Cyberbit A/S Development Resource Guide – Secure Payment Form

Internet Merchant Payment Solution

Version 1.1

Change Register

Version	Description	Author	Date
V 1.0	Document Created	Jimmi Kristensen	30-03-2007
V 1.1	Edited email and error code -16 Added the Terminal parameter	Jimmi Kristensen	06-06-2007

Contact Details

General Product Support	support@cyberbit.com Phone: +45 7027 0585
Technical Support / Questions	support@cyberbit.com Phone: +45 7027 0585

Table of Contents

Introduction

Cyberbit's Secure Payment Form (SPF) is an easily integrated solution, which permits secure and trusted transactions over the Internet. The following document describes how to integrate the payment solution into a merchant's website. The document will guide a developer through the integration process required to use Cyberbit's Secure Payment Form. To make the integration even easier, this guide also provides code examples for the various development requirements.

Secure connectivity is obtained over the Internet by establishing an SSL encrypted connection between the merchant's website and the Cyberbit Payment Gateway. To integrate a merchant's website with Cyberbit using the SPF, a developer must be able to provide client side security for receiving information. This is done by connecting to Cyberbit using the HTTPS protocol so as to pass this information using SSL.

This document provides information on how to implement the following e-commerce transaction types:

- Non-3D Secure Authorization Only.
- Non-3D Secure Authorization and Capture.

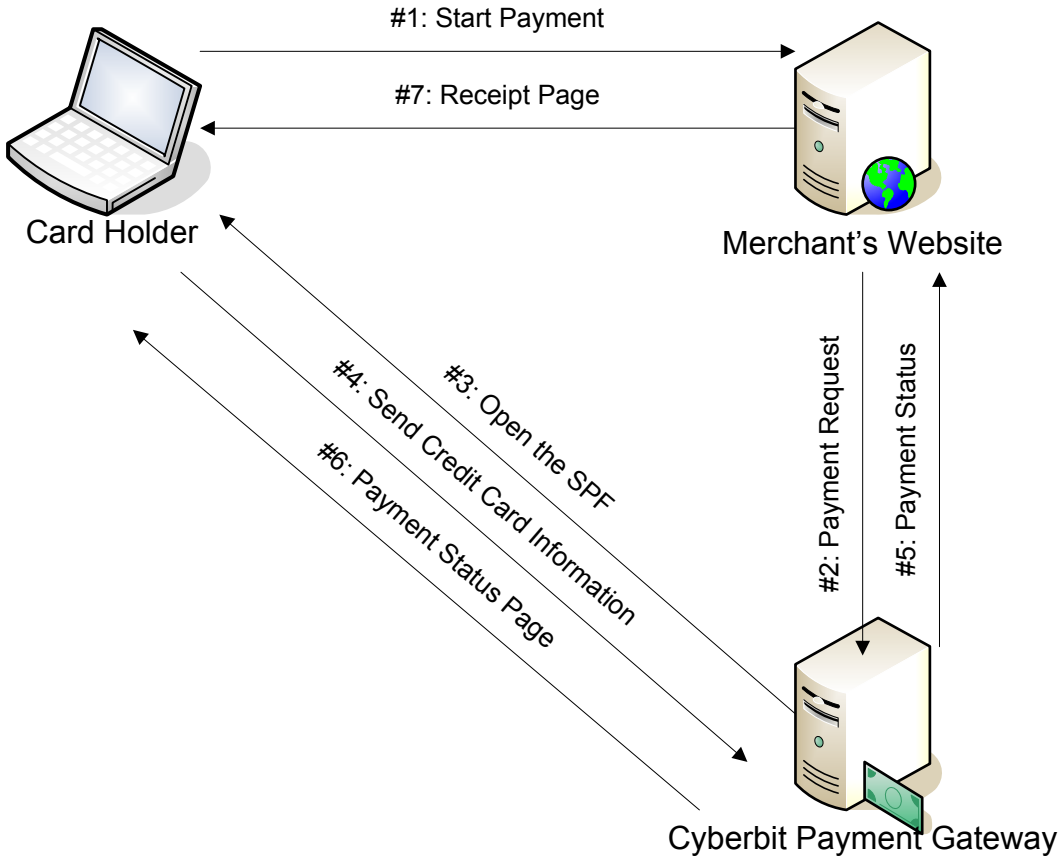
Requirements

Before the merchant will be able to start testing, a few things have to be in place. The merchant must have received a test account with the following information:

