEIVER_CARD_NOT_ACTIVE	Receiver card is not active
400 - Bad Request	UNKNOWN_ERROR	Unknown error
404 - Not Found	PRODUCT_NOT_FOUND	Product not found based on sent header: *Product-Name*
404 - Not Found	CANT_FIND_CARD	Not found card
404 - Not Found	FRIEND_NOT_EXISTS	Not found friend
500 - Internal Server Error	INTERNAL_SERVER_ERROR	Internal application error
500 - Internal Server Error	ERROR_ON_GETTING_DEFAULT_CARD	Error on getting card for friend
500 - Internal Server Error	FENIGE_ERROR	Fenige error

Examples

Send Money

Request body with header: *X-Encryption-Public-Key*.

This method is used to full MoneySend transaction (funding and payment).
Transfers can be make in any currency.
1 - User by selecting type = **SENDER** defines amount of funding in given currency.
This amount is collected from sender card in selected currency. 2 - User by selecting type = **RECEIVER** defines amount of payment in given currency.
This amount is transferred to receiver card in selected currency.

In case there’s need revaluation from one currency to another, system uses lowerRate for situation 1 and higherRate for situation 2. For more details about specific rates please refer to Currency Rate method.
This method adds friend to sender after successful transaction.

Additionally, you can perform full MoneySend transaction with externalAuthentication (see: ??? and Authentication)

Request

Receiver.receiverType = WALLET_CARD_ID.

```
POST /mobile-api/send-money HTTP/1.1
Content-Type: application/json
Authorization: Mobile bG9nzW46YWNrbWU=
Product-Name: TestProduct
Content-Length: 56

{
    "amount": 1000,
    "cvc2": ["1","2","3"],
    "type": "RECEIVER",
    "addressIp": "192.168.0.1",
    "sender": {
        "firstName": "Mark",
        "lastName": "Wards",
        "street": "Olszewskiego",
        "houseNumber": "17A",
        "city": "Lublin",
        "postalCode": "20-400",
        "flatNumber": "2",
        "email": "senderEmail@fenige.pl",
        "currency": "PLN",
        "expirationDate": "03/20",
        "personalId": "AGC688910",
        "cardId": "219708"
    },
    "receiver": {
        "firstName": "Rob",
        "lastName": "Wring",
```

```
        "currency": "PLN",
        "card": ["2","1","9","7","0","8"],
        "displayName": "Rob W.",
        "phoneNumber": "48718222333",
        "receiverType": "WALLET_CARD_ID",
        "userId": "13001"
    }
}
```

Receiver.receiverType = FRIEND_ID.

```
POST /mobile-api/send-money HTTP/1.1
Content-Type: application/json
Authorization: Mobile bG9nzW46YWNrbWU=
Product-Name: TestProduct
Content-Length: 56

{
    "amount": 1000,
    "cvc2": ["1","2","3"],
    "type": "RECEIVER",
    "addressIp": "192.168.0.1",
    "sender": {
        "firstName": "Mark",
        "lastName": "Wards",
        "street": "Olszewskiego",
        "houseNumber": "17A",
        "city": "Lublin",
        "postalCode": "20-400",
        "flatNumber": "2",
        "email": "senderEmail@fenige.pl",
        "currency": "PLN",
        "expirationDate": "03/20",
        "personalId": "AGC688910",
        "cardId": "219708"
    },
    "receiver": {
        "firstName": "Rob",
        "lastName": "Wring",
        "currency": "PLN",
```

```
        "displayName": "Rob W.",
        "receiverType": "FRIEND_ID",
        "userId": "123"
    }
}
```

Receiver.receiverType = BARE_CARD_NUMBER.

```
POST /mobile-api/send-money HTTP/1.1
Content-Type: application/json
Authorization: Mobile bG9nzW46YWNrbWU=
Product-Name: TestProduct
Content-Length: 56

{
    "amount": 1000,
    "cvc2": ["1","2","3"],
    "type": "RECEIVER",
    "addressIp": "192.168.0.1",
    "sender": {
        "firstName": "Mark",
        "lastName": "Wards",
        "street": "Olszewskiego",
        "houseNumber": "17A",
        "city": "Lublin",
        "postalCode": "20-400",
        "flatNumber": "2",
        "email": "senderEmail@fenige.pl",
        "currency": "PLN",
        "expirationDate": "03/20",
        "personalId": "AGC688910",
        "cardId": "219708"
    },
    "receiver": {
        "firstName": "Rob",
        "lastName": "Wring",
        "currency": "PLN",
        "card": ["5","1","4","2","3","3","3","6","2","9","5","2","3","7","3","2"],
        "displayName": "displayName",
        "phoneNumber": "48299000111",
```

```
        "receiverType": "BARE_CARD_NUMBER"
    }
}
```

