Grants Payment Platform Documentation

Web Payment Frontend

Version 3.2.3
## Release History

<table>
<thead>
<tr>
<th>Release</th>
<th>Description</th>
<th>Date</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9</td>
<td>Beta Version</td>
<td>15-Feb-2004</td>
<td>Created document</td>
</tr>
<tr>
<td>1.0</td>
<td>Initial Version</td>
<td>02-Mar-2004</td>
<td></td>
</tr>
<tr>
<td>[...</td>
<td>[...</td>
<td>[...</td>
<td></td>
</tr>
<tr>
<td>2.1.0</td>
<td>Major Release</td>
<td>01-Sep-2009</td>
<td>MD5 HASH generation to validate WPF response message</td>
</tr>
<tr>
<td>2.2.0</td>
<td>Major Release</td>
<td>02-Aug-2010</td>
<td>Major Cleanups</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Minor Release</td>
<td>15-Nov-2010</td>
<td>Language Parameter added</td>
</tr>
<tr>
<td>3.0.0</td>
<td>Major Release</td>
<td>07-Dec-2010</td>
<td>Major text revision and clarification</td>
</tr>
<tr>
<td>3.1.0</td>
<td>Minor Release</td>
<td>20-Apr-2011</td>
<td>Added UTF-8 encoding rules</td>
</tr>
<tr>
<td>3.2.0</td>
<td>Major Release</td>
<td>16-Feb-2012</td>
<td>MD5 Security-HASH generation replaced by SHA-1</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Minor Release</td>
<td>19-June-2012</td>
<td>Minor typo errors corrected</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Minor Release</td>
<td>02-Nov-2012</td>
<td>New language support: Russian, Turkish, Chinese</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Minor Release</td>
<td>19-Nov-2012</td>
<td>Additional information for individual .css in chapter 9.1</td>
</tr>
</tbody>
</table>
Preface

PSP Platform Documentation – Web Payment Frontend

Copyright © 2012 Grants Payment - All rights reserved.
Printed in Germany / European Union

The information contained in this document is intended only for the person or entity to which it is addressed and contains confidential and/or privileged material. Any review, retransmission, dissemination or other use of, or taking of any action in reliance upon, this information by persons or entities other than the intended recipient is prohibited. If you received this in error, please contact Grants Payment and delete the material from any computer.
## Content

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release History</td>
<td>2</td>
</tr>
<tr>
<td>Preface</td>
<td>3</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>5</td>
</tr>
<tr>
<td>2. Page Flow</td>
<td>6</td>
</tr>
<tr>
<td>3. Control Flow</td>
<td>10</td>
</tr>
<tr>
<td>4. Examples</td>
<td>12</td>
</tr>
<tr>
<td>4.1 PHP Code</td>
<td>12</td>
</tr>
<tr>
<td>4.1.1 call-wpf-page.php</td>
<td>12</td>
</tr>
<tr>
<td>4.1.2 response-page.php</td>
<td>15</td>
</tr>
<tr>
<td>4.2 ASP Code</td>
<td>15</td>
</tr>
<tr>
<td>4.2.1 call-wpf-page.aspx</td>
<td>15</td>
</tr>
<tr>
<td>4.2.2 response-page.aspx</td>
<td>18</td>
</tr>
<tr>
<td>5. Request Encoding</td>
<td>19</td>
</tr>
<tr>
<td>6. WPF Request</td>
<td>20</td>
</tr>
<tr>
<td>7. WPF Asynchronous Response</td>
<td>22</td>
</tr>
<tr>
<td>7.1 Validation of the Response Parameters (optional)</td>
<td>23</td>
</tr>
<tr>
<td>7.2 Merchant Response to the WPF Asynchronous Response</td>
<td>24</td>
</tr>
<tr>
<td>8. Registration with WPF</td>
<td>25</td>
</tr>
<tr>
<td>9. WPF UI Configuration</td>
<td>26</td>
</tr>
<tr>
<td>9.1 Add your own CSS</td>
<td>26</td>
</tr>
<tr>
<td>9.2 Change default behaviour with Java Script</td>
<td>28</td>
</tr>
<tr>
<td>10 Other Configuration Options</td>
<td>31</td>
</tr>
<tr>
<td>10.1 Languages</td>
<td>31</td>
</tr>
<tr>
<td>10.2 Banners</td>
<td>31</td>
</tr>
<tr>
<td>10.3 Hide or Show Payment Methods and Types</td>
<td>32</td>
</tr>
<tr>
<td>10.4 Links Configuration</td>
<td>32</td>
</tr>
<tr>
<td>10.5 WPF Size Configuration</td>
<td>33</td>
</tr>
<tr>
<td>10.6 Change Button Appearance</td>
<td>33</td>
</tr>
<tr>
<td>10.7 Immediate Redirect after Payment / Registration</td>
<td>34</td>
</tr>
<tr>
<td>11 WPF LIGHT</td>
<td>35</td>
</tr>
<tr>
<td>12 FAQ</td>
<td>36</td>
</tr>
<tr>
<td>13 SHA-1 Hash generation</td>
<td>38</td>
</tr>
<tr>
<td>13.1 Java Sample Code</td>
<td>38</td>
</tr>
<tr>
<td>13.2 PHP Sample Code</td>
<td>39</td>
</tr>
</tbody>
</table>
1 Introduction

This document provides a detailed technical documentation of the Web Payment Frontend. It gives a general overview of the workflow and shows how to integrate it into an existing shop.

For a detailed list of all parameters available for the Web Payment Frontend refer to the document "Post Transactions".
2 Page Flow

The page flow when using the Web Payment Frontend (WPF) is an asynchronous one which consists of visible and invisible pages that are executed on the merchant’s server as well as on the payment server. Despite the asynchronous nature of the whole process your customer will never notice those complexities and (if conducted correctly) he or she might not even notice the fact that he or she is leaving your page (this can be done by embedding the WPF into a frame within your shopping cart).

The general flow is described in Figure 1.

![Figure 1 WPF Page Flow](image)

The response page must be running on a web server accessible through the default ports: HTTP port 80 or HTTPS port 443.

The user usually starts with a shopping cart at your system where you collect some of the data necessary to fulfill the order, for example the name, shipping address, contact info and of course the product the customer desires to order. During that process the customer clicks on a button to place the order. See Figure 2 for a sample shopping cart.
Example WebShop in ASP

This web shop example shows you how to call the web payment frontend (WPF) from ASP.