MerchantId	Unique ID assigned to merchants.
Password	Password used to login to https://test.eccpro.com and https://eccpro.com
Secret Code	Unique code sent with every transaction.
Hashing Code	Secret hashing code used to validate transaction data. This code must only be known by the merchant and Cyberbit A/S.

Besides the information above, the merchant must supply Cyberbit A/S with the exact URL(s) he/she want to be able to start the SPF from. If for example the merchant's payment form is located at https://company/start_payment.php, this URL must be given to Cyberbit A/S. The SPF can only be started from these URLs else an error will be shown.

System Overview



The diagram above shows the payment process when using Cyberbit's SPF.

1. When the card holder have selected the goods he/she wants to buy and have filled a form, on merchant's website, stating his/hers name, address, shipping info, etc., the card holder presses a link/button to start the payment.
2. Merchant's website opens a payment window linked to Cyberbit's SPF. For further details see the chapter "Implementing the Methods".
3. The card holder is presented with the SPF located on a Cyberbit web server.
4. The card holder fills the SPF with the needed credit card information, and sends payment information to Cyberbit's Payment Gateway.
5. Payment status is sent to the merchant's web server to ensure that the transaction/order is registered at the merchant's website. For further details please see the chapter "Callback from the Payment Gateway".
6. A payment status is sent back to the card holder, this status page displays if the transaction was a success.
7. Card holder is directed back to merchant's website to get his/hers receipt.

Sending Transactions

This chapter will describe the procedures and parameters to send the different types of transactions to the SPF. All transactions will be made through a HTTPS POST as described below.

Format Values Key:
A = Alphabetical (a-z, A-Z)
N = Numeric (0-9)
AN = Alpha-Numeric (a-z, A-Z, 0-9)
URL = Uniform Resource Locator

Presence Values Key:
R = Required
O = Optional
C = Conditional

Parameter Name	Format	Presence	Description
TransType	N(1)	R	1 = Authorize only 3 = Authorize and Capture
Code	A(40)	R	Secret transactions code provided by Cyberbit A/S
AcceptURL	URL	R	URL to return the customer to when a successful transaction have been made. To see an example of an accept page written in PHP, go to “PHP code example”.
CallbackURL	URL	R	URL to return the callback response.
ShopIP	AN(15)	R	IP of the online shop
MerchantId	AN(10)	R	Merchant ID provided by Cyberbit A/S
OrderId	AN(22)	C	Order ID supplied by the merchant. If the **Terminal** parameter is set this parameter is ignored and does not have to be set. **Note** that if an error occurs and the OrderId parameter is used the SPF have to be closed before a new attempt can be made. For example if the cardholder writes his/her card number, CVC or expire date wrong, the SPF will show an error message telling the cardholder to close the SPF and try again, it is then up to the merchant to generate a new OrderId before starting a new SPF. To avoid this problem use the Terminal parameter instead.
CurrencyCode	N(3)	R	Currency code according to ISO-4217
CurrencyTLA	A(3)	R	If CurrencyCode = 000 a given CurrencyTLA is converted to the main currency chosen. Example: If the main currency is USD and a converstion from GBP to USD is desired, CurrencyCode