ExternalAuthentication.authenticationId.

```
POST /mobile-api/send-money HTTP/1.1
Content-Type: application/json
Authorization: Mobile bG9nzW46YWNrbWU=
Product-Name: TestProduct
Content-Length: 56

{
  "amount" : 1000,
  "cvc2" : [ "1", "2", "3" ],
  "type" : "RECEIVER",
  "addressIp" : "192.168.0.1",
  "sender" : {
    "firstName" : "Mark",
    "lastName" : "Asdasd",
    "street" : "Olszewskiego",
    "houseNumber" : "17A",
    "city" : "Lublin",
    "postalCode" : "20-400",
    "flatNumber" : "2",
    "email" : "senderEmail@fenige.pl",
    "currency" : "PLN",
    "expirationDate" : "03/20",
    "personalId" : "AGC688910",
    "cardId" : "219708"
  },
  "receiver" : {
    "firstName" : "Rob",
    "lastName" : "Wring",
    "currency" : "PLN",
    "card" : [ "2", "1", "9", "7", "0", "8" ],
    "displayName" : "displayName",
    "phoneNumber" : "phoneNumber",
    "receiverType" : "WALLET_CARD_ID",
    "userId" : "123"
```

```
  },
  "externalAuthentication" : {
    "authenticationId" : "authenticationId"
  }
}
```

ExternalAuthentication.cavv, eci, transactionXId, authenticationStatus.

```
POST /mobile-api/send-money HTTP/1.1
Content-Type: application/json
Authorization: Mobile bG9nzW46YWNrbWU=
Product-Name: TestProduct
Content-Length: 56

{
  "amount" : 1000,
  "cvc2" : [ "1", "2", "3" ],
  "type" : "RECEIVER",
  "addressIp" : "192.168.0.1",
  "sender" : {
    "firstName" : "Mark",
    "lastName" : "Asdasd",
    "street" : "Olszewskiego",
    "houseNumber" : "17A",
    "city" : "Lublin",
    "postalCode" : "20-400",
    "flatNumber" : "2",
    "email" : "senderEmail@fenige.pl",
    "currency" : "PLN",
    "expirationDate" : "03/20",
    "personalId" : "AGC688910",
    "cardId" : "219708"
  },
  "receiver" : {
    "firstName" : "Rob",
    "lastName" : "Wring",
    "currency" : "PLN",
    "card" : [ "2", "1", "9", "7", "0", "8" ],
    "displayName" : "displayName",
    "phoneNumber" : "phoneNumber",
```

```
    "receiverType" : "WALLET_CARD_ID",
    "userId" : "123"
  },
  "externalAuthentication" : {
    "cavv" : "jEu04WZns7pbARAApU4qgNdJTag",
    "eci" : "PLN",
    "authenticationStatus" : "Y",
    "transactionXId" : "9742432a-dfdc-41ca-9ae9-b6595de65f1d"
  }
}
```

Request headers

Type	Value	Constraints	Description
Authorization	Mobile bG9naW46YWNrbWU=	Required	Device token with "Mobile " prefix
Product-Name	TestProduct	Required	Application product name
Content-Type	application/x-jwe-encryption-body+json	Optional	Header must be present if the request body is encrypted using the JWE standard.
X-Encryption-Public-Key		Optional	Header must be present if the response body is to be encrypted using the JWE standard. Public key must be encoded Base64.

Request fields

Response

Error response - ERROR_VALIDATION.

```
HTTP/1.1 400 BAD REQUEST
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
```

```
Expires: 0
X-Frame-Options: DENY

{
    "traceId": "{{traceId}}",
    "errorStatus": "ERROR_VALIDATION",
    "message": "Some fields are invalid",
    "data": [
        {
            "field": "{{field_name_from_request}}",
            "message": "{{message}}"
        }
    ]
}
```

Error response - ERROR_BAD_TOKEN.

```
HTTP/1.1 400 BAD REQUEST
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "traceId": "{{traceId}}",
    "errorStatus": "ERROR_BAD_TOKEN"
}
```

Error response - PRODUCT_NOT_FOUND.

```
HTTP/1.1 404 NOT FOUND
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
```

```
X-Frame-Options: DENY

{
    "traceId": "{{traceId}}",
    "errorStatus": "PRODUCT_NOT_FOUND",
    "message": "Product by name {{product_name}} not found."
}
```

Error response - INTERNAL_SERVER_ERROR.

```
HTTP/1.1 500 INTERNAL SERVER ERROR
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "traceId": "{{traceId}}",
    "errorStatus": "INTERNAL_SERVER_ERROR"
}
```

Response fields

Errors

Encrypted response fields when sent header: *X-Encryption-Public-Key*

Http Status	Error Status	Description
400 - Bad Request	ERROR_VALIDATION	Some fields are invalid
400 - Bad Request	ERROR_BAD_TOKEN	Invalid authorization token
400 - Bad Request	CRYPTOGRAPHY_ERROR	Error decoding public key has sent in header: *X-Encryption-Public-Key*
400 - Bad Request	CRYPTOGRAPHY_ERROR	Error on decrypting request
400 - Bad Request	CRYPTOGRAPHY_ERROR	Error on encrypting response
400 - Bad Request	CRYPTOGRAPHY_ERROR	JWE encryption Key is invalid

400 - Bad Request	CRYPTOGRAPHY_ERROR	JWE payload is expired
400 - Bad Request	ERROR_WHILE_GETTING_COUNTRY_CODE	Could not get card country code
400 - Bad Request	ERROR_MERCHANT_NOT_SUPPORT_CARD_PROVIDER	Merchant not support card provider
400 - Bad Request	ERROR_SENDER_CARD_NOT_ACTIVE	Sender card is not active
400 - Bad Request	ERROR_RECEIVER_CARD_NOT_ACTIVE	Receiver card is not active
400 - Bad Request	ERROR_SENDER_CARD_IS_BLOCKED	Sender card is blocked
400 - Bad Request	ERROR_RECEIVER_CARD_IS_BLOCKED	Receiver card is blocked
400 - Bad Request	UNKNOWN_ERROR	Unknown error
404 - Not Found	PRODUCT_NOT_FOUND	Product not found based on sent header: *Product-Name*
404 - Not Found	CANT_FIND_CARD	Not found card
404 - Not Found	FRIEND_NOT_EXISTS	Not found friend
500 - Internal Server Error	INTERNAL_SERVER_ERROR	Internal application error
500 - Internal Server Error	FENIGE_ERROR	Fenige error
500 - Internal Server Error	ERROR_ON_GETTING_DEFAULT_CARD	Error on getting card for friend

Examples

Add Friend

Request body with header: *X-Encryption-Public-Key*.

