CIB BANK

Electronic Card Transactions on the Internet

CIB Bank Zrt.

eCommerce

Internet-based Card Acquiring Service

Technical Documentation

Developers guide

CIB BANK

Contents

CIB BANK

Preamble

This description is for developers who design payment modules for webshops, which are suitable for the usage the API provided by CIB Bank.
The description is of a professional nature, and it assumes the knowledge of the programming language used for its implementation.

Before reading this document, please read the reference description, where the steps referred to herein are explained in detail.

The concrete examples included in the description are written in PHP language, since, based on our experience, this language has been chosen most frequently for implementing web applications. In these examples the same key is used for encryption as in the sample applications.

CIB BANK

The payment process

Preparation

Customer identification

The customer must be identified prior to starting the bank transaction. Identification may be executed in the generally accepted manner, provided that it complies with the following conditions:

- The customer can be clearly identified
- The customer's notification channel/address can be determined

Identification is executed by the webshop, in which case the transaction receives an internal identifier suitable for identifying either the customer or the session opened by the customer. This internal identifier must be linked to the identifier of the bank transaction (see the following section).

Generating the transaction data

Based on the already identified customer, the related basket and the data provided by the bank at the time of contract conclusion the parameters necessary for the bank transaction are to be generated.

PID

This is provided by the bank to the merchant at the time of contract conclusion. The first 3 characters correspond to the name of the encryption key, and the last four figures identify the virtual POS terminal. The value in the examples used in this document is 'IEB0001', although under no circumstances can this identifier be the same as the one that can be used in the bank test and on the live server.

TRID

A unique identifier generated by the webshop. This is a 16-digit random number that identifies the transaction throughout its lifecycle (authorisation, withdrawal of authorisation or refund). Since the value of the TRID will be necessary during a possible future transaction analysis, please keep it for at least one year in each case. If the webshop also uses its own identifier, you should link together the two values.
Example of generation:

```
mt_srand();
$trid="";
for ($i=0; $i<4; $i++)
    $trid .= mt_rand(1000, 9999);
```

Considering that the identifier is unique, the final value must be generated until it yields a value that has never been used with any of the transactions.

UID

Although the value of the customer ID is irrelevant from the aspect of the bank message, it should be linked to the webshop's customer ID (keeping in mind that in the bank message this cannot be more than 11 characters long).

CUR

The currency of the amount to be paid in the transaction in a 3-letter format complying with the ISO3166 standard. Currently, the values of this field may be as follows:

- EUR
- HUF

AMO

For Hungarian forint the transaction amount must be an integer number, in the case of euro it must be a fraction rounded to two decimals. Examples of amounts:

```
$cur = "EUR";
$orig_amo = 100;
switch ($cur)
{
    case "EUR":
        $amo = number_format($orig_amo, 2, ".", "");
        break;
    case "HUF":
        $amo = number_format($orig_amo, 0, ".", "");
        break;
    default:
        $amo = 0;  // NOTE: this should not happen
}
```

Note: although the authorisation message sent to the customer's issuer bank contains the same amount as the one indicated in the AMO value, the value deducted from the account may be different, due to any conversion operations (e.g. the customer pays with a USD card).

TS

The system time of the hardware that runs the web server, in YYYYMMDDHHMMSS format:

- YYYY: calendar year, including the century (1970-2050)
- MM: month (01-12)
- DD: day (01-31)
- HH: hour (00-23)
- MM: minutes (00-59)
- SS: seconds (00-59)

It is not a requirement that the system time be synchronised with an atomic clock, but the use of another format may result in unsuccessful processing. Examples of timestamps:

```
$ts = date("YmdHis");
```

AUTH

Type of authorisation. In case of web-based transactions its value must be '0'.

LANG

When the customer is redirected, this is the language of text on the payment page in the customer's browser. The accepted values are specified in the reference description. The use of a language code other than those specified in the description may result in incorrect processing.

URL

The customer must always be directed to the bank's payment page for entering the card data. After authorisation, the customer is returned to the address provided in this parameter. The URL must contain an absolute path address with an http or https protocol. It must not contain any parameters (the customer's session value can be easily retrieved based on the TRID value that is mandatorily present in all the messages). The value of the URL must not be URL-encoded prior to encryption.

