PLATE-FORMES LIBRES *AMP

CYBER PAIEMENT / CYBER PLUS PAIEMENT

GUIDE DE MIGRATION
FACILE ET EN UNE HEURE

PLATE-FORME SYSTEMPAY

Référence	8-002A-10-GMIG01/02
Version	2
Auteur	V. Graux
Date	30/05/10
Dernière page	20

Historique du Document

TABLEAU DE MISE A JOUR				
IND	DATE	PAGES MODIFIEES	SOMMAIRE DE LA MODIFICATION	MOTIF, REFERENCE DE L'ORIGINE
02	30/05/10	4, 12, §5.2 et §6	modification code sur retour prise en compte de mises à jour de la documentation Systempay	Lyra Networks
01	24/05/10	toutes	Document initial 	n.a.
Cyber (Plus) Paiement : Guide de Migration facile				8-002A-10-GMIG01/02

Validation/Approbation du Document

Indice	Rédaction	Approbation Technique	Validation Qualité	Autorisation de Diffusion
2	Nom : V. Graux (VGR) Concepteur-Analyste Date : 24/05/2010	Nom : N/A Date :	Nom : Systempay Date :	Nom : Systempay Maître d'Oeuvre Date :
	Visa	Visa	Visa	Visa

Sommaire

Table des matières

1. But du document....4
2. Le cadre du problème....4
3. Une solution Cyber P@yment....6
4. La migration vers Cyber Plus Paiement....10
5. Les nouveaux scripts....14
5.1) erreurcpp.php : la page appelée en cas d'erreur dans la requête envoyée....14
5.2) retourcpp.php : appel "de serveur à serveur"....17
6. Autres éléments mal couverts par la documentation....20
6.1 Le logo de la boutique....20
6.2 la mise en place....20
6.3 les tests....20
....20

1. But du document.

Le présent document est le 8-002A-10-GMIG01 ; c'est le Guide de Migration (facile et en une heure) qui va à l'essentiel pour migrer une solution de paiement à base de Cyber P@yment vers Cyber Plus Paiement, la nouvelle plate-forme groupe "Systempay" destinée à la remplacer à l'été 2010.

Issu de l'expérience, il démontre que malgré les différences techniques, les deux systèmes peuvent assez facilement se substituer l'un à l'autre et que la migration de l'un vers l'autre d'un formulaire de paiement en ligne peut se faire en une heure.

Il s'adresse aux développeurs ouaibe (ou VARs) ayant en charge le site marchand, et qui n'ont pas utilisé de solutions "boîtes noires" comme osCommerce [3] mais du PHP pur et simple.

Après avoir présenté le problème dans sa généralité, nous présenterons la migration par l'exemple d'un formulaire Cyber P@yment vers la nouvelle plate-forme.

Le lecteur peut s'affranchir pour l'heure de lire les documents essentiels fournis comme documentation par Systempay, à savoir :

- [1] "Guide de démarrage Cyberplus Paiement - V1.2" , Natixis paiements, 2009, 44 pages
- [2] "Descriptif de l'interface avec la page de paiement - version 1.8", Lyra-network, 26/03/2010, 37 pages
- [3] "intégration du module de paiement pour la plateforme osCommerce", Lyra-network, v 2.3a, (19/04/2010 17:40), 9 (12) pages

Ce document lui donnera directement les références utiles dans ces documents, tout en pointant leurs inexactitudes éventuelles ; par exemple le code PHP présenté page 20 de [2] est insuffisant.

2. Le cadre du problème

La solution Cyber P@yment de la Banque Populaire Lorraine-Champagne (BPLC) est une solution du début des années 2000, similaire à CyberMut du Crédit Mutuel et qui sont à elles deux l'essentiel des solutions de paiement installées en France. La solution CyberMUT a d'ailleurs disposé assez tôt de sa propre API, l'extension cybermut ajoutée en standard dans la version 4.0.6 de PHP (23/06/2001) puis déplacée en PECL en décembre 2002. Pour les deux solutions, l'utilisation d'une extension n'est pas nécessaire et du code PHP très simple suffit.

Cyber Plus Paiement est la nouvelle solution BPLC ; la migration est obligatoire (*cf* mél du 17/03/2010 annonçant la fin de Cyber P@yment) à l'été 2010.

Les deux solutions BPLC fonctionnent de manière similaire :

- les deux acceptent toutes les cartes,
- les deux fonctionnent en HTTPS sécurisé,
- les deux offrent une interface de gestion de la boutique du cyber-marchand et
- les deux disposent de mécanismes de sécurité,

mais l'une fonctionne avec un appel en GET et l'autre avec un formulaire en POST, ce qui complique le travail du développeur lorsque le formulaire de paiement n'appelle pas directement la plate-forme (ce qui est le cas, la signature devant être calculée en fonction du montant et de l'horodatage de la transaction).

Evidemment, les deux solutions ne sont pas non plus compatibles de manière ascendante dans le nommage des champs à transmettre.

D'autre part, le nouveau système "Systempay" offre l'intéressante particularité d'un appel "de serveur à serveur" (sic) pour disposer du réel état de la transaction au niveau bancaire (pas seulement "paiement accepté" ou "paiement refusé").