This method allow user to add Friend.

Request

friendType = WALLET.

```
POST /mobile-api/wallet-users/friends HTTP/1.1
Content-Type: application/json
Authorization: Mobile bG9nzW46YWNrbWU=
Product-Name: TestProduct
Content-Length: 56
```

```
{
  "friendWalletDataCoreId": 1,
  "displayName": "Display name",
  "phoneNumber": "48999111222",
  "friendType": "WALLET",
  "firstName": "First",
  "lastName": "Last",
}
```

friendType = EXTERNAL.

```
POST /mobile-api/wallet-users/friends HTTP/1.1
Content-Type: application/json
Authorization: Mobile bG9nzW46YWNrbWU=
Product-Name: TestProduct
Content-Length: 56

{
  "displayName": "Display name",
  "phoneNumber": "48999111222",
  "friendType": "EXTERNAL",
  "firstName": "First",
  "lastName": "Last",
  "cardNumber": ["5","5","2","7","4","7","9","6","6","8","3","9","0","9","5","7"]
}
```

Request headers

Request body with header: *X-Encryption-Public-Key*

Type	Value	Constraints	Description
Authorization	Mobile bG9naW46YWNrbWU=	Required	Device token with "Mobile " prefix
Product-Name	TestProduct	Required	Application product name
Content-Type	application/x-response-body+json	Optional	Header must be present if the response body must have body.

Content-Type	application/x-jwe-encryption-body+json	Optional	Header must be present if the request body is encrypted using the JWE standard.
X-Encryption-Public-Key		Optional	Header must be present if the response body is to be encrypted using the JWE standard. Public key must be encoded Base64.

Request fields

Response

Response fields

Examples

Get User friends

Request body with header: *X-Encryption-Public-Key*.

This method allow user to get all his friends

Request

Request headers

Encrypted request body with header: Content-Type: application/x-jwe-encryption-body+json

Type	Value	Constraints	Description
Authorization	Mobile bG9naW46YWNrbWU=	Required	Device token with "Mobile " prefix
Product-Name	TestProduct	Required	Application product name

X-Encryption-Public-Key		Optional	Header must be present if the response body is to be encrypted using the JWE standard. Public key must be encoded Base64.

Response

Response fields

Examples

Update Friend

Request body with header: *X-Encryption-Public-Key*.

This method allow user to update friend. For a friend of the type: WALLET, can update only the field: displayName. For a friend of the type: EXTERNAL, can update the fields: phoneNumber, displayName, firstName, lastName, cardNumber.

Request

friendType = WALLET.

```
PUT /mobile-api/wallet-users/friends/24 HTTP/1.1
Content-Type: application/json
Authorization: Mobile bG9naW46YWNrbWU=
Product-Name: TestProduct
Content-Length: 101

{
   "displayName":"Display name"
}
```

friendType = EXTERNAL.

```
PUT /mobile-api/wallet-users/friends/24 HTTP/1.1
Content-Type: application/json
Authorization: Mobile bG9naW46YWNrbWU=
Product-Name: TestProduct
Content-Length: 101

{
   "phoneNumber":"48999000111",
   "displayName":"Display name",
   "firstName":"First",
   "lastName":"Last",
   "cardNumber":["4","4","4","0","0","0","0","4","4","4","0","4","0"]
}
```

Request headers

Encrypted request body with header: *Content-Type: application/x-jwe-encryption-body+json*

Type	Value	Constraints	Description
Authorization	Mobile bG9naW46YWNrbWU=	Required	Device token with "Mobile " prefix
Product-Name	TestProduct	Required	Application product name
Content-Type	application/x-jwe-encryption-body+json	Optional	Header must be present if the request body is encrypted using the JWE standard.

Request fields

Response

Examples

Delete friend

Encrypted request body with header: *Content-Type: application/x-jwe-encryption-body+json*.

This method allow user to delete friend

Request

Request headers

Type	Value	Constraints	Description
Authorization	Mobile bG9naW46YWNrbWU=	Required	Device token with "Mobile " prefix
Product-Name	TestProduct	Required	Application product name

Response

Examples

Get publicKey

This method allow user to get publicKey

Request

Request headers

Type	Value	Constraints	Description
Authorization	Mobile bG9naW46YWNrbWU=	Required	Device token with "Mobile " prefix
Product-Name	TestProduct	Required	Application product name

Response

Response fields

Examples

MC Send

Every single method should contains Authorization and Mobile-Product headers.