Upon clicking the "order now" button the shopping cart calls the wpf-call-page. This is a hidden page that should evaluate the values already collected in the shopping cart and build the WPF request out of them. It then redirects the user browser to the WPF on the payment server.
Figure 3 WPF Page
Depending on the values the shopping cart has already collected the user must enter some more data (especially of course credit card data or account information). If he or she has entered everything that is necessary, he or she clicks the "pay now" button, then the payment server executes the payment request and after that the processing system calls the (hidden) response-page on the merchant server. The response page decides upon receiving the request which page to show next. If the payment request was successful you can e.g. display a success-page as shown in figure 4.

Figure 4 Success Page

If the payment request failed you can e.g. display an error-page as shown in Figure 5:

Figure 5 Error Page
3 Control Flow

As explained in chapter 2 the workflow for the Web Payment Frontend (WPF) is highly asynchronous where both, the merchant server and the payment server, share the control over the customer browser. During that flow one server acts once as the serving party and the other server as the client before they change roles. The following figure shows which server acts in which role during that workflow.

Figure 6 Control Flow of the WPF

a) The user enters the merchants shopping cart (the merchant server is in control).

b) The user enters his personal data and selects the products he or she desires to order. At some point he clicks the "order now" button.

c) The call-wpf-page is requested. This is a hidden page on the merchant’s server that creates the initial WPF request out of the data received from the shopping cart. The merchant server sends that request to the payment server. Please ensure to define a parameter for the URL of the response-page (i.e. the page where the merchant server receives the results of the payment).

d) The payment server receives the initial request and creates a payment session. Then it sends a URL identifying the session back to the merchants server.

e) The merchant server redirects the user’s browser to the payment server. Effectively the merchant’s server cedes control to the payment server.
f) The user enters now the remaining data necessary to execute the payment (especially the credit card data or the bank account data) and clicks on the "pay now" button.


g) The payment server now calls the response-page on the merchant’s server and posts all parameters that describe the result of the payment to the merchant server. The response-page evaluates the post parameters.

h) After evaluation of the response parameters the response-page MUST reply with a plain text URL that identifies the next page to be displayed in the user’s browser AND that is also hosted on the merchant’s server.

i) The payment server redirects the user’s browser to the URL it received from the previous request. Usually it is a success-page if the payment succeeded otherwise it is an error-page. Effectively the payment server returns control over the user’s browser back to the merchant server.
4 Examples

The most complex parts within the workflow are the call-wpf-page and the response-page. To give you a quick start we have added sample code here so that you can easily cut and paste and run. Of course it will not run out of the box as you might need different URLs and login data. Please read carefully the comments within the code.
Also see the document “Implementation Packages” for more details. This document also contains samples in other programming languages.

Important Note:
The Response Page must be situated on a web server running on the default ports: HTTP port 80 or HTTPS port 443.

4.1 PHP Code

For all transactions replace the following parameters with your test or live parameters:
- SECURITY.SENDER
- USER.LOGIN
- USER.PWD
- TRANSACTION.CHANNEL
See document “Technical Quick Start” for more details.

4.1.1 call-wpf-page.php

```php
<?php

//this URL for the TEST web payment frontend
$url = "https://test.ctpe.net/frontend/payment.prc";

//this URL for the LIVE web payment frontend
// $url = "https://ctpe.net/frontend/payment.prc";

//these are the parameters that identify the caller
to the system. the ones below wont work as they are for
//a local test only.
//please contact your sales rep for the login data for your
//test account and after successful integration for the
//LIVE account data

$parameters['SECURITY.SENDER'] = "ff8080820249b2ac010249b74aa10009";
$parameters['USER.LOGIN'] = "ff8080820249b2ac010249b74aa40015";
$parameters['USER.PWD'] = "demomerchant";
$parameters['TRANSACTION.CHANNEL'] = "ff8080820249b2ac010249b74aac003a";

//switch this parameter to "LIVE" when starting the LIVE integration
$parameters['TRANSACTION.MODE'] = "INTEGRATOR_TEST";

$parameters['REQUEST.VERSION'] = "1.0";
//its at the moment not mandatory to send a unique tx-id to
//the web payment frontend but it is considered a good practice
//as you will be able to identify your tx when the WPF calls your
//result page.
$parameters['IDENTIFICATION.TRANSACTIONID'] = "your tx id";
$parameters['FRONTEND.ENABLED'] = "true";
$parameters['FRONTEND.POPUP'] = "true";
$parameters['FRONTEND.MODE'] = "DEFAULT";
$parameters['FRONTEND.LANGUAGE'] = "en";
$parameters['PAYMENT.CODE'] = "CC.DB";

//this is the URL the Web Payment Frontend call on YOUR system
//after the customer finished payment (pay or cancel button)
//please ADJUST the url to YOUR needs.

//we need to collect the data from the start page (POST-parameters) and put it
//into the properties that the web payment frontend expects
//for a complete list of properties see the document
//POST_Integrator_Transactions
$parameters['NAME.GIVEN'] = $_POST['given'];
$parameters['NAME.FAMILY'] = $_POST['family'];
$parameters['ADDRESS.STREET'] = $_POST['street'];
$parameters['ADDRESS.ZIP'] = $_POST['zip'];
$parameters['ADDRESS.CITY'] = $_POST['city'];
$parameters['ADDRESS.COUNTRY'] = $_POST['country'];
$parameters['CONTACT.EMAIL'] = $_POST['email'];
$parameters['PRESENTATION.AMOUNT'] = "99.00";
$parameters['PRESENTATION.CURRENCY'] = "EUR";

//building the postparameter string to send into the WPF
foreach (array_keys($parameters) AS $key) {
    $$key .= $parameters[$key];
    $$key = urlencode($$key);
    $$key .= ";
    $var = strtoupper($$key);
    $value = $$key;
    $result .= "$var=$value";
}

$strPOST = stripslashes($result);

//open the request url for the Web Payment Frontend
$cpt = curl_init();
curl_setopt($cpt, CURLOPT_URL, $url);
curl_setopt($cpt, CURLOPT_SSL_VERIFYHOST, 2);
curl_setopt($cpt, CURLOPT_USERAGENT, "php post");
curl_setopt($cpt, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($cpt, CURLOPT_SSL_VERIFYPEER, FALSE);
curl_setopt($cpt, CURLOPT_POSTFIELDS, $strPOST);

$curlresultURL = curl_exec($cpt);
$curlerror = curl_error($cpt);
$curlinfo = curl_getinfo($cpt);
curl_close($cpt);

// here you can get all variables returned from the processing server (see post integration transactions documentation for help)
//print "$curlresultURL";
// parse results
$r_arr=explode("&",$curlresultURL);
foreach($r_arr AS $buf)
{
    $temp=urldecode($buf);
    $temp=split("=", $temp, 2);
    $postatt=$temp[0];
    $postvar=$temp[1];
    $returnvalue[$postatt]=$postvar;
    //print "<br>var: $postatt - value: $postvar<br>";
}
$processingresult=$returnvalue['POST_VALIDATION'];
$redirectURL=$returnvalue['FRONTEND_REDIRECT_URL'];

// everything ok, redirect to the WPF
if ($processingresult=="ACK")
{
    if (strstr($redirectURL,"http")) // redirect url is returned ==> everything ok
    {
        header("Location: $redirectURL");
    }
    else // error-code is returned ... failure
    {
        header("Location: http://www.merchant.com/error.php");
    }
} // there is a connection-problem to the processing server ... redirect to error page (change the URL to YOUR error page)
else
{
    header("Location: http://www.merchant.com/error.html");
}
?>
### 4.1.2 response-page.php