CIB BANK

Authorisation

Transaction initialisation

Using the required parameters after generation and encryption, the bank's *merchant* URL must be called. Please note that you can only send the parameters that are necessary for the message, and any deviation from this rule may result in incorrect processing. The RC value in the bank's decrypted message indicates whether the message has been successfully processed. If the message sent to the bank cannot be interpreted at all, the bank gives an unencrypted error code that indicates the cause of the error. Example of initialisation (for the interpretation of eki_* functions see sections Encryption or Communication):

```
$try = 0;
do {
    generateparams(); $try++
    $msgt10clear =  "PID=".$pid."&TRID=".$trid."&MSGT=10&UID=".$uid;
    $msgt10clear .=  "&AMO=".$amo."&CUR=".$cur."&TS=".$ts;
    $msgt10clear .=  "&AUTH=0&LANG=".$lang."&URL=".$url;
    $msgt10crypt = eki_encrypt($msgt10clear);
    $msgt11crypt = eki_call($eki_merchanturl, $msgt10crypt);
    $msgt11clear = eki_decrypt($msgt11crypt);
    parse_str($msgt11clear, $msgt11arr);
    $msgt11rc = $msgt11arr['RC'];
} while (($msgt11rc == "02") && ($try <= 3));
```

Based on the example, outside the loop, the possible values of $msgt11rc and the related interpretations are as follows:

- 00: successful initialisation
- 01: initialisation has failed due to other technical reasons
- 02: initialisation has failed because the generated TRID is reserved

An unencrypted error code returned by the bank, if any, may refer to a general error, the correction of which is described in the section entitled Error management.

If the value of $msgt11rc is '01', the transaction **must be interrupted,** and the customer must be informed of the failure of the payment. In the case of a '02' code initialisation should be repeated with a new TRID value. In the case of a successful initialisation ($msgt11rc='00') the TRID value must be saved, because it is to be used in all the subsequent messages related to the transaction.

Redirecting the customer to the payment page

After successful initialisation the customer's browser must be redirected to the bank's *customer* URL. Any method may be used for executing the redirection, its only purpose being to ensure that the bank's payment page (and nothing else) appears in the customer's browser. The parameter of the bank's URL is the (no. 20) message that is necessary for an encrypted redirection. Example:

```
$msgt20clear =  "PID=".$pid."&TRID=".$trid."&MSGT=20";
$msgt20crypt = eki_encrypt($msgt20clear);
header('Location: $eki_customerurl?$msgt20crypt');
```

As a result of this, the web server loses connection with the customer, who then returns to the bank's payment page to enter the card information.

Managing the customer's return

After the card information has been entered, the bank redirects the customer to the address indicated in the URL parameter provided in the initialisation message. Together with the redirection, a GET parameter also arrives with the bank's message containing the decryption key followed by the transaction ID. Based on this, the webshop's session can be assigned again to the bank transaction.

```
$msgt21clear = eki_decrypt(rawurlencode($_SERVER['QUERY_STRING']));
parse_str($msgt21clear, $msgt21arr);
$trid = $msgt21arr['TRID'];
```

If the session cannot be identified based on the received $trid value (assuming that these two values have previously been saved), the transaction must not be continued, and the customer must be informed of the error.

Customer information

If the customer successfully returns to the webshop, he/she must be informed on the result of the transaction there. The retrieval of the result and the closing of the transaction (MSGT32) must be implemented similarly to the initialisation, as a server-to-server message.

```
$msgt32clear =  "PID=".$pid."&TRID=".$trid."&MSGT=32&AMO=".$amo;
$msgt32crypt = eki_encrypt($msgt32clear);
$msgt31crypt = eki_call($eki_merchanturl, $msgt32crypt);
$msgt31clear = eki_decrypt($msgt31crypt);
parse_str($msgt31clear, $msgt31arr);
$msgt31rc = $msgt31arr['RC'];
$msgt31rt = $msgt31arr['RT'];
$msgt31anum = $msgt31arr['ANUM'];
```

The following parameters must be indicated on the confirmation page:

- Transaction ID (TRID)
- Amount (AMO)
- Currency (CUR)
- Result Code (RC)
- Result Text (RT)
- Authorisation code (ANUM)

The customer should be reminded to save the above data, and the data must be transferred to the customer in an alternative way (by e-mail or sms).
Considering that the customer interrupts all contact with the webshop for the period of the payment, it is possible that he/she does not return at all to the webshop (typically for technical reasons). The payment can be completed nevertheless, but if the webshop does not close the result within a predefined time interval (by default within 10-15 minutes) the bank *reverses* the transaction (releases the reservation on the customer's card). If the webshop requests the result of the transaction prior to the completion of the payment, the bank's server sends an error message. In order to solve the timing problem resulting from the above, the bank provides an inquiry message (MSGT33) which, apart from the card number, is fully identical to the MSGT32 message, but it provides an RC value in any case. This message must be applied in parallel to the traditional payment algorithm, using it once every minute after the customer's redirection to the bank. Interpretation of the RC codes provided in the response to the MSGT33 message:

- PR: the payment has not been completed, but it has not been interrupted either due to time-out
- TO: the payment has been interrupted due to time-out
- 00: the payment has been successfully completed
- Anything else: the payment has ended unsuccessfully. The most frequently occurring error codes and their possible reasons are listed in Annex 3, irrespective of which, the application must be made suitable for handling any other error codes, by creating a general error branch that closes the transaction.