Master Card Send

Methods allow sending money in MasterCard Send 2.0

Request

Sender.paymentAccountType = WALLET_CARD_ID.

```
POST /mobile-api/mc-send HTTP/1.1
Content-Type: application/json
Authorization: Mobile bG9naW46YWNrbWU=
Product-Name: TestProduct
Content-Length: 885

{
  "transactionId" : "bbb8597d-582c-4a12-a1c8-be9377aed6f9",
  "amount" : 40,
  "currency" : "INR",
  "sender" : {
    "account" : "walletCardId",
    "cvc2": ["3","2","1"],
    "addressId" : "123",
    "paymentAccountType" : "WALLET_CARD_ID"
```

```
  },
  "recipient" : {
    "name" : "Juniper Jane",
    "accountUri" : "402414000000006",
    "nationality" : "USA",
    "dateOfBirth" : "2011-05-13",
    "address" : {
      "city" : "Cape Girardeau",
      "country" : "USA",
      "state" : "MO",
      "postalCode" : "23232",
      "street" : "Mastercard Blvd"
    },
    "phone" : "1234567890",
    "email" : "jane.doe@mastercard.com",
    "governmentIds" : [ "123456789", "123456789" ],
    "receiverType" : "BARE_CARD_NUMBER"
  },
  "qrData" : "12",
  "transactionPurpose" : "07",
  "additionalMessage" : "message",
  "merchantCategoryCode" : "6536"
}
```

Sender.paymentAccountType = IBAN_ID.

```
POST /mobile-api/mc-send HTTP/1.1
Content-Type: application/json
Authorization: Mobile bG9naW46YWNrbWU=
Product-Name: TestProduct
Content-Length: 885

{
  "transactionId" : "bbb8597d-582c-4a12-a1c8-be9377aed6f9",
  "amount" : 40,
  "currency" : "INR",
  "sender" : {
    "account" : "ibanId",
    "addressId" : "123",
    "paymentAccountType" : "IBAN_ID"
```

```
  },
  "recipient" : {
    "name" : "Juniper Jane",
    "accountUri" : "402414000000006",
    "nationality" : "USA",
    "dateOfBirth" : "2011-05-13",
    "address" : {
      "city" : "Cape Girardeau",
      "country" : "USA",
      "state" : "MO",
      "postalCode" : "23232",
      "street" : "Mastercard Blvd"
    },
    "phone" : "1234567890",
    "email" : "jane.doe@mastercard.com",
    "governmentIds" : [ "123456789", "123456789" ],
    "receiverType" : "BARE_CARD_NUMBER"
  },
  "qrData" : "12",
  "transactionPurpose" : "07",
  "additionalMessage" : "message",
  "merchantCategoryCode" : "6536"
}
```

Recipient.receiverType = WALLET_CARD_ID.

```
POST /mobile-api/mc-send HTTP/1.1
Content-Type: application/json
Authorization: Mobile bG9naW46YWNrbWU=
Product-Name: TestProduct
Content-Length: 885

{
  "transactionId" : "bbb8597d-582c-4a12-a1c8-be9377aed6f9",
  "amount" : 40,
  "currency" : "INR",
  "sender" : {
    "account" : "ibanId",
    "addressId" : "123",
    "paymentAccountType" : "IBAN_ID"
```

```
  },
  "recipient" : {
    "name" : "Juniper Jane",
    "accountUri" : "4024",
    "nationality" : "USA",
    "dateOfBirth" : "2011-05-13",
    "address" : {
      "city" : "Cape Girardeau",
      "country" : "USA",
      "state" : "MO",
      "postalCode" : "23232",
      "street" : "Mastercard Blvd"
    },
    "phone" : "1234567890",
    "email" : "jane.doe@mastercard.com",
    "governmentIds" : [ "123456789", "123456789" ],
    "userId" : 13001,
    "receiverType" : "WALLET_CARD_ID"
  },
  "qrData" : "12",
  "transactionPurpose" : "07",
  "additionalMessage" : "message",
  "merchantCategoryCode" : "6536"
}
```

Recipient.receiverType = FRIEND_ID.

```
POST /mobile-api/mc-send HTTP/1.1
Content-Type: application/json
Authorization: Mobile bG9naW46YWNrbWU=
Product-Name: TestProduct
Content-Length: 885

{
  "transactionId" : "bbb8597d-582c-4a12-a1c8-be9377aed6f9",
  "amount" : 40,
  "currency" : "INR",
  "sender" : {
    "account" : "ibanId",
    "addressId" : "123",
```

```
    "paymentAccountType" : "IBAN_ID"
  },
  "recipient" : {
    "name" : "Juniper Jane",
    "nationality" : "USA",
    "dateOfBirth" : "2011-05-13",
    "address" : {
      "city" : "Cape Girardeau",
      "country" : "USA",
      "state" : "MO",
      "postalCode" : "23232",
      "street" : "Mastercard Blvd"
    },
    "phone" : "1234567890",
    "email" : "jane.doe@mastercard.com",
    "governmentIds" : [ "123456789", "123456789" ],
    "userId" : 13001,
    "receiverType" : "FRIEND_ID"
  },
  "qrData" : "12",
  "transactionPurpose" : "07",
  "additionalMessage" : "message",
  "merchantCategoryCode" : "6536"
}
```