			should be 000 and CurrencyTLA should be GBP Learn more: http://en.wikipedia.org/wiki/ISO_4217
Amount	N(11)	R	Total amount of purchase. Amount is given in the smallest amount (Example: if the currency is USD then an amount of 100 will be equal to 1 Dollar)
Terminal	Terminal	O	If Terminal is set the Cyberbit Payment Gateway will automatically generate an order id for the order being processed. This means instead of using an order id generated by the merchant the Payment Gateway will generate an order id and return this id to the callback URL. If the Terminal parameter is set the OrderId parameter will be ignored and can be left out.
OwnerEmail	AN(100)	C	Cardholder's email address
OwnerAddress	AN(50)	C	Cardholder's address
OwnerAddressNumber	N(20)	C	Cardholder's address number
OwnerCity	AN(50)	C	Cardholder's city
OwnerCountryCode	A(2)	C	ISO-3166-1 alpha-2 code of the cardholder country (Example: if the card holder is from the United Kingdom the country code will be GB)
OwnerState	A(2)	C	Cardholder's state, for a list of valid states see "State Codes"
OwnerFirstName	A(20)	R	Cardholder's first name
OwnerLastName	A(20)	R	Cardholder's last name
OwnerZIP	AN(10)	C	Cardholder's zip/postal code
OwnerPhone	AN(20)	C	Cardholder's phone number
OwnerDayOfBirth	N(2)	C	Cardholder's day of birth (DD)
OwnerMonthOfBirth	N(2)	C	Cardholder's month of birth (MM)
OwnerYearOfBirth	N(4)	C	Cardholder's year of birth (YYYY)

The parameters above are the only data required to make a transactions. The parameters below are all optional, and will only be shown on the SPF and in the returned XML answer from Cyberbit's Payment Gateway.

Parameter Name	Format	Presence	Description
Numcolumns	N(30)	O	Number of columns shown in the "Details" window on the SPF
Numrows	N(30)	O	Number of rows shown in the "Details" window on the SPF
Shopname	AN(100)	O	Name of the online shop (will only be shown in the SPF)

Implementing the Methods

An authorize or an authorize/capture request is made by sending a HTTPS POST to the Cyberbit Payment Gateway.

The Cyberbit Payment Gateway is accessible through the following URL:
Test: https://test.eccpro.com/`securepayment/payment.php`
Live: https://eccpro.com/`securepayment/payment.php`

Below is a code example of how to make the HTTPS POST and start the SPF.

Firstly the SPF have to be prepared to take the POST from merchant's payment window. In the head (`<head>`) part of the merchant's payment page put the following JavaScript code to start the SPF:

```
<script type="text/javascript">
window.name = 'windowOp';

function paymentWindow() {
   var win =
window.open("https://test.eccpro.com/securepayment/payment.php","payment","width
=480,height=520");
   win.focus();
   return true;
}

function startSPF() {
   paymentWindow();
   document.myform.submit();
}
</script>
```