The sending of MSGT33 messages must continue until the RC value of the returning message is 'PR'. If the RC value of the received MSGT31 response message is '00', the webshop may send a closing message (MSGT32), thus finalising the successful payment. In this case the bank does not release the reservation, regardless of whether the specified time interval has elapsed (just as if the customer had successfully returned to the webshop).

If the customer only returns to the webshop after the closure, he/she must be informed based on the already known data, and a new closing is not possible.

The webshop may, after returning the customer to the bank, retrieve all the previous steps of the transaction, in addition to the current status of the payment:

```
$msgt37clear =  "PID=".$pid."&TRID=".$trid."&MSGT=37&AMO=".$amo;
$msgt37crypt = eki_encrypt($msgt37clear);
$msgt38crypt = eki_call($eki_merchanturl, $msgt37crypt);
$msgt38clear = eki_decrypt($msgt38crypt);
parse_str($msgt38clear, $msgt38arr);
$msgt38rc = $msgt38arr['RC'];
$msgt38hist = $msgt38arr['HISTORY'];
```

The 'RC' value in the block indicates whether the message has been successfully executed (00: successful, 01: an error occurred), in case of successful execution, the string in the 'HISTORY' element contains the individual statuses, in their chronological order. A list of possible values is contained in the reference description (3.1 Field definitions, HISTORY). The message also signals an error even if the transaction has been initiated but none of the statuses indicated in the above description has occurred (e.g. the customer has not yet been directed to the payment page).

CIB BANK

Authorisation withdrawal

A successful and closed reservation (the RC value of a MSGT31 message sent in response to a MSGT32 message is ’00’) may also be released by the webshop. The effect of this is the same as when the bank does so after a time-out period. Prior to the release it is necessary to check whether the transaction can be actually released or not. This can be achieved using the MSGT70 message, which supplements the transaction authorisation result with the current status of the settlement (MSGT71 STATUS field). If the status is ’10’ (reserved but not yet debited), the transaction can be reversed using the MSGT74 message. In case of a successful reversal the value of the STATUS field in the response message (MSGT75) is ’40’, after which the MSGT71 message will provide the same value. A withdrawn reservation cannot be debited.

```
if ($msgt31rc = $msgt31arr['RC']) {
   $msgt70clear =  "PID=".$pid."&TRID=".$trid."&MSGT=70&AMO=".$amo;
   $msgt70crypt = eki_encrypt($msgt70clear);
   $msgt71crypt = eki_call($eki_merchanturl, $msgt70crypt);
   $msgt71clear = eki_decrypt($msgt71crypt);
   parse_str($msgt71clear, $msgt71arr);
   if ($msgt71arr['STATUS'] == "10") {
          $msgt74clear =  "PID=".$pid."&TRID=".$trid."&MSGT=74&AMO=".$amo;
          $msgt74crypt = eki_encrypt($msgt74clear);
          $msgt75crypt = eki_call($eki_merchanturl, $msgt74crypt);
          $msgt75clear = eki_decrypt($msgt75crypt);
   }
}
```