Recipient.receiverType = BARE_CARD_NUMBER.

```
POST /mobile-api/mc-send HTTP/1.1
Content-Type: application/json
Authorization: Mobile bG9naW46YWNrbWU=
Product-Name: TestProduct
Content-Length: 885

{
  "transactionId" : "bbb8597d-582c-4a12-a1c8-be9377aed6f9",
  "amount" : 40,
  "currency" : "INR",
  "sender" : {
    "account" : "ibanId",
    "addressId" : "123",
```

```
    "paymentAccountType" : "IBAN_ID"
  },
  "recipient" : {
    "name" : "Juniper Jane",
    "accountUri" : "402414000000006",
    "nationality" : "USA",
    "dateOfBirth" : "2011-05-13",
    "address" : {
      "city" : "Cape Girardeau",
      "country" : "USA",
      "state" : "MO",
      "postalCode" : "23232",
      "street" : "Mastercard Blvd"
    },
    "phone" : "1234567890",
    "email" : "jane.doe@mastercard.com",
    "governmentIds" : [ "123456789", "123456789" ],
    "receiverType" : "BARE_CARD_NUMBER"
  },
  "qrData" : "12",
  "transactionPurpose" : "07",
  "additionalMessage" : "message",
  "merchantCategoryCode" : "6536"
}
```

Request headers

Request body with header: *X-Encryption-Public-Key*

Type	Value	Constraints	Description
Authorization	Mobile bG9naW46YWNrbWU=	Required	Device token with "Mobile " prefix
Product-Name	TestProduct	Required	Application product name
Content-Type	application/x-jwe-encryption-body+json	Optional	Header must be present if the request body is encrypted using the JWE standard.

X-Encryption-Public-Key		Optional	Header must be present if the response body is to be encrypted using the JWE standard. Public key must be encoded Base64.

Request fields

Response

errorStatus = INVALID_INPUT_FORMAT.

```
HTTP/1.1 400 BAD REQUEST
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "traceId": "b4ce7ad5-758d-444f-90b3-ffbadb757e3f",
    "errorStatus": "INVALID_INPUT_FORMAT",
    "message": "Invalid Format",
    "data": {
        "error": [
            {
                "source": "recipient.accountURI.Expiration date",
                "reasonCode": "INVALID_INPUT_FORMAT",
                "errorDetailCode": "062000",
                "description": "Invalid Format"
            }
        ]
    }
}
```

A formal table with Reason Code

Error Detail Code	Reason Code	Description

062000	INVALID_INPUT_FORMAT	Value contains invalid character
072000	INVALID_INPUT_LENGTH	Invalid length
082000	INVALID_INPUT_VALUE	Invalid value
092000	MISSING_REQUIRED_INPUT	Value is required
110501	RESOURCE_ERROR	Duplicate value
110503	RESOURCE_ERROR	Account not eligible
110505	RESOURCE_ERROR	Invalid currency
110507	RESOURCE_UNKNOWN	Record not found
110510	RESOURCE_ERROR	Invalid Request
110537	RESOURCE_ERROR	Value is not supported for the merchant
130004	DECLINE	Per transaction maximum amount limit reached
130006	DECLINE	Transaction Limit is less than the minimum configured for the partner
130010	DECLINE	Partner not onboarded for the network to reach the account

errorStatus = ERROR_BAD_TOKEN.

```
HTTP/1.1 400 BAD REQUEST
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "traceId": "{{traceId}}",
    "errorStatus": "ERROR_BAD_TOKEN"
}
```

errorStatus = CANT_FIND_PAYMENT_TOKEN.

```
HTTP/1.1 404 NOT FOUND
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
```

```
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "traceId": "89cdfc2b-346e-42d0-b20d-f3afa01cec68",
    "errorStatus": "CANT_FIND_PAYMENT_TOKEN",
    "message": "Payment token with given id was not found"
}
```

errorStatus = SYSTEM_ERROR.

```
HTTP/1.1 500 INTERNAL SERVER ERROR
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "traceId": "1c8d4f1f-16db-4c43-bdce-0fe43ae39195",
    "errorStatus": "SYSTEM_ERROR",
    "message": "Internal exception occurred.",
    "data": {
        "error": [
            {
                "source": "SYSTEM",
                "reasonCode": "SYSTEM_ERROR",
                "errorDetailCode": null,
                "description": "Internal exception occurred."
            }
        ]
    }
}
```

Response fields

Examples

Authentication

Every single method should contains Authorization and Mobile-Product headers.

Init Authentication

The authentication stage flow is indicated by the following field: threeDsMode

Method allows us to do initialize authentication using ThreeDs 2.0 protocol.

After this method you have 3 options:

- **FRICTIONLESS** - In response: authenticationStatus, transactionXId, cavv, eci and **threeDsMode = FRICTIONLESS** are present. This response denotes that authentication was finished.
- **ThreeDsMethod flow** - In response: threeDsMethodData and **threeDsMode = THREE_DS_METHOD** are present. This response denotes that you should perform ThreeDs method flow. After executing ThreeDs method flow, make a request for the method: Continue Authentication
- **CHALLENGE** - In response: acsUrl, creq, challengeHtmlFormBase64 and **threeDsMode = CHALLENGE** are present. This response denotes that you should perform challenge. After executing challenge, make a request for the method: Finalize Authentication

Request

Request headers

Request body with header: *X-Encryption-Public-Key*

Type	Value	Constraints	Description

Authorization	Mobile bG9naW46YWNrbWU=	Required	Device token with "Mobile " prefix
Product-Name	TestProduct	Required	Application product name
Content-Type	application/x-jwe-encryption-body+json	Optional	Header must be present if the request body is encrypted using the JWE standard.
X-Encryption-Public-Key		Optional	Header must be present if the response body is to be encrypted using the JWE standard. Public key must be encoded Base64.