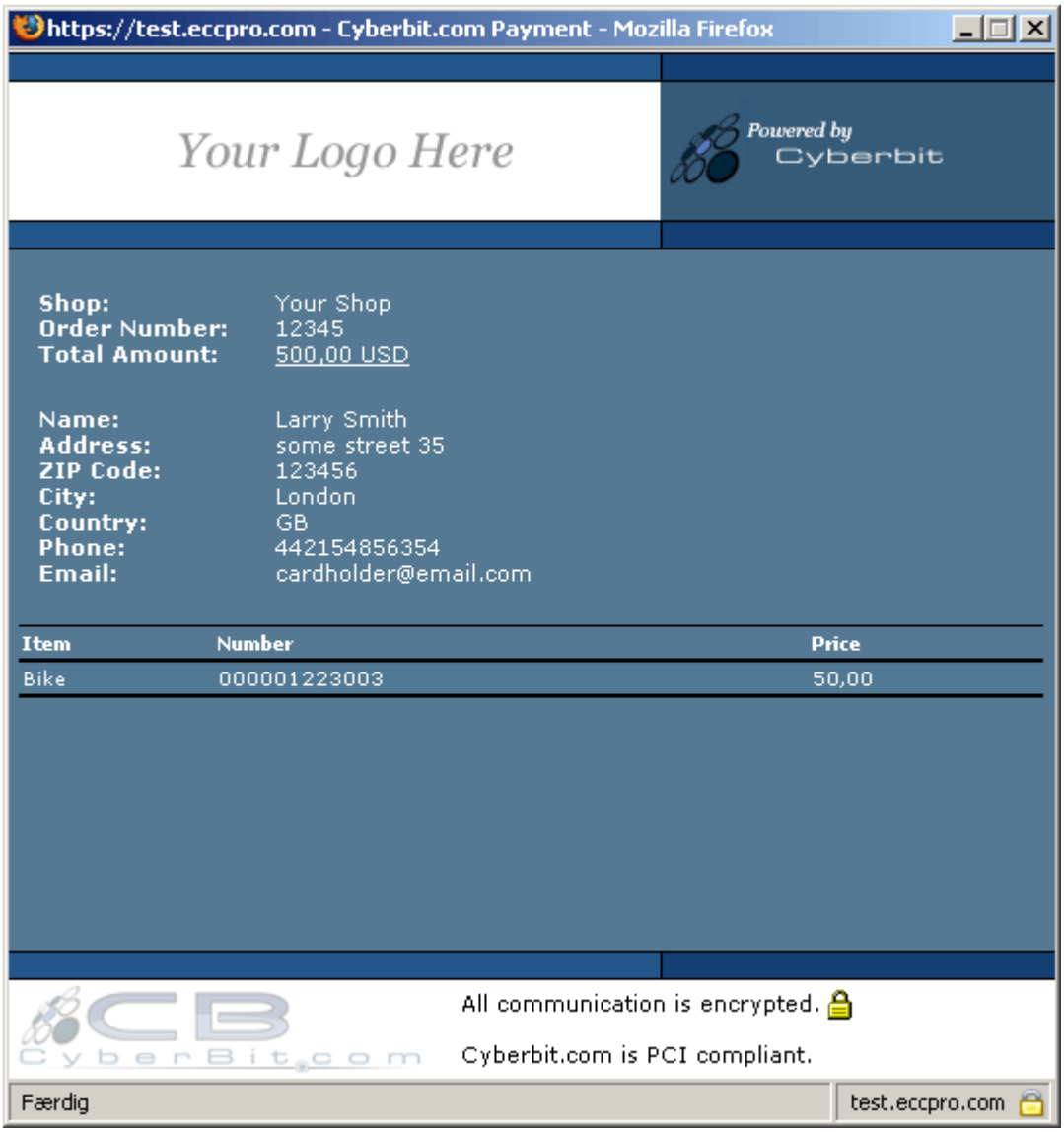
Next, prepare the POST. Below is an example on how a POST could look like:

```
<form name="myform" method="POST" target="payment"
action="https://test.eccpro.com/securepayment/paymentstart.php"
onsubmit="startSPF();">
<input type="hidden" value="1" name="transtype">
<input type="hidden" value="xxxxxxxxx" name="code">
<input type="hidden" value="https://path/to/your/accept/page/accept.php"
name="accepturl">
<input type="hidden" value="https:// path/to/your/accept/page/callback.php"
name="callbackurl">
<input type="hidden" value="xxx.xxx.xxx.xxx" name="shopip">
<input type="hidden" value="xxxxxxxxx" name="merchantid">
<input type="hidden" value="12345" name="orderid">
<input type="hidden" value="840" name="currencycode">
<input type="hidden" value="50000" name="amount">
<input type="hidden" value="Your Shop" name="shopname">
<input type="hidden" value="cardholder@email.com" name="owneremail">
<input type="hidden" value="some street" name="owneraddress">
<input type="hidden" value="123" name="owneraddressnumber">
<input type="hidden" value="London" name="ownercity">
<input type="hidden" value="OO" name="ownerstate">
<input type="hidden" value="GB" name="ownercountrycode">
<input type="hidden" value="Larry" name="ownerfirstname">
<input type="hidden" value="Smith" name="ownerlastname">
<input type="hidden" value="123456" name="ownerzip">
<input type="hidden" value="+442154856354" name="ownerphone">
<input type="hidden" value="3" name="numcolumns">
<input type="hidden" value="1" name="numrows">
<input type="hidden" value="Item" name="column1">
<input type="hidden" value="Number" name="column2">
<input type="hidden" value="Price" name="column3">
<input type="hidden" value="Bike" name="column1_1">
<input type="hidden" value="000001223003" name="column2_1">
<input type="hidden" value="50,00" name="column3_1">
<input type="submit" value="Make Payment">
</form>
```