```php
<?php
//this page is called after the customer finishes payment with the Web Payment Frontend.
//It must be hosted YOUR system and accessible to the outside world.
//It always must respond with a URL that defines which page the WPF should redirect to.
//this new page also MUST be hosted on your system
//AND it must be accessible so that the WPF can redirect the users browser to it.
// PROCESSING.RESULT gets PROCESSING_RESULT when posting back (URL encoding)
$returnvalue=$_POST['PROCESSING_RESULT'];
if ($returnvalue)
{
    if (strstr($returnvalue,"ACK"))
    {
        // URL after successful transaction (change the URL to YOUR success page: e.g. return to shopping)
        print "http://www.merchant.com/success.html";
    }
    else
    {
        // URL error in transaction (change the URL to YOUR error page)
        print "http://www.merchant.com/error.html";
    }
}
?>
```

### 4.2 ASP Code

#### 4.2.1 call-wpf-page.aspx

```csharp
<%@ Import Namespace="System.Net" %>
<%@ Import Namespace="System.IO" %>
<%@ Import Namespace="System.Security.Cryptography.X509Certificates" %>
<script runat="server" language="C#"/>

class CertificatesPolicy : System.Net.ICertificatePolicy
{
    public CertificatesPolicy()
    {
    
    }
    public bool CheckValidationResult(
        ServicePoint _sp,
        X509Certificate _cert,
        WebRequest _req,
        int _problem)
    {
        return true;
    }
}
```
```csharp
void process()
{
    IDictionary parameters = new Hashtable();

    //this URL for the TEST web payment frontend
    String url = "https://test.ctpe.net/frontend/payment.prc";
    //this URL for the LIVE web payment frontend
    //String url = "https://ctpe.net/frontend/payment.prc";

    //these are the parameters that identify the caller
    //to the system. the ones below wont work as they are for
    //a local test only.
    //please contact your sales rep for the login data for your
    //test-account and after successful integration for the
    //LIVE-account data
    parameters["SECURITY.SENDER"] = "ff8080820249b2ac010249b74aa10009";
    parameters["USER.LOGIN"] = "ff8080820249b2ac010249b74aa40015";
    parameters["USER.PWD"] = "demomerchanr";
    parameters["TRANSACTION.CHANNEL"] = "ff8080820249b2ac010249b74aac003a";

    //switch this parameter to "LIVE" when starting the LIVE integration
    parameters["TRANSACTION.MODE"] = "INTEGRATOR_TEST";

    parameters["REQUEST.VERSION"] = "1.0";
    //its at the moment not mandatory to send a unique tx-id to
    //the web payment frontend but it is considered a good practice
    //as you will be able to identify your tx when the WPF calls your
    //result page.
    parameters["IDENTIFICATION.TRANSACTIONID"] = "your tx id";
    parameters["FRONTEND.ENABLED"] = "true";
    parameters["FRONTEND.MODE"] = "DEFAULT";
    parameters["FRONTEND.POPUP"] = "true";
    parameters["FRONTEND.LANGUAGE"] = "en";
    parameters["PAYMENT.CODE"] = "CC.DB";

    //this is the URL the Web Payment Frontend call on YOUR system
    //after the customer finished payment (pay or cancel button)
    //please ADJUST the url to YOUR needs.
    parameters["FRONTEND.RESPONSE_URL"] = "http://shopserver.com/response-page.aspx"

    //we need to collect the data from the start page and put it
    //into the properties that the web payment frontend expects
    //for a complete list of properties see the document
    //POST_Integrator_Transactions
    parameters["NAME.GIVEN"] = Request["given"];
    parameters["NAME.FAMILY"] = Request["family"];  
    parameters["ADDRESS.STREET"] = Request["street"];  
    parameters["ADDRESS.ZIP"] = Request["zip"];  
    parameters["ADDRESS.CITY"] = Request["city"];  
    parameters["ADDRESS.COUNTRY"] = Request["country"];  
    parameters["CONTACT.EMAIL"] = Request["email"];  

    string product = Request["product"];  
    if (product.Equals("ASP/PHP/JSP SUPPORTPACKAGE - 99 USD"))
    {
        parameters["PRESENTATION.AMOUNT"] = "99.00";
        parameters["PRESENTATION.CURRENCY"] = "USD";
    }
    else
```
```csharp
{
    parameters["PRESENTATION.AMOUNT"] = "70.00";
    parameters["PRESENTATION.CURRENCY"] = "EUR";
}

// building the postparameter string to send into the WPF
// probably there is a smarter way to do that.
string request_data = "";
foreach (DictionaryEntry de in parameters)
{
    request_data += string.Format("{0}={1}&", de.Key, de.Value);
}
request_data = request_data.Substring(0, request_data.Length - 1);

// this is a work around so that we can connect to a SECURE http
// server. it is definitely better to import the certificate and
// make it available instead of circumventing the security as done here
ServicePointManager.CertificatePolicy = new CertificatesPolicy();

// open the request url for the Web Payment Frontend
HttpWebRequest request = (HttpWebRequest) WebRequest.Create(url);
request.Method = "POST";
request.ContentType = "application/x-www-form-urlencoded";

    // send the request data
    Stream input = request.GetRequestStream();
    byte[] data = System.Text.Encoding.ASCII.GetBytes(request_data);
    input.Write(data, 0, data.Length);
    input.Close();