CIB BANK

Refund

This API message provides a wayto refund an already debited transactions, either partially or in full, within one calendar year. The minimum value that can be set as amount to be retransferred is EUR 1 or HUF 100. If the original debit was executed with an amount lower than this, no refund is permitted for that specific transaction. After setting the amount to be refunded (MSGT80) and requesting a refund (MSGT78) the bank refunds the set amount to the customer's card. By default (ignoring the MSGT80 message) the operation is executed with 0 amount, and therefore the use of the MSGT80 message prior to the refund is strongly recommended.

```
if ($msgt31rc = $msgt31arr['RC']) {
   $msgt70clear =  "PID=".$pid."&TRID=".$trid."&MSGT=70&AMO=".$amo;
   $msgt70crypt = eki_encrypt($msgt70clear);
   $msgt71crypt = eki_call($eki_merchanturl, $msgt70crypt);
   $msgt71clear = eki_decrypt($msgt71crypt);
   parse_str($msgt71clear, $msgt71arr);
   $status = $msgt71arr['STATUS'];
   if (($status == "20") || ($status == "30")) {
          $msgt80clear =  "PID=".$pid."&TRID=".$trid;
          $msgt80clear .= ."&MSGT=80&AMOORIG=".$amo."&AMONEW=".$amonew;
          $msgt80crypt = eki_encrypt($msgt80clear);
          $msgt81crypt = eki_call($eki_merchanturl, $msgt80crypt);
          $msgt81clear = eki_decrypt($msgt81crypt);
          $msgt78clear =  "PID=".$pid."&TRID=".$trid;
          $msgt78clear .= ."&MSGT=78&AMO=".$amo;
          $msgt78crypt = eki_encrypt($msgt78clear);
          $msgt79crypt = eki_call($eki_merchanturl, $msgt78crypt);
          $msgt79clear = eki_decrypt($msgt79crypt);
   }
}
```

The above example retrieves the transaction status, and if the status is 'debited' (in $amo value), it sets the amount to be refunded ($amonew), then launches a refund. In this case the status of the refunded transaction changes to '50'. Although it is not possible to initiate a new refund for a previously refunded transaction, the amount to be refunded may be modified several times prior to the refund.

CIB BANK

Encryption

All the messages sent to the bank must be encrypted prior to sending. The steps of the encryption are described in the reference manual, in addition to which an encryption sample application written in JAVA, PHP and C# languages is attached as an annex to this documentation. A list of third-party the libraries necessary for translating the sample applications is contained in the comment part of each source.

In addition to the algorithm, encryption also requires an encryption key, which is in each case provided to the merchant by the bank. The key names are always identical to the first 3 characters in the value of the PID parameter to be used in the individual messages. The version with a .reg extension is recommended for Windows-based systems. After registration, the key will be stored under the \HKLM\Software\IEB\Eki key, the key name is the same as the file name, and the only value (des) contains the full key file in binary format. The use of the file with a .des extension is recommended to webshops using a non-Windows system, and its binary content is fully identical to the value that can be stored in the registry. The encryption key is in the last 24 byte of the file in key1, key2, iv order (the keys and the initialisation vector are 8 byte long each). Encryption is to be implemented applying the 3DES CBC method. Example of encryption (it is to be assumed that the key file is located in the current library):

```
function eki_encrypt($cleartext) {
     $arr=split("&",$cleartext);
     $ciphertext="";
     $pid="";
     for ($i=0;$i<count($arr);$i++) {
         if (strtoupper($arr[$i])!="CRYPTO=1")
             $ciphertext.="&".$arr[$i];
         if (substr(strtoupper($arr[$i]),0,4)=="PID=")
             $pid=substr(strtoupper($arr[$i]),4,7);
     }
     $ciphertext=substr($ciphertext,1);
     $ciphertext=rawurlencode($ciphertext);
     $ciphertext=str_replace("%3D","=",$ciphertext);
     $ciphertext=str_replace("%26","&",$ciphertext);
     $crc=str_pad(dechex(crc32($ciphertext)),8,"0",STR_PAD_LEFT);
     for ($i=0;$i<4;$i++)
         $ciphertext.=chr(base_convert(substr($crc,$i*2,2),16,10));
     $pad=8-(strlen($ciphertext) % 8);
     for ($i=0;$i<$pad;$i++)
         $ciphertext.=chr($pad);
     $f=fopen(substr($pid,0,3).".des","r");
     $keyinfo=fread($f,38);
     fclose($f);
     $key1=substr($keyinfo,14,8);
     $key2=substr($keyinfo,22,8);
     $iv=substr($keyinfo,30,8);
     $key=$key1.$key2.$key1;
     $td=mcrypt_module_open("tripledes","","cbc","");
     mcrypt_generic_init($td,$key,$iv);
     $ciphertext=mcrypt_generic($td,$ciphertext);
     mcrypt_module_close($td);
     $pad=3-(strlen($ciphertext) % 3);
     for ($i=0;$i<$pad;$i++)
         $ciphertext.=chr($pad);
     $ciphertext=base64_encode($ciphertext);
     $ciphertext=rawurlencode($ciphertext);
     $ciphertext="PID=".$pid."&CRYPTO=1&DATA=".$ciphertext;
     return $ciphertext;
 }
```