The example above will create a payment form with the above details. Notice the fields under the “numcolumns” and “numrows” fields. The “numcolumns” and “numrows” fields defines how many columns and rows to display in the “Detials” window. All fields below those two fields will create a table containing any desired data. In this example a table, like the one below will be created, in the “Details” window.

	Column 1	Column 2	Column 3
	Item	Number	Price
Row 1	Bike	000001223003	50,00

To create a table like the one above, the first thing that has to be added is how many rows and columns we want to add, which are defined by the “numcolumns” and “numrows” fields. The next thing needed is the titles of the table. These are defined by the “columnN” fields. If, like in this example, three titles are desired, create three column fields “column1”, “column2” and “column3”. To add something to the table, create fields “column<col>_<row>” like in the example above, “column1_1”, “column2_1” and “column3_1”. This will create a “Detail” window like in the illustration below:



Example of the "Detail" window.

NOTE: Note that the "numcolumns" and "numrows" and all fields related to the table are optional.

Callback from Payment Gateway

After every transaction the merchant will receive a **POST** message with status of the transaction. This **POST** will be sent to the callback URL defined by the merchant's website when the transaction were started. The merchant's callback page will receive four callback parameters:

Parameter Name	Description
Statuscode	Status code of the transaction, this will show if the transaction was successful.See "Transaction Status Codes" for a list of status codes.
Statustext	Status text will show, in text, if the transaction was successful.
Fingerprint	The SHA1 hash of the transaction, see "Validating Transaction Data" for further details.
Xml	Will return all data sent by the merchant's website to Cyberbit's Payment Gateway in XML format. Besides data from the merchant's website, this XML will also hold the transaction data from the Payment Gateway. To see an example of a returned XML message, see the chapter "XML Example".

The purpose of the callback message is to inform the merchant's website of the transaction status. The callback message is sent promptly after a transaction have been made, in this way the merchant's website will be able to acknowledge a transaction, even though the cardholder might close the SPF immediately after a transaction have been made and therefore not getting a receipt from the merchant's receipt page.

It is important that the merchant acknowledges the transaction on the callback URL instead of on the accept URL. The reason to this is that the cardholder might close the SPF immediately after a transaction have been made, and therefore never reach the accept page. If this is the case and the merchant only acknowledges a successful transaction on the accept page, the money will be drawn from the cardholder's bank account, but the merchant will never register the transaction, and therefore will never send the goods. If the transaction is acknowledged on the callback page the merchant will always be able to acknowledge a transaction, even though the SPF is closed by the cardholder.

All errors that may happen in a transaction will also be returned to the callback URL. In this way the merchant is able to log the error message returned by the Payment Gateway.

Validating Transaction Data

The merchant must validate the transaction data sent from the Payment Gateway, to make sure it has not been altered in any way. The merchant will receive a unique hashing code by Cyberbit A/S which will be used in the validation process. The validation must take place on the page linked to the callback URL when the transaction is acknowledged. To validate the data create a SHA1 hashing of the unique key received by Cyberbit A/S and the XML parameter of the callback message. Below is an example if the SHA1 creation written in PHP:

```
$fingerprint = sha1($uniqueKey.$xmlReturn);
```

Note that the unique key is before the XML message. When the SHA1 hashing has been made, compare this value to the value received in the fingerprint parameter of the callback message.

IMPORTANT: The unique key is only known by Cyberbit A/S and the merchant; make sure the key is now shown on the merchant's website.