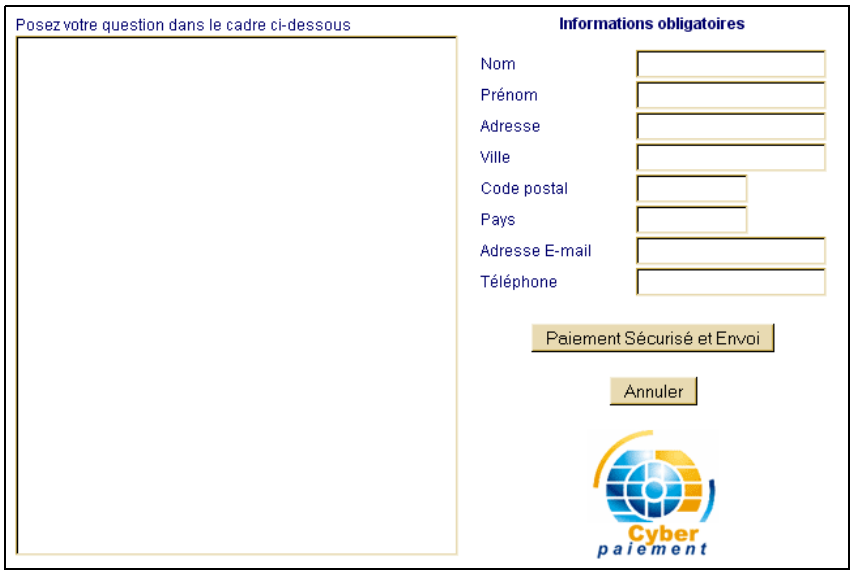
Le nouveau système permet aussi la gestion du "3D Secure", pour les banques qui demandent à leur client (le "porteur de carte") sa date de naissance comme "sécurité" supplémentaire avant d'accepter d'honorer le paiement demandé.

Le nouveau système permet enfin un appel d'une adresse spéciale pour les erreurs dans la requête, découlant des données du formulaire (on obtient donc la cause précise du refus essuyé à l'entrée sur la plate-forme Systempay).

Ceci explique qu'il existe de "nouveaux scripts" (voir section 5) en plus de l'adaptation des scripts existants (sections 3 et 4).

3. UNE SOLUTION CYBER P@YMENT

Voici la forme la plus simple de pages accédant à la plate-forme de paiement Cyber P@yment : un formulaire de paiement en ligne appelé question.php et, comme la requête à Cyber P@yment doit se faire en GET et qu'il vaut mieux cacher cet appel, une page d'envoi effectif appelée golive.php.



```
<?php
//
// question.php : poser une question de consultation en ligne pour somewebsite.com
//
//VGR01082001 Creation
//
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>SomeWebsite : snip</title>
<meta name="description" content="SomeWebsite : avocat snip">
<meta name="keywords" content="consultations juridiques en ligne : snip">
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta name="Generator" content="Wordpad">
<script language="JavaScript" type="text/javascript">
<!--- hide script from old browsers

function FValidateControl(control) {
  if (control.value=="") {
    alert(control.name+": ce champ est indispensable et doit être rempli.")
    control.focus()
    return false
  }
  return true
}

function FSubmitValidation(form) {
  if (!FValidateControl(form.nom)) return false
  snip
  return true
}

// end hiding from old browsers -->
</script>

</head>
<body>
<form method="get" action="golive.php" onsubmit="return FSubmitValidation(this);">
<input type="hidden" name="CHAMP000" value="somecode">
<input type="hidden" name="CHAMP001" value="somecode2">
<input type="hidden" name="CHAMP002" value="somecode3">
<input type="hidden" name="CHAMP003" value="I">
<input type="hidden" name="CHAMP004" value="somewebsite.com">
<input type="hidden" name="CHAMP005" value="http://somewebsite.com/pageretour.php">
<input type="hidden" name="CHAMP006" value="Jean Gouffre, sarl">
<!--non (RSTS) input type=hidden name="CHAMP007" value="http://somewebsite.com/index.php"-->
<input type="hidden" name="CHAMP008" value="jean@somewebsite.com">
<input type="hidden" name="CHAMP200" value="999999">
<input type="hidden" name="CHAMP201" value="92">
<input type="hidden" name="CHAMP202" value="EUR">
<input type="hidden" name="CHAMP900" value="04">

<input name="nom" maxlength="255" size="43" type="text" value="">
<input name="prenom" maxlength="255" size="43" type="text" value="">
<input name="adresse_ligne1" maxlength="255" size="43" type="text" value="">
<input name="ville" maxlength="255" size="43" type="text" value="">
<input name="code_postal" maxlength="255" size="16" type="text" value="">
<input name="pays" maxlength="255" size="16" type="text" value="">
<input name="e_mail" maxlength="255" size="43" type="text" value="">
<input name="teleph" maxlength="255" size="43" type="text" value="">
<input value="Paiement Sécurisé et Envoi" type="submit">
<input value="Annuler" type="reset">
</form>
</body>
</html>
```

C'est du code assez ancien, mais qui fonctionne depuis 10 ans. Nettoyé de la mise en page, il ne s'agit que de champs de saisie des données utilisateur, de champs cachés contenant les paramètres de configuration de la boutique en ligne, de l'URL de retour *etc*. La validation en Javascript est minimale dans cet exemple.

Pour des raisons pratiques, j'utilise une page appelée golive.php qui se charge d'envoyer la requête vers la plate-forme de paiement.

Dans une logique de sécurisation, ce second script golive.php aurait dû valider les champs saisis par l'utilisateur (*sanitization*[1]), ajouter les informations sensibles ci-dessus (CHAMP000 à CHAMP900) et procéder à l'envoi sous conditions. Le formulaire ne devrait pas contenir les champs CHAMP*.

Ce découpage en deux scripts a aussi le bénéfice de permettre d'enregistrer la requête dans la