Decryption is to be interpreted similarly (each response message sent by the bank is encrypted):

```
function eki_decrypt($ciphertext)
{
    $arr=split("&",$ciphertext);
    $cleartext="";
    $pid="";
    for ($i=0;$i<count($arr);$i++) {
        if (substr(strtoupper($arr[$i]),0,5)=="DATA=")
            $cleartext=substr($arr[$i],5);
        if (substr(strtoupper($arr[$i]),0,4)=="PID=")
            $pid=substr(strtoupper($arr[$i]),4,7);
    }
    $cleartext=rawurldecode($cleartext);
    $cleartext=base64_decode($cleartext);
    $lastc=ord($cleartext[strlen($cleartext)-1]);
    $validpad=1;
    for ($i=0;$i<$lastc;$i++)
        if (ord(substr($cleartext,strlen($cleartext)-1-$i,1))!=$lastc)
            $validpad=0;
    if ($validpad==1)
        $cleartext=substr($cleartext,0,strlen($cleartext)-$lastc);
    $f=fopen(substr($pid,0,3).".des","r");
         $keyinfo=fread($f,38);
    fclose($f);
    $key1=substr($keyinfo,14,8);
    $key2=substr($keyinfo,22,8);
    $iv=substr($keyinfo,30,8);
    $key=$key1.$key2.$key1;
    $td=mcrypt_module_open("tripledes","","cbc","");
    mcrypt_generic_init($td,$key,$iv);
    $cleartext=mdecrypt_generic($td,$cleartext);
    mcrypt_module_close($td);
    $lastc=ord($cleartext[strlen($cleartext)-1]);
    $validpad=1;
    for ($i=0;$i<$lastc;$i++)
        if (ord(substr($cleartext,strlen($cleartext)-1-$i,1))!=$lastc)
            $validpad=0;
    if ($validpad==1)
        $cleartext=substr($cleartext,0,strlen($cleartext)-$lastc);
    $crc=substr($cleartext,strlen($cleartext)-4);
    $crch="";
    for ($i=0;$i<4;$i++)
        $crch.=str_pad(dechex(ord($crc[$i])),2,"0",STR_PAD_LEFT);
    $cleartext=substr($cleartext,0,strlen($cleartext)-4);
    $crc=str_pad(dechex(crc32($cleartext)),8,"0",STR_PAD_LEFT);
    if ($crch!=$crc)
        return "";
    $cleartext=str_replace("&","%26",$cleartext);
    $cleartext=str_replace("=","%3D",$cleartext);
    $cleartext=rawurldecode($cleartext);
    return $cleartext;
}
```

Until the time of the activation, the bank issues two key pairs: a test key pair and an activation key pair. The test key pair is only suitable for communication with the bank's test server, and the activation key pair is only suitable for communication with the bank's live server. If used incorrectly, the bank's server sends a general error message (RC=S01). Considering that the test key pair has the same name as the activation key pair (in order to facilitate activation), it is recommended to store the checksum of the keys (e.g. MD5) so that in the future they can be clearly identified.

CIB BANK

Communications

Webshop-Bank communication

All the merchant's messages are exchanged with the bank's server via a http channel, using the URL specified by the bank. The generated and encrypted parameters can be forwarded to the bank's server using the GET or the POST method. The bank's response always arrives in the content part of the requested message, in text/plain format. If the request is processed successfully, the http status of the connection is 200, and the response arriving in the content is encrypted based on the EKI standard. In case of incorrect processing the http status is not 200 (typically, it is 403 or 500), the response in the content is not encrypted and it only contains the error code (e.g. if the request cannot be decrypted, the http status is 403 and the content is 'RC=S01'). Example of sending/receiving merchant messages:

```
function eki_call($url, $params)
{
   $ch = curl_init();
   curl_setopt($ch, CURLOPT_URL, ($url);
   curl_setopt($ch, CURLOPT_POST, 1);
   curl_setopt($ch, CURLOPT_POSTFIELDS, $params);
   curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
   curl_setopt($ch, CURLOPT_TIMEOUT, 30);
   $http_response = curl_exec ($ch);
   $http_status = curl_getinfo($ch, CURLINFO_HTTP_CODE);
   curl_close($ch);
   if ($http_status != "200") {
          echo "HTTP STATUS ".$http_status.", EKI error: ".$http_response;
          exit;
   }
   $ekiCryptedResponse = $http_response;
   return($ekiCryptedResponse);
}
```

Customer-Bank communication

Any method can be used to redirect the customer to the bank's payment page. In the case of a PHP code, if no content has been transferred to the browser as yet, the use of the header() function is recommended.

```
   header("Location: ".$eki_customerurl."?".$msgt20crypt);
   exit;
```

In addition to the above, it is also possible to use JavaScript window.location.replace(%customerurl%), or the html <meta http-equiv="refresh" content="0; url=%customerurl%" /> tag, where % customerurl% is the whole GET-parametered payment page URL of the bank.