Request fields

Response

threeDsMode = FRICTIONLESS.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "authenticationId": "authenticationId",
    "authenticationStatus": "Y",
    "transactionXId": "9742432a-dfdc-41ca-9ae9-b6595de65f1d",
    "cavv": "jEu04WZns7pbARAApU4qgNdJTag",
    "eci": "02",
    "threeDsMode": "FRICTIONLESS"
}
```

threeDsMode = THREE_DS_METHOD.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "authenticationId": "authenticationId",
    "threeDsMethodData":
"eyJ0aHJlZURWZpY2F0aW9uVVJMIjoiaHR0cHM6Ly93ZWJob29rLnNpdGUvc3MiLCJ0aHJlZURTU2VydmVyVHJhbnNJRCI
6IjNmYWYwZjFZi1iYjQyLThkN2RhM2M0NjY5OSJ9",
    "threeDsMethodUrl": "https://threeDsMethodUrl-test.verestro.com/acs-mock",
    "threeDsMode": "THREE_DS_METHOD"
}
```

threeDsMode = CHALLENGE.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "authenticationId": "authenticationId",
    "acsUrl": "https://acs-url.verestro.com/mock-acs",
    "creq":
"eyJjYXJkQXV0aGVudGljYNmODlhLTk2MjQtNGQ1OS04NzZmLTNkMWViYTcyNzM3NiIsIm5vdGlmaWNhdGlvblVybvd2Vi
aG9vay5zaXRlLzE5ODI3MWMyLTljYWYtNGEyMy05ZGJiLWRlZTc3ODExMDdlOSIsInRocmVlRFNTZXJ2ZXJUcmFuc0lEIj
oiM2ZhZjBmMWQtM2YxNy00MTJmLWJiNDItOGQ3ZGEzYzQ2Njk5IiwibWVzc2FnZVZlcnNpb24iOiIyLjEuMCJ9",
    "challengeHtmlFormBase64":
"PGh0bWw+PFNDUklQVCBMQU5mF2YXNjcmlwdCI+ZnVuY3Rpb24gT25Mb2FkRXZlbW1lbnQuZG93bmxvYWRGb3JtLnN1Ym1
pdCgpOyB9PC9TQ1JJUFQ+PGJvZHkgT25Mb2FkmVudCgpOyI+PGZvcm0gbmFtZT0iZG93bmxvYWRGb3JtIiBhY3Rpb249Im
h0dHBzOi8vbXBpLXN0YWdpbmcuZmVuaWdlLnBsL21vY2stYWNzIiBtZXRob2Q9IlBPU1QiPjxJTlBVVCB0eXBlPSJoaWRk
```