Finishing the Payment Process

When a successful transaction has been made, the cardholder will be directed back to the merchant's website. Because of the cross-domain policies in the mozilla browser, the merchant's accept page will be shown in the payment window. Because of this it is recommended that the merchant's accept page takes the parameters from the POST given to the accept page, send the data to a receipt page on the merchant's website and closes the payment window. In this way the cardholder will be directed back to the merchant's website showing a receipt page. For a PHP example of a possible accept page, see "PHP Code Example".

For a complete "Copy/Paste" example of the entire payment process written in PHP and JavaScript, see "PHP Code Example".

Transaction Status Codes

Status Codes	Status Text
-1	In correct Matching Code
-2	Couldn't connect to MySQL Server
-3	Couldn't select MySQL Database
-5	TransType not valid
-8	No Authorization to Capture
-9	Capture Error
-10	No Capture To Refund
-11	Refund Error
-12	Duplicated OrderID
-13	Invalid Credit Card Number
-14	Too Many transactions from the same card within 24 hours
-15	Too much money cleared from the same card within 24 hours
-16	Amount cleared exceeds the max ticket price
000	Status Code Text
001	Missing or wrong MerchantId
002	Redirection not active
003	No Clearing URL specified
004	Wrong or Empty SecretCode
005	No Processing Available.
006	Access Error! Restricted by Ip-Address.
007	Illegal characters in InputField
008	OwnerCountry is not valid. Should have a compatible ISO 3166 Value
009	TransType are not supported by this Clearing-Gateway
010	Error From Processor
011	HTTP_GET are not supported use. HTTP_POST instead
012	OwnerState is not valid. Hint use OO for outsite USA/Canada.
013	Error in our Perl-Gateway Software. Critical Error.
014	Your are currently not Online
015	The AmountCleared should be at least 100 !!
016	Hello I'm Online PHP.
017	Hello I'm Online Perl.
018	CashServer Developer Breakpoint. (internal only).
019	Cashserver is down for maintainment.
020	Required Fields.
021	Missing InternalOrderId.
022	The Character backslash detected in (Field). Invalid input.
023	Cannot find at Authorize/Capture to process.
024	Terminal Unique Id Failure
025	Terminal Security Issue. Wrong Harddisc serial Number
026	Aquirer UserId is not valid
027	Invalid ChargeBack Code
028	Invalid XML code from Acquirer
029	The creditcard expiredate is not valid
030	CurrencyCode is not valid
031	Illegal Currency
032	Illegal Internal Currency
033	Currency Convertion Failed
034	CurrencyConvert (PHP)
035	ShowCurrencies
036	Length of InternalOrderId may not exceed ($number) characters
037	PerlLink does not Match
038	Aquirer ClearingCard MID is not valid
039	TestCard Number not allowed on Live System
040	ShowAcquireCodes
041	CreditCardCVC code is required
042	RefundLimit is exceeded
043	CreditCards from $country is Blocked
044	This acquirer does not support recurring transactions
045	Billing interval must be -5 or higher, check merchant guide for more info
046	Transtype Authorize or Sale after Authorize is not allowed here. Use Transtype 3 or 8 Instead
047	IP and Credit Card is not from same country

State Codes

US/Canadian State Codes	State
OO	Outside US or Canada
AL	Alabama
AK	Alaska
AZ	Arizona
AR	Arkansas
CA	California
CO	Colorado
CT	Connecticut
DC	District of Columbia
DE	Delaware
FL	Florida
GA	Georgia
HI	Hawaii
ID	Idaho
IL	Illinois
IN	Indiana
IA	Iowa
KS	Kansas
KY	Kentucky
LA	Louisiana
ME	Maine
MD	Maryland
MA	Massachusetts
MI	Michigan
MN	Minnesota
MS	Mississippi
MO	Missouri
MT	Montana
NE	Nebraska
NV	Nevada
NH	New Hampshire
NJ	New Jersey
NM	New Mexico
NY	New York
NC	North Carolina
ND	North Dakota
OH	Ohio
OK	Oklahoma
OR	Oregon
PA	Pennsylvania
PR	Puerto Rico
RI	Rhode Island
SC	South Carolina
SD	South Dakota
TN	Tennessee
TX	Texas
UT	Utah
VT	Vermont
VA	Virginia
WA	Washington
WV	West Virginia
WI	Wisconsin
WY	Wyoming
AB	Alberta
BC	British Columbia
MB	Manitoba
NB	New Brunswick
NF	New Foundland
NT	Northwest Territories
NS	Nova Scotia
NT	Nunavut

ON	Ontario
PE	Prince Edward Island
QC	Quebec
SK	Saskatchewan
YT	Yukon

PHP Code Example

This chapter will show examples in PHP and JavaScript of the entire payment process. The code can be copy/pasted for testing purpose.