    // get the answer from the frontend server
    HttpWebResponse response = (HttpWebResponse)request.GetResponse();
    StreamReader sr = new StreamReader(response.GetResponseStream(), Encoding.ASCII);
    string initial_response = sr.ReadToEnd();
    sr.Close();
    response.Close();

    // parse the data and split it into properties
    // probably there is also a better way to do that
    string[] initial_response_parts = initial_response.Split('&');
    IDictionary props = new Hashtable();
    foreach (string response_part in initial_response_parts)
    {
        string[] key_value = response_part.Split('="');
        props[key_value[0]] = key_value[1];
    }

    // if the frontend respondend with an ACKNOLEDGE redirect
    // the customers browser to the the graphical form
    // of the web payment frontend
    if (props["POST.VALIDATION"].Equals("ACK"))
    {
        string redirect = (string) props["FRONTEND.REDIRECT_URL"];
        redirect = redirect.Replace("%2F", "/");
        redirect = redirect.Replace("%3A", ":");
        redirect = redirect.Replace("%3D", ";");
        redirect = redirect.Replace("%3D", ";");
        Response.Redirect(redirect);
    }
```
4.2.2 response-page.aspx

<script runat="server" language="C#">
    //this page is called after the customer finishes payment with the Web Payment Frontend.
    //It must be hosted YOUR system and accessible to the outside world.
    //It always must respond with a URL that defines which page the WPF should redirect to.
    //this new page also MUST be hosted on your system
    //AND it must be accessible so that the WPF can redirect the users browser to it.
    void process() {
        //read the processing result value so that you can decide what to do.
        string value = Request["PROCESSING.RESULT"];  

        //if the payment was acknowledged redirect to the success page
        if (value.Equals("ACK")) {
        }
        //if not redirect to an error page
        else {
            Response.Write("http://shopserver.com/error-page.aspx");
        }

        //of course your processing at this page can be more complex
        //you could for example send an email to the customer to notify him of the successful payment or you could notify your staff at the warehouse to pack and send the paid product to the customer. only your imagination is the limit.
    }
</script>
<% process(); %>
5 Request Encoding

For all payment requests containing shopping and possibly payment data, the request header must contain the Content-Type / charset parameter with the charset encoding set to "UTF-8". The actual content type may differ; the decisive information is the charset value.

Accordingly, all request data must be encoded using the UTF-8 character set.

For POST requests, you can use:
Content-Type: application/x-www-form-urlencoded; charset=UTF-8

Examples:

**Integration via PHP/cURL:**  http://php.net/manual/de/book.curl.php

```php
$ch = curl_init();
curl_setopt($ch, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/x-www-form-urlencoded; charset=UTF-8"
));
// attach parameter
curl_exec($ch);
```

**Integration via Java:**

```java
UrlRequest req = new UrlRequest(UrlRequest.POST, CORE_URL );
// attach parameter to request
req.addHeaderParam("Content-Type", " application/x-www-form-urlencoded; charset=UTF-8");
req.send();
```
6 WPF Request

Besides the standard POST parameters that can (but do not have to) be submitted a few frontend specific parameters can be sent to the Web Payment Server. These parameters are listed in the "POST Transactions" document.

For all transactions replace the following parameters with your test or live parameters:

- **SECURITY.SENDER**
- **USER.LOGIN**
- **USER.PWD**
- **TRANSACTION.CHANNEL**

See the "Technical Quick Start" document for more details.

A typical sample request for the Web Payment Frontend contains parameters like the following:

- **REQUEST.VERSION=1.0**
- **TRANSACTION.CHANNEL=0000.0000.0000**
- **IDENTIFICATION.TRANSACTIONID=1234567890**
- **TRANSACTION.MODE=INTEGRATOR_TEST**
- **PRESENTATION.AMOUNT=298.00**
- **PRESENTATION.CURRENCY=EUR**
- **SECURITY.SENDER=4028fb19fabf01a200fabf01abae0019**
- **USER.LOGIN=2328fb19fa2345200fabf01abaeeeee**
- **USER.PWD=geheim**
- **FRONTEND.ENABLED=true**
- **FRONTEND.POPUP=true**
- **FRONTEND.MODE=DEFAULT**
- **FRONTEND.RESPONSE_URL=http://www.merchantshop.com/paymentResult;jsessionid=1234321543243214213**
- **FRONTEND.LANGUAGE=de**
- **FRONTEND.LINK.1.KIND=TERMS**
- **FRONTEND.LINK.1.LINK=https://www.merchantshop.com/agb.php**
- **FRONTEND.LINK.1.AREA=EMBEDDED_CHECKBOX**
- **FRONTEND.LINK.1.POPUP_WIDTH=320**
- **FRONTEND.LINK.1.POPUP_HEIGHT=400**

The Web Payment Server will respond the URL to redirect to this POST request and a validation code of the request.

The result typically looks like this:

FRONTEND.REDIRECT_URL=https://test.ctpe.net/frontend/startFrontend.prc;jsessionid=12345435643543435&POST.VALIDATION=ACK
The list of possible validation codes is available in the document “POST Transactions”.

The most important parameters (and the most likely you need to change depending on your requirements) are:

- **FRONTEND.ENABLED**
  Set this to true, in case you are using the WPF. Setting this parameter to false means you are doing direct payment via the POST interface (check document “POST Transactions” for details)

- **FRONTEND.POPUP**
  Set this to “true” to let the WPF appear inside a popup window, or to false if you want to have it embedded.

- **PAYMENT.CODE**
  By default, “CC.DB” (Credit Card Debit) will be used for the WPF. In case you want to do a Preauthorization instead of a Debit the code has to be CC.PA. For a registration process use CC.RG.
  For a complete list of payment codes, check the document “POST_Transactions”.

- **FRONTEND.RESPONSE_URL**
  The URL where the payment system shall post the payment response of the payment to. This is not the URL to which the end user's browser will be redirected at the end! The redirect URL at the end must be replied to the call to this URL. This allows the shop to dynamically react on the result of the payment and make the decision where to redirect the user to at the end of the whole process.

In general you can send any parameter you like (also not documented parameters!) and it will be returned to you at the end of the process as part of the payment response message.
7 WPF Asynchronous Response

The Web Payment Server (WPS) will communicate the payment result back to the merchant’s shop when the user has finished or cancelled the payment process. The target URL where the result gets posted to gets sent to the WPS in the initial request.

A typical response looks like this:

- RESPONSE.VERSION = 1.0
- TRANSACTION.MODULE = INTEGRATOR_TEST
- TRANSACTION.RESPONSE = SYNC
- TRANSACTION.CHANNEL = 0000.0000.0000
- IDENTIFICATION.TRANSACTIONID = MerchantAssignedID
- IDENTIFICATION.UNIQUEID = 402880e5faf35d0700faf35d0cec0002
- IDENTIFICATION.SHORTID = 1101.9571.9800
- PAYMENT.CODE = CC.DB
- PRESENTATION.AMOUNT = 298.00
- PRESENTATION.CURRENCY = EUR
- PRESENTATION.USAGE = Order Number 1234
- CLEARING.AMOUNT = 298.00
- CLEARING.CURRENCY = EUR
- CLEARING.DESCRIPTOR = 1101.9571.9800 - Order Number 1234
- PROCESSING.CODE = CC.DB.90.00
- PROCESSING.TIMESTAMP = 2004-02-26 20:46:26
- PROCESSING.RESULT = ACK
- PROCESSING.STATUS = NEW
- PROCESSING.STATUS.CODE = 90
- PROCESSING.REASON = Successful System Entry
- PROCESSING.REASON.CODE = 00
- PROCESSING.RETURN = Request successfully processed in Integrator Test Mode
- PROCESSING.RETURN.CODE = 000.100.110
- FRONTEND.MODE = DEFAULT
- FRONTEND.REQUEST.CANCELLED = false
- SECURITY.HASH=2d0ec783cb7c2d5b117499e6211caef4
ATTENTION: This is how the response is sent to the merchant’s server. Because of the URL encoding of the response, the parameters you receive might be slightly changed. “.” Is converted to ‘_’, e.g. PROCESSING结果 becomes PROCESSING_RESULT

7.1 Validation of the Response Parameters (optional)

The parameter HASH contains a generated SHA-1 hash of the following response parameters:

- PAYMENT.CODE
- IDENTIFICATION.TRANSACTIONID
- IDENTIFICATION.UNIQUEID
- CLEARING.AMOUNT
- CLEARING.CURRENCY
- PROCESSING.RISK_SCORE
- TRANSACTION.MODE
- PROCESSING.RETURN.CODE
- PROCESSING.REASON.CODE
- PROCESSING.STATUS.CODE
- and the “secret” generated while the Merchant Account setup and only the merchant knows

To guarantee the integrity of the response parameters and to make sure they have not been manipulated you have to regenerate the SHA-1 hash value and compare it with the received HASH.

The String to generate the hash is built like this:

```
PAYMENT.CODE + "|" + IDENTIFICATION.TRANSACTIONID + "|" + IDENTIFICATION.UNIQUEID + "|" + CLEARING.AMOUNT + "|" + CLEARING.CURRENCY + "|" + PROCESSING.RISK_SCORE + "|" + TRANSACTION.MODE + "|" + PROCESSING.RETURN.CODE + "|" + PROCESSING.REASON.CODE + "|" + PROCESSING.STATUS.CODE + "|" + secret
```

For the sample response above the String looks like this (sample secret is “abcd1234”):

```
"CC.DB|MerchantAssignedID|402880e5fafe35d0700fafe35e2cc00002|000.100.110|298.00|EUR||INTEGRATOR_TEST|90|00|abcd1234"
```

**Note:** If one of the parameters above is not part of the response message (in the example this is PROCESSING.RISK_SCORE), you have to set an empty string instead.

For this string a SHA-1 hash needs to be generated and compared with the received HASH.

In case your hash and the received HASH parameter are not the same you can assume that the response parameters were manipulated.

The generation of the hash in Java would look like this:

```
// digest the string with SHA-1
byte[] digest = MessageDigest.getInstance("SHA-1").digest(stringToDigest.getBytes("UTF-8"));
StringBuilder sBuffer = new StringBuilder();
for (byte b : digest)
{
    // every byte is separated in two parts and converted to HEX string
    // upper byte part is converted to hex string like this
    sBuffer.append(Integer.toHexString((0xF0&b)>>4));
    // lower byte part is converted to hex string like this
```

sBuffer.append(Integer.toHexString(0x0F&b));
}
String hash = sBuffer.toString();
// generated hash with the String above is “2d0ec783cb7c2d5b1l7499e6211caef4”

Full sample Java and PHP code is available in chapter 13.

7.2 Merchant Response to the WPF Asynchronous Response

The shop must respond on the payment response with a URL to redirect the user back into the shop. The payment system then redirects the end user’s browser to this URL. **This URL must be replied as plain text to the WPF Asynchronous Response.** The response of the merchant must not contain any other information than this URL:
- no HTML header
- no other text

**JUST THE URL ONLY!**

Typically it just looks like
http://myshop.com/scripts/continue_shopping;jsessionid=32443q54325432

In any case the payment server will try to redirect the user to the URL specified in the response. Therefore this will lead to errors in the end user’s browser if your script contains errors (to simulate such an error, try to redirect to something that is not a URL).

If you want to receive the account information the user entered in the WPF, send in the parameter “FRONTEND.RETURN_ACCOUNT=true” with your initial request. You will then receive the account parameters (check chapter “Account Group” in the document “POST Transactions”) in the payment response. Of course you will receive the parameters in accordance with the PCI regulations, i.e. in a masked form.
8 Registration with WPF

If you want to use the WPF to register customer and account information only, just send in the payment code <Payment Method>.RG (for example CC.RG or DD.RG with your initial request within the parameter PAYMENT.CODE.

If the user is registered in the system, you can debit his account by simply specifying the UniqueId (parameter IDENTIFICATION.UNIQUEID) you got back in the registration response in future payments. To be able to do so, store the IDENTIFICATION.UNIQUEID together with the user record in your database.

Payment can then be done without the WPF by simply using the POST transaction interface. You switch into direct payment mode by sending in FRONTEND.ENABLED=false. The unique id of the registration needs to specified within the parameter ACCOUNT.REGISTRATION.

You can also reregister a customer by sending in the UniqueId of the registration with IDENTIFICATION.REFERENCEID and specifying the payment code <Payment Method>.RR (for example CC.RR or DD.RR with your initial request within the parameter PAYMENT.CODE.

Find more details about payment with the POST Transaction interface in the document “POST Transactions”.
9 WPF UI Configuration

There are two parameters that allow you to change the Look & Feel and the behaviour of the WPF:
- FRONTEND.CSS_PATH: to override default style sheet settings
- FRONTEND.JSCRIPT_PATH: allows to do initial setup or changes via Java Script

9.1 Add your own CSS

Using the parameter FRONTEND.CSS_PATH you can pass in a path to your own CSS:
Example: FRONTEND.CSS_PATH = https://www.merchant-page.com/merchant.css

-NB: the path has to be HTTPS.

The following CSS shows an example for the content of a custom CSS:

```html
html, body {background-color: #ffffff;
    scrollbar-face-color: #red;}
td {color: #000000; }  
a:link { color: #red; text-decoration: none; } 
a:visited {color: #orange; text-decoration: none; }
a:hover { color: #grey; text-decoration: underline; }  
a:active { color: #orange; text-decoration: none; }  
.bar {background-color: #ffffff;}
.frm_box { border-style:ridge; border-width:1; border-color:orange; font-size: 13px; background-color: #ffffff; font-style:normal; font-variant:normal; font-weight:normal; color:#000000; margin-left:4px;margin-right:4px; padding-left:4px; padding-right:4px; padding-top:1; padding-bottom:1; }
.tab {background-color:#ffffff; border-style: none;}
```
The resulting WPF payment page looks like this:

![WPF Payment Page](image)

**Figure 7 WPF Payment Page**

The following list provides an overview over all the available styles that can be changed:

<table>
<thead>
<tr>
<th>CSS Styles</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>html, body</td>
<td>Default settings for the page</td>
</tr>
<tr>
<td>td</td>
<td>Default settings for table cells</td>
</tr>
<tr>
<td>a (a:link, a:visited, a:hover, a:active, a:button, a:button:link, a:button:visited)</td>
<td>Default settings for links</td>
</tr>
<tr>
<td>.bar</td>
<td>Bar styles on top and bottom of payment page</td>
</tr>
<tr>
<td>input</td>
<td>Input fields</td>
</tr>
</tbody>
</table>
select | Select fields
.text_bold | Bold text
.text_small | Small text
.btn | Default input buttons

The default CSS can be downloaded from https://test.ctpe.net/frontend/scripts/css/0.css.

9.2 Change default behaviour with Java Script

ATTENTION: If you are using FRONTEND.MODE=WPF_LIGHT (as described in chapter 11), the content of the chapter at hand does not apply!

The parameter FRONTEND.JSCRIPT_PATH allows you to define a JavaScript file that can be used to change the default behaviour of the WPF:
Example: FRONTEND.JSCRIPT_PATH = http://www.merchant-page.com/merchant.js
The Javascript file must contain a method called
  function init()
which is called automatically when the page is loaded.

A sample file could look like this:

```javascript
function init()
{
    // The method “getElem” can be used to get an item on the page by its id
    getElem('id', 'contactBlock', 0).style.display="none";
    getElem('id', 'userInfoBlock', 0).style.display="none";
    getElem('id', 'addressBlock', 0).style.display="none";
}
```
This code would result in the following appearance of the WPF:

![Web Payment Frontend](image)

Figure 8 WPF Payment Page without Customer data

In this example the Java Script code is used to hide a number of fields on the payment page. The method “getElem()” can be used to find a component by its ID.

The following table shows a list of the IDs of all available components on the payment page.

<table>
<thead>
<tr>
<th>IDs</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userInfoBlock</td>
<td>Block with all the user information</td>
</tr>
<tr>
<td>salutationRow</td>
<td>First row of the user information block, containing salutation and first name</td>
</tr>
<tr>
<td>familyNameRow</td>
<td>Second row of the user information block, containing the family name</td>
</tr>
<tr>
<td>companyNameRow</td>
<td>Third row of the user information block, containing the company name</td>
</tr>
<tr>
<td>addressBlock</td>
<td>Block with address information</td>
</tr>
<tr>
<td>streetRow</td>
<td>First row of the address block, containing the street name</td>
</tr>
<tr>
<td>cityRow</td>
<td>Second row of the address block, containing the city name</td>
</tr>
<tr>
<td>countryRow</td>
<td>Third row of the address block, containing the country</td>
</tr>
<tr>
<td>stateRow</td>
<td>Fourth row of the address block, containing the state</td>
</tr>
<tr>
<td>contactBlock</td>
<td>Block with contact information</td>
</tr>
<tr>
<td>emailRow</td>
<td>First row of the contact block, containing the email address</td>
</tr>
<tr>
<td>phoneRow</td>
<td>Second row of the contact block, containing the phone number</td>
</tr>
<tr>
<td>mobileRow</td>
<td>Third row of the contact block, containing the mobile number</td>
</tr>
<tr>
<td>paymentSelection</td>
<td>Payment Type Selection Row, typically containing a radio button selection with “Credit Card” and “Direct Debit” to choose from</td>
</tr>
<tr>
<td>ccBlock</td>
<td>Block with all the Credit Card info</td>
</tr>
<tr>
<td>ddBlock</td>
<td>Block with all the Direct Debit info</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>notMandatoryRow</td>
<td>Row containing the information which fields are not mandatory</td>
</tr>
</tbody>
</table>

It is also possible to use style definitions within your custom CSS to show and hide fields or rows. The following example shows how to hide the full address block:

```css
#addressBlock {display:none;}
```

In case you want to change the style of the CVV code row, this could look like this:

```css
#cvvRow
{color:#303030;background-color:#F7F7F7;
border-color:#DEDEDE; border-style:solid solid solid none;
border-width:1px 1px 1px 0pt; font-size:11px; margin:0px 0px 3px;
padding:3px; width:425px; }
```
10 Other Configuration Options

This chapter describes the most important configuration options which are of interest to most system users. See the “POST_Transactions” document (chapter “1.20 Frontend Group”) for a list of all possible parameters.

10.1 Languages

You can optionally use the FRONTEND.LANGUAGE parameter to preselect the displayed language. Default language is English (en).

Possible language codes are (ISO 3166):

de=Deutsch
en=English
it=Italiano
es=Espanol
fr=Francais
nl=Nederlands
fi=Suomksi
sl=Slovenscina
pt=Português
pl=Polski
se=Svenska
sk=Slovencina
cz=Cesky
hu=Magyar Nyelv
ja=Japanese
bg=Bulgarian
ro=România
no=Norway
dk=Denmark
ru=Russian Federation
tr=Turkey
zh=Chinese

10.2 Banners

**ATTENTION:** If you are using FRONTEND.MODE=WPF_LIGHT (as described in chapter 11), the content of the chapter at hand does not apply!

You can optionally use the area on top or the bottom of the WPF as banner areas to display any link you like. To do this, use the FRONTEND.BANNER parameters.
The following example shows how to display a URL in the top area with a height of 80px:

FRONTEND.BANNER.1.LINK=https://myserver.com/banner1.html
FRONTEND.BANNER.1 AREA=TOP
FRONTEND.BANNER.1.HEIGHT=80

10.3 Hide or Show Payment Methods and Types

**ATTENTION: If you are using FRONTEND.MODE=WPF_LIGHT (as described in chapter 11), the content of the chapter at hand does not apply!**

You can optionally restrict the payment methods and payment subtypes a end user can see and select from in the WPF.

Typically a merchant’s account is set up for a set of payment methods on the payment server. By default all these methods are selectable for a WPF user. If there has already been a pre-selection of the payment method on the merchant’s site, a merchant might want to prohibit the end user to change this in the WPF. Therefore it is possible to restrict certain payment methods in the WPF.

The following example shows how to show “German and Austrian direct debit only” to the WPF user:

FRONTEND.PM.DEFAULT_DISABLE_ALL=true
FRONTEND.PM.1.METHOD=DD
FRONTEND.PM.1.ENABLED=true
FRONTEND.PM.1.SUBTYPES=AT,DE

Another example shows how to display “Credit Cards only” to the end user

FRONTEND.PM.DEFAULT_DISABLE_ALL=true
FRONTEND.PM.1.METHOD=CC
FRONTEND.PM.1.ENABLED=true
FRONTEND.PM.1.SUBTYPES=VISA,AMEX

If you do not specify any information with these parameters all payment methods configured on the payment server are selectable to the end users.

Another option is to make sure that certain subtypes are not displayed. The following example makes sure that all Direct Debit types but not Germany is displayed:

FRONTEND.PM.DEFAULT_DISABLE_ALL=true
FRONTEND.PM.1.METHOD=DD
FRONTEND.PM.1.ENABLED=true
FRONTEND.PM.1.SUBTYPES_DISABLED=DE

If you define a method with ENABLED=true, you are required to submit the subtypes you want to have available!

10.4 Links Configuration

**ATTENTION: If you are using FRONTEND.MODE=WPF_LIGHT (as described in chapter**
11. the content of the chapter at hand does not apply!

External links can be integrated into the WPF to display things like Terms and Conditions, copyright info, external help pages and such. The links can either be displayed embedded right above the “Pay” button as part of a checkbox (e.g. where the user accepts the terms and conditions) or in the status bar at the bottom.

The following example shows how to integrate terms and conditions into the page. The user has to check a checkbox to accept the terms before he can commit the payment transaction. The terms are opened in a popup with the size of 320x400 pixels.

FRONTEND.LINK.1_KIND=TERMS
FRONTEND.LINK.1_LINK=https://www.merchantshop.com/agb.php
FRONTEND.LINK.1_AREA=EMBEDDED_CHECKBOX
FRONTEND.LINK.1_POPUP_WIDTH=320
FRONTEND.LINK.1_POPUP_HEIGHT=400

10.5 WPF Size Configuration

Use the following two parameters to define the size of the WPF as it is supposed to appear:

- FRONTEND.FORM_WIDTH
- FRONTEND.HEIGHT

By default the two parameters are in pixels, meaning a value of 100 results in 100px. You can also send the sizes in % by specifying the percentage amount as well.

Example:

- FRONTEND.FORM_WIDTH = 100%
- FRONTEND.HEIGHT = 450

10.6 Change Button Appearance

ATTENTION: If you are using FRONTEND.MODE=WPF_LIGHT (as described in chapter 11), the content of the chapter at hand does not apply!

The buttons on the payment page and on the confirmation page can be replaced by images located on the server of the merchant (or located anywhere else). By default the buttons are standard buttons. Their general appearance can be changed via CSS as well.

The following sample parameters change the buttons on the payment page to be images:

FRONTEND.BUTTON.1_NAME=PAY
FRONTEND.BUTTON.1_TYPE=IMAGE
FRONTEND.BUTTON.1_LINK= https://www.merchantshop.com/images/pay.gif
FRONTEND.BUTTON.2_NAME=CANCEL
FRONTEND.BUTTON.2_TYPE=IMAGE
FRONTEND.BUTTON.2_LINK= https://www.merchantshop.com/images/cancel.gif
10.7 Immediate Redirect after Payment / Registration

If you do not want the end user to see the payment result or cancellation page at the end, just set the redirect time to 0. This will lead to an immediate redirect back into the shop.

Example:
FRONTEND.REDIRECT_TIME=0
11 WPF LIGHT

If you want to make sure that the WPF also works on browsers with JavaScript disabled, you can use the WPF in the mode "WPF_LIGHT".

Technically, the workflow is exactly the same as with standard WPF. The following parameters must be sent in for WPF_LIGHT:

- `FRONTEND.MODE = WPF_LIGHT`
- `PAYMENT.CODE = <your payment code>`
- `ACCOUNT.COUNTRY = country of the bank account in case your payment code starts with DD`

This means, for WPF light you must let the end user choose the payment method (and the country for Direct Debit) on your page before calling the WPF. On the WPF no changes are possible any more!

The following parameters will be ignored for WPF_LIGHT:

- `FRONTEND.POPUP (is always false)`
- `FRONTEND.LANGUAGE_SELECTOR (no language selection possible)`
- `FRONTEND.LINK.*`
- `FRONTEND.JSCRIPT_PATH`
- `FRONTEND.STATUSBAR_VISIBLE`
- `FRONTEND.BANNER.*`
- `FRONTEND.PM.*`
12 FAQ

See http://support.ctpe.net/faq/ for all FAQs available.

Why not simply call the web payment URL from the browser and directly show the payment frontend?
There are several reasons why this is not working. First of all this process is very much security driven. The authentication service of the web payment frontend only allows payment requests for a merchant that are sent from a specific IP or IP range, in this case the IP of the merchant's shop server. Therefore nobody else can initiate a payment process for the merchant.
Secondly, unlike using the standard XML or POST integrator, this is an asynchronous payment method. This means the payment result has to be communicated back to the merchant’s server. The merchant server needs to know if the payment was successful or not. Again this is happening via Server-to-Server communication. It would be very unreliable to somehow communicate this information back via the browser.

The payment window should have the same look and feel like my shop, what can I do?
The web payment frontend can be configured in various ways. See chapter 6 of this document for more details.

Is there a demo shop available for other programming languages?
Currently the demo shop is fully developed in Java. There are other implementations planned as well, however, the document “Integration Packages” contains integration instructions for the XML and the POST integrator in all kind of programming languages. Since the POST integrator is the base of the Web Payment Frontend this document will enable you to integrate the payment services in other languages without encountering major challenges.

Do I need to send Account information for the WPF integration?
No. This is not necessary and not recommended. The account information (Credit Card or Bank Accounts) are entered by the end user in the WPF. If you are able to send the account information as well, WPF is probably not the right solution for you, as you already got the payment data for the user. However, you can use the parameter PAYMENT.CODE to pre-select a payment method for the end user. If you prefer that the end user is paying with his bank account, send PAYMENT.CODE=DD.DB, if you prefer him paying with his credit card, send PAYMENT.CODE=CC.DB to the server.

How do I know if the payment transaction was successful or not?
WPF is based on an asynchronous process. This means, after sending your initial request to the server, you are loosing control of the process. However, after the end user has entered his payment data in the Web Payment Frontend, the payment server sends the result of the payment to the URL you initially specified in the parameter FRONTEND. One of the parameters you receive as part of this response request is called PROCESSING.RESULT. If PROCESSING.RESULT is ACK, the transaction was successful, if it is NOK the transaction failed.

The end user has finished payment. Now how does he get back to my page?
After payment is finished, the payment server sends the payment result to the URL you initially specified in the parameter FRONTEND.RESPONSE_URL. Simply respond to this request with the URL you want the end user to be redirected to. You can write this URL as plain text to the output stream of your servlet or script. The payment server will read the response and redirect the user's browser to
this URL. This process gives you the chance to redirect the end user to different pages depending on the payment result.

**I get a Javascript Error on the “payment-finished” page, what is the problem?**

In this case it is very likely that the URL you returned as a response to the payment result message posted to your system is not valid.

The payment-finished page contains a Javascript-Snippet similar to the following:

```javascript
function gotoMerchantSite()
{
    parent.location.href = “http://myshop.com/thankyou.html”
}
```

The URL that is used in this method is the one we received back from you after posting the payment result to your response URL. If this URL is not valid (or contains “) it is possible that you get a Javascript error.
13 SHA-1 Hash generation

These source code snippets show the usage of SHA-1 Hash generation to validate the asynchronous response message.

13.1 Java Sample Code

```java
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Properties;

public class SampleWPFHashUtil
{
    public static void main(String[] args)
    {
        Properties p = new Properties();
        p.setProperty("PAYMENT.CODE", "CC.DB");
        p.setProperty("IDENTIFICATION.TRANSACTIONID", "MerchantAssignedID");
        p.setProperty("IDENTIFICATION.UNIQUEID", "402880e5f935d075f5af35d0ce0002");
        p.setProperty("CLEARING.AMOUNT", "298.00");
        p.setProperty("CLEARING.CURRENCY", "EUR");
        p.setProperty("PROCESSING.RISK_SCORE", "90");
        p.setProperty("TRANSACTION.MODE", "INTEGRATOR_TEST");
        p.setProperty("PROCESSING.REASON.CODE", "90");
        p.setProperty("PROCESSING.STATUS.CODE", "90");

        String hash = buildHashFromRequestProperties(p, "abcd1234");
        System.out.println("generated hash code: " + hash);
    }

    public static String buildHashFromRequestProperties(Properties _responseProperties, String _secret)
    {
        String _paymentCode = replaceIfEmpty(_responseProperties.getProperty("PAYMENT.CODE"));
        String _txID = replaceIfEmpty(_responseProperties.getProperty("IDENTIFICATION.TRANSACTIONID"));
        String _uuid = replaceIfEmpty(_responseProperties.getProperty("IDENTIFICATION.UNIQUEID"));

        String _amount = replaceIfEmpty(_responseProperties.getProperty("CLEARING.AMOUNT"));
        String _currency = replaceIfEmpty(_responseProperties.getProperty("CLEARING.CURRENCY"));
        String _riskScore = replaceIfEmpty(_responseProperties.getProperty("PROCESSING.RISK_SCORE"));
```
String _txMode = replaceIfEmpty(_responseProperties.getProperty("TRANSACTION.MODE"));
String _returnCode = replaceIfEmpty(_responseProperties.getProperty("PROCESSING.RETURN.CODE"));
String _reasonCode = replaceIfEmpty(_responseProperties.getProperty("PROCESSING.REASON.CODE"));
String _statusCode = replaceIfEmpty(_responseProperties.getProperty("PROCESSING.STATUS.CODE"));

return makeHash(_paymentCode, _txID, _uuid, _amount, _currency, _riskScore, _txMode, _returnCode, _reasonCode, _statusCode, _secret);

{
    String stringToDigest = _paymentCode + "|" + _txID + "|" + _uuid + "|" + _amount + "|" + _currency + "|" + _riskScore + "|" + _txMode + "|" + _returnCode + "|" + _reasonCode + "|" + _statusCode + "|" + _secret;
    try
    {
        return DigestUtils.shaHex(stringToDigest);
    }
    catch (NoSuchAlgorithmException ex)
    {
        throw new IllegalStateException("NoSuchAlgorithmException, SHA-1 missing");
    }
    catch (UnsupportedEncodingException ex)
    {
        throw new IllegalStateException("UnsupportedEncodingException, UTF-8 missing");
    }
}

private static String replaceIfEmpty(String _paymentCode)
{
    if(_paymentCode == null || _paymentCode.trim().isEmpty())
    {
        return "";
    }
    return _paymentCode;
}

13.2 PHP Sample Code
<?php
$data = array(
    'CC.DB', // PAYMENT.CODE
    'MerchantAssignedID', // IDENTIFICATION.TRANSACTIONID
    '402880e5faf35d0700faf35d00e0002', // IDENTIFICATION.UNIQUEID
    '298.00', // CLEARING.AMOUNT
    'EUR', // CLEARING.CURRENCY
    '100', // PROCESSING.RISK_SCORE
    'INTEGRATOR_TEST', // TRANSACTION.MODE
    '000.100.110', // PROCESSING.RETURN.CODE
    '90', // PROCESSING.REASON.CODE
    '80', // PROCESSING.STATUS.CODE
    'abcd1234' // secret
);
$data_implode = implode('|', $data);
$hash = sha1($data_implode);
echo $hash;
?>

+++ end of document +++