```
ZW4iXEiIHZhbHVlPSJleUpqWVhKa1FYVjBhR1Z1ZEdsallYUnBiMjVKWkNJNkltRmpZbU5tT0RsaExUazJNalF0TkdRMU9
TMDROelptTFROa01XVmlZVGN5TnpNM05pSXNJbTV2ZEdsbWFXTmhkR2x2YmxWeWJDSTZJbWgwZEhCek9pOHZkMlZpYUc5d
mF5NXphWFJsTHpFNU9ESTNNV015TFRsallXWXROR0V5TXkwNVpHSmlMV1JsWlRjM09ERXhNRGRsT1NJc0luUm9jbVZsUkZ
OVFpYSjJaWEpVY21GdWMwbEVJam9pTTJaaFpqQm1NV1F0TTJZeE55MDBNVEptTFdKaU5ESXRPR1EzWkdFell6UTJOams1S
Wl3aWJXVnpjMkZuWlZabGNuTnBiMjRpT2lJeUxqRXVNQ0o5Ij48SU5QVVQgdHlwZT0iaGlkZGVuIiBuYW1lPSJ0aHJlZUR
TU2Vzc2lvbkRhdGEiIHZhbHVlPSJZV05pWTJZNE9XRXRPVFl5TkMwMFpEVTVMVGczTm1ZdE0yUXhaV0poTnpJM016YzIiP
jwvZm9ybT48L2JvZHk+PC9odG1sPg==",
    "threeDsSessionData": "YWNiY2Y4OWEtONC00ZDU5LTg3NmYtM2QxZWJhNzI3Mzc2",
    "threeDsMode": "CHALLENGE"
}
```


Response fields

Base response fields

Path	Type	Description
authenticationId	String	Unique authentication identifier
threeDsMethodData	String	Encoded data used for request to ACS
threeDsMethodUrl	String	ACS endpoint for hidden request. If endpoint is not present then request is not required.

authenticationStatus	String	Indicates whether a transaction qualifies as an authenticated transaction or account verification. Possible values are: **Y** - Authentication/account verification successful **N** - Not authenticated/account not verified; transaction denied **U** - Authentication/account verification could not be performed; technical or other problem as indicated in ARes or RReq **A** - Attempts processing performed; not authenticated/verified, but a proof of attempted authentication/verification is provided **C** - Challenge required; additional authentication is required using the CReq/CRes **R** - Authentication/account verification rejected; issuer is rejecting authentication/verification and request that authorization not be attempted **D** - Challenge required; decoupled authentication confirmed **I** - Informational only; ThreeDs Requestor challenge preference acknowledged The CRes message can contain only a value of Y or N. Values of D and I are only applicable for ThreeDs version 2.2.0.
transactionXId	String	This field indicates the transactionXid from recurring initial authentication.
cavv	String	This property is determined by the Access Control Server. This property will be valid if the TransactionStatus is "Y" or "A". The value may be used to provide proof of authentication.
eci	String	This property is determined by the Access Control Server. This property contains the two digit Electronic Commerce Indicator (ECI) value, which is to be submitted in a credit card authorization message. This value indicates to the processor that the customer data in the authorization message has been authenticated. The data contained within this property is only valid if the TransactionStatus is "Y" or "A".

acsUrl	String	If challenge is required, data for building a form such as challengeHtmlFormBase64
creq	String	If challenge is required, data for building a form such as challengeHtmlFormBase64
challengeHtmlFormBase64	String	This field is a BASE64 encrypted html source file containing the challenge 3-D Secure frame
threeDsSessionData	String	ThreeDsSessionData value
threeDsMode	String	ThreeDs process mode which informs about. One of: [FRICTIONLESS, THREE_DS_METHOD, CHALLENGE] **FRICTIONLESS** - this is where the authentication process was finished. **THREE_DS_METHOD** - next step is to execute the ThreeDs method process. After it is done, we need to make a request to the method: Continue Authentication **CHALLENGE** - next step is to execute the challenge process. After it is done, we need to make a request to the method: Finalize Authentication

Examples

Errors

Request body with header: *X-Encryption-Public-Key*

Http Status	Error Status	Description
400 - Bad Request	PROCESS_NOT_ALLOWED	Method not allowed - invoke calculate commission method is necessary first.
400 - Bad Request	ERROR_SENDER_CARD_NOT_ACTIVE	Sender card is not active

Continue Authentication

The authentication stage flow is indicated by the following field: threeDsMode

Method allows us to do continue authentication using ThreeDs 2.0 protocol. Use this method after perform process ThreeDsMethod. This step is optional in the authentication process. Required only

if ThreeDsMethod case is present.

After this method you have 2 options:

- **FRICTIONLESS** - In response: authenticationStatus, transactionXId, cavv, eci and **threeDsMode = FRICTIONLESS** are present. This response denotes that authentication was finished.
- **CHALLENGE** - In response: acsUrl, creq, challengeHtmlFormBase64 and **threeDsMode = CHALLENGE** are present. This response denotes that you should perform challenge. After executing challenge, make a request for the method: Finalize Authentication

Request

Request headers

Request body with header: *X-Encryption-Public-Key*

Type	Value	Constraints	Description
Authorization	Mobile bG9naW46YWNrbWU=	Required	Device token with "Mobile " prefix
Product-Name	TestProduct	Required	Application product name
Content-Type	application/x-jwe-encryption-body+json	Optional	Header must be present if the request body is encrypted using the JWE standard.
X-Encryption-Public-Key		Optional	Header must be present if the response body is to be encrypted using the JWE standard. Public key must be encoded Base64.

Request fields

Response

threeDsMode = FRICTIONLESS.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
```

```
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "authenticationId": "authenticationId",
    "authenticationStatus": "Y",
    "transactionXId": "9742432a-dfdc-41ca-9ae9-b6595de65f1d",
    "cavv": "jEu04WZns7pbARAApU4qgNdJTag",
    "eci": "02",
    "threeDsMode": "FRICTIONLESS"
}
```

threeDsMode = CHALLENGE.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY

{
    "authenticationId": "authenticationId",
    "acsUrl": "https://acs-url.verestro.com/mock-acs",
    "creq":
"eyJjYXJkQXV0aGVudGljYNmODlhLTk2MjQtNGQ1OS04NzZmLTNkMWViYTcyNzM3NiIsIm5vdGlmaWNhdGlvblVybvd2Vi
aG9vay5zaXRlLzE5ODI3MWMyLTljYWYtNGEyMy05ZGJiLWRlZTc3ODExMDdlOSIsInRocmVlRFNTZXJ2ZXJUcmFuc0lEIj
oiM2ZhZjBmMWQtM2YxNy00MTJmLWJiNDItOGQ3ZGEzYzQ2Njk5IiwibWVzc2FnZVZlcnNpb24iOiIyLjEuMCJ9",
    "challengeHtmlFormBase64":
"PGh0bWw+PFNDUklQVCBMQU5mF2YXNjcmlwdCI+ZnVuY3Rpb24gT25Mb2FkRXZlbW1lbnQuZG93bmxvYWRGb3JtLnN1Ym1
pdCgpOyB9PC9TQ1JJUFQ+PGJvZHkgT25Mb2FkmVudCgpOyI+PGZvcm0gbmFtZT0iZG93bmxvYWRGb3JtIiBhY3Rpb249Im
h0dHBzOi8vbXBpLXN0YWdpbmcuZmVuaWdlLnBsL21vY2stYWNzIiBtZXRob2Q9IlBPU1QiPjxJTlBVVCB0eXBlPSJoaWRk
ZW4iXEiIHZhbHVlPSJleUpqWVhKa1FYVjBhR1Z1ZEdsallUnBiMjVKWkNJNkltRmpZbU5tT0RsaExUazJNalF0TkdRMU9
TMDROelptTFROa01XVmlZVGN5TnpNM05pSXNJbTV2ZEdsbWFXTmhkR2x2YmxWeWJDSTZJbWgwZEhCek9pOHZkMlZpYUc5d
mF5NXphWFJsTHpFNU9ESTNNV015TFRsallXWXROR0V5TXkwNVpHSmlMV1JsWlRjM09ERXhNRGRsT1NJc0luUm9jbVZsUkZ
```