The payment page. The payment form is built and the Secure Payment Form is prepared. When the "Make Payment" button is pressed, the SPF will start and the cardholder can start the payment.

```
<html>
<head>

<script type="text/javascript">
window.name = 'windowOp';

function paymentWindow() {
   var win =
window.open("https://test.eccpro.com/securepayment/payment.php","payment","width
=480,height=520");
   win.focus();
   return true;
}

function startSPF() {
   paymentWindow();
   document.myform.submit();
}
</script>

</head>
<body>

<form name="myform" method="POST" target="payment"
action="https://test.eccpro.com/securepayment/paymentstart.php"
onsubmit="startSPF();">
<input type="hidden" value="1" name="transtype">
<input type="hidden" value="xxxxxxxx" name="code">
<input type="hidden" value="https://path/to/your/accept/page/accept.php"
name="accepturl">
<input type="hidden" value="https:// path/to/your/callback/page /callback.php"
name="callbackurl">
<input type="hidden" value="xxx.xxx.xxx.xxx" name="shopip">
<input type="hidden" value="xxxxxxxx" name="merchantid">
<input type="hidden" value="1234" name="orderid">
<input type="hidden" value="840" name="currencycode">
<input type="hidden" value="50000" name="amount">
<input type="hidden" value="Your Shop" name="shopname">
<input type="hidden" value="cardholder@email.com" name="owneremail">
<input type="hidden" value="Some Street" name="owneraddress">
<input type="hidden" value="123" name="owneraddressnumber">
<input type="hidden" value="London" name="ownercity">
<input type="hidden" value="OO" name="ownerstate">
<input type="hidden" value="GB" name="ownercountrycode">
<input type="hidden" value="Larry" name="ownerfirstname">
<input type="hidden" value="Smith" name="ownerlastname">
<input type="hidden" value="123456" name="ownerzip">
<input type="hidden" value="+451234567890" name="ownerphone">
<input type="hidden" value="3" name="numcolumns">
<input type="hidden" value="1" name="numrows">
<input type="hidden" value="Item" name="column1">
```

```
<input type="hidden" value="Number" name="column2">
<input type="hidden" value="Price" name="column3">
<input type="hidden" value="Bike" name="column1_1">
<input type="hidden" value="000001223003" name="column2_1">
<input type="hidden" value="50,00" name="column3_1">
<input type="submit" value="Make Payment">
</form>

</body>
</html>
```

Page linked to the Callback URL. The code below is for test purpose only to see what data is returned. It writes all callback data to a file. In a "Live" situation this page should be where the order is acknowledged and reserved for the cardholder.

OBS: Be sure the chante (`$hashingKey = "xxxxxxx";`) with the hashing code provided by Cyberbit A/S.

```
<?
// This PHP example writes all callback data to a file called callback.txt

$hashingKey = "xxxxxxx";
$statuscode = trim($_POST['statuscode']);
$statustext= trim($_POST['statustext']);
$fingerprint= trim($_POST['fingerprint']);
$xml = trim($_POST['xml']);

// Check if fingerprint matches
if (sha1($hashingKey.$xml) == $fingerprint) {
   $string = "STATUSCODE: " . $statuscode . "\n";
   $string.= "STATUSTEXT: " . $statustext . "\n";
   $string.= "FINGERPRINT: " . $fingerprint . "\n\n";
   $string.= $statuscode;
} else {
   $string = "Fingerprint did not match";
}

$handle = fopen("callback.txt","w");
fwrite($handle, $string);
fclose($handle);
?>
```

Accept page. This is the accept page the cardholder will be returned to when a payment is done. Because if the cross-domain policies in mozilla, as described earlier, this page will be shown in the payment window. The example below will take the data returned by the Payment Gateway and direct the cardholder and the data to a receipt page on the merchant's website.

```
<html>
<head>
</head>
<body onload="submitToReceipt()">

<script language="JavaScript" type="text/javascript">
function submitToReceipt() {
   document.forward_receipt.action="https://path/to/your/receipt/page/receipt.ph
p"
   document.forward_receipt.target = parent.opener.name;
   document.forward_receipt.submit();
```

```
   parent.close();
}
</script>

<form name="forward_receipt" method="post">
   <input type="hidden" name="statuscode" value="<?=$_POST['statuscode']?>">
   <input type="hidden" name="statustext" value="<?=$_POST['statustext']?>">
   <input type="hidden" name="fingerprint" value="<?=$_POST['fingerprint']?>">
   <input type="hidden" name="xml" value="<?=$_POST['xml']?>">
</form>

</body>
</html>
```

Receipt page. The example below is for testing purpose, and will take the data sent from the merchant's accept page above and parse the XML.

```
<?
$statuscode = $_POST['statuscode'];
$statustext = $_POST['statustext'];
$fingerprint = $_POST['fingerprint'];
$xml = urldecode($_POST['xml']);

print $statuscode . "<br>";
print $statustext . "<br>";
print $fingerprint . "<br><br>";

$p = xml_parser_create();
xml_parse_into_struct($p, $xml, $vals, $index);
xml_parser_free($p);

for ($i = 0; $i < count($vals); $i++) {
   if ($vals[$i][type] == 'complete') {
      print $vals[$i][tag] . " - " . $vals[$i][value];
   }
}
?>
```

XML Example

The example below is how an XML message could look. The information in the top, the <Response> tag, is information specific from the Payment Gateway and the information in the lower part, the <ReturnInfo> is information received from the merchant's website.

```
<ECCPro>
  <Response>
    <StatusCode>000</StatusCode>
    <StatusText>Success From Processor</StatusText>
    <MethodCall>POST</MethodCall>
    <SiteURL>test.eccpro.com</SiteURL>
    <IpAddress>xxx.xxx.xxx.xxx</IpAddress>
    <ProcessDate>2007-03-29 16:16:34</ProcessDate>
    <ProcessUsedTime>1.3863520622253</ProcessUsedTime>
    <MerchantId>xxxxxxxx</MerchantId>
    <ProcessStatus>0</ProcessStatus>
    <ProcessStatusText>Transaction OK</ProcessStatusText>
    <AcquireCode>0</AcquireCode>
    <AcquireText>Transaction OK</AcquireText>
    <OrderID>test1234</OrderID>
    <AuthResponse>422666</AuthResponse>
    <ProcessOrderID>C381600117517779515367_422666</ProcessOrderID>
    <ReturnInfo>
      <cardtype>Visa</cardtype>
      <shopip>xxx.xxx.xxx.xxx</shopip>
      <orderid>1234</orderid>
      <currencycode>840</currencycode>
      <amount>50000</amount>
      <shopname>Your Shop</shopname>
      <owneremail>cardholder@email.com</owneremail>
      <owneraddress>some street</owneraddress>
      <ownercity>London</ownercity>
      <ownercountrycode>GB</ownercountrycode>
      <ownerfirstname>Larry</ownerfirstname>
      <ownerlastname>Smith</ownerlastname>
      <ownerzip>123456</ownerzip>
      <ownerphone>+441234567890</ownerphone>
      <numcolumns>3</numcolumns>
      <numrows>1</numrows>
      <column1>Item</column1>
      <column2>Number</column2>
      <column3>Price</column3>
      <column1_1>Bike</column1_1>
      <column2_1>000001223003</column2_1>
      <column3_1>50,00</column3_1>
    </ReturnInfo>
  </Response>
</ECCPro>
```