CIB BANK

Problem management

In the above sections of this documentation we assumed that all the circumstances necessary for the execution of a successful transaction are present. In a live environment, however, any of the steps detailed in the above points may result in an error. The purpose of this section is to provide assistance in detecting and eliminating these errors.

Encryption problems

A successful encryption/decryption requires the availability of the encryption key provided by the bank to the merchant, the flawlessly operating encryption application and the correctly compiled processable message. The absence of any of the above conditions automatically entails the erroneous execution of the operation.

Encryption keys

the bank provides the webshop with two keys (key packages). Since the bank's encryption algorithm is a synchronous encryption, the test key can only be used for the bank's test server and the activation key can only be used for the live server. The test key and the activation key are different in terms of content, but not in names. If the bank's response code is continuously 403 and the content is 'RC=Sxx', this is most likely due to most likely due to an error of the key. It is possible (though unlikely) that the key got damaged during the period between its issue by the bank and the completion of installation on the web server, in which case it is advisable to verify the checksum (e.g. MD5).
The bank makes available the keys to the merchant in two formats: in .reg format (typically for Windows-based systems) and in .des format (typically for Unix-based systems). The former needs to be installed in the registry of the webshop.
During installation it must be ensured that (only) the technical user performing the encryption (typically the one operating the web server) can read the key files. It must also be ensured that the first 3 characters in the name of the encryption key (only capital letters) is exactly identical to the first 3 characters in the value of the PID parameter in the message that is, or is to be, encrypted.

Applications

Each step of the EKI encryption protocol is obligatory. In case of native implementation the final result of the series of encryption steps may be different from the expected result. This can be traced back to a faulty operation of any of these steps. When developing the application, the following must be kept in mind:

- When converting to text type, the correct encoding (ASCII) must always be provided, or, as long as possible, it is advisable to store the result in binary format.
- When calculating the CRC32, the big endian order must be applied.
- If the CRC32 value calculated during decoding is different from the received value, this qualifies as an error.
- If the applied 3DES encryption function uses implicit padding, then the message in front of it must not be padded.
- The RFC 1738 standard must be applied for URL encoding.

The above list is not complete, but it contains the most frequent errors.
As far as possible, the use of the attached example codes and the sakide functions library (and application) is recommended. In case of problem, running the sakide application with a '-v' switch

will give a more detailed error analysis. If the calling application can only handle the standard output, it is advisable to redirect the standard error channel:

```
/<path>/<to>/sakide -v -e -s"<string_to_encode>" 2>&1
```

Data

If, in response to the merchant's message, the status code of the bank's server is 500 and the response message is 'RC=Dxx', then it is likely that the error is in the transferred data. When putting together the data, the following must be kept in mind:

- The message must contain all the parameters listed in the reference guide.
- All the parameters must have the value format indicated in the guide
- During the lifecycle of each transaction, all the related messages must contain the same TRID value
- The amount to be authorised must be the same in each message related to the same transaction (e.g. because of an unsolved error branch it may happen that the confirmation message contains a 0 amount)

Communication problems

Messages are sent and received via the standard protocol (http), but generally not through the port assigned to it by default (80). This may cause a time-out, which eventually results in transaction failure or in the non-operation of the entire payment method. Whenever a communication problem is detected, the following should be checked:

- Correctness of the name of the bank's server at the place of calling (the address of the bank's test server is not identical to the live address)
- The web server's ability to identify the name of the bank's server
- Accessibility of the IP address and the target point of the bank's server on any traffic-limitation devices (e.g. firewall) before the web server

Generally, the communication problem can be reproduced on the web server using command-line devices (curl, wget) When reporting an error, please attach the related outputs as well.

CIB BANK

Support

The bank provides free assistance to the merchant for the investigation of any problematic transactions. The investigation must be requested by e-mail (ecommerce@cib.hu, with the text “Transaction investigation request <merchant identifier>” in the subject line), specifying the following in the message:

- Transaction ID of the problematic transaction
- Description of the problem, as precisely as possible
- Screenshot, if feasible
- IP address of the merchant’s server
- The sent/received encrypted messages provided with time stamps