```
OVFpYSjJaWEpVY21GdWMwbEVJam9pTTJaaFpqQm1NV1F0TTJZeE55MDBNVEptTFdKaU5ESXRPR1EzWkdFell6UTJOams1S
Wl3aWJXVnpjMkZuWlZabGNuTnBiMjRpT2lJeUxqRXVNQ0o5Ij48SU5QVVQgdHlwZT0iaGlkZGVuIiBuYW1lPSJ0aHJlZUR
TU2Vzc2lvbkRhdGEiIHZhbHVlPSJZV05pWTJZNE9XRXRPVFl5TkMwMFpEVTVMVGczTm1ZdE0yUXhaV0poTnpJM016YzIiP
jwvZm9ybT48L2JvZHk+PC9odG1sPg==",
    "threeDsSessionData": "YWNiY2Y4OWEtONC00ZDU5LTg3NmYtM2QxZWJhNzI3Mzc2",
    "threeDsMode": "CHALLENGE"
}
```


Response fields

Base response fields

Path	Type	Description
authenticationId	String	Unique authentication identifier
authenticationStatus	String	Indicates whether a transaction qualifies as an authenticated transaction or account verification. Possible values are: **Y** - Authentication/account verification successful **N** - Not authenticated/account not verified; transaction denied **U** - Authentication/account verification could not be performed; technical or other problem as indicated in ARes or RReq **A** - Attempts processing performed; not authenticated/verified, but a proof of attempted authentication/verification is provided **C** - Challenge required; additional authentication is required using the CReq/CRes **R** - Authentication/account verification rejected; issuer is rejecting authentication/verification and request that authorization not be attempted **D** - Challenge required; decoupled authentication confirmed **I** - Informational only; ThreeDs Requestor challenge preference acknowledged The CRes message can contain only a value of Y or N. Values of D and I are only applicable for ThreeDs version 2.2.0.
transactionXId	String	This field indicates the transactionXid from recurring initial authentication.

cavv	String	This property is determined by the Access Control Server. This property will be valid if the TransactionStatus is "Y" or "A". The value may be used to provide proof of authentication.
eci	String	This property is determined by the Access Control Server. This property contains the two digit Electronic Commerce Indicator (ECI) value, which is to be submitted in a credit card authorization message. This value indicates to the processor that the customer data in the authorization message has been authenticated. The data contained within this property is only valid if the TransactionStatus is "Y" or "A".
acsUrl	String	If challenge is required, data for building a form such as challengeHtmlFormBase64
creq	String	If challenge is required, data for building a form such as challengeHtmlFormBase64
challengeHtmlFormBase64	String	This field is a BASE64 encrypted html source file containing the challenge 3-D Secure frame
threeDsSessionData	String	ThreeDsSessionData value
threeDsMode	String	ThreeDs process mode which informs about. One of: [FRICTIONLESS, CHALLENGE] **FRICTIONLESS** - this is where the authentication process was finished. **CHALLENGE** - next step is to execute the challenge process. After it is done, we need to make a request to the method: Finalize Authentication

Examples

Finalize Authentication

Request body with header: *X-Encryption-Public-Key*.

Method allows us to do finalize authentication using ThreeDs 2.0 protocol.

Request

Request headers

Request body with header: *X-Encryption-Public-Key*

Type	Value	Constraints	Description
Authorization	Mobile bG9naW46YWNrbWU=	Required	Device token with "Mobile " prefix
Product-Name	TestProduct	Required	Application product name
Content-Type	application/x-jwe-encryption-body+json	Optional	Header must be present if the request body is encrypted using the JWE standard.
X-Encryption-Public-Key		Optional	Header must be present if the response body is to be encrypted using the JWE standard. Public key must be encoded Base64.

Request fields

Response

Response fields

Base response fields

Path	Type	Description
authenticationId	String	Unique authentication identifier

authenticationStatus	String	Indicates whether a transaction qualifies as an authenticated transaction or account verification. Possible values are: **Y** - Authentication/account verification successful **N** - Not authenticated/account not verified; transaction denied **U** - Authentication/account verification could not be performed; technical or other problem as indicated in ARes or RReq **A** - Attempts processing performed; not authenticated/verified, but a proof of attempted authentication/verification is provided **C** - Challenge required; additional authentication is required using the CReq/CRes **R** - Authentication/account verification rejected; issuer is rejecting authentication/verification and request that authorization not be attempted **D** - Challenge required; decoupled authentication confirmed **I** - Informational only; ThreeDs Requestor challenge preference acknowledged The CRes message can contain only a value of Y or N. Values of D and I are only applicable for ThreeDs version 2.2.0.
transactionXId	String	This field indicates the transactionXid from recurring initial authentication.
cavv	String	This property is determined by the Access Control Server. This property will be valid if the TransactionStatus is "Y" or "A". The value may be used to provide proof of authentication.
eci	String	This property is determined by the Access Control Server. This property contains the two digit Electronic Commerce Indicator (ECI) value, which is to be submitted in a credit card authorization message. This value indicates to the processor that the customer data in the authorization message has been authenticated. The data contained within this property is only valid if the TransactionStatus is "Y" or "A".

Examples

Revision #1
Created 21 March 2023 08:07:38 by Mateusz Rosa
Updated 21 March 2023 08:33:07 by Mateusz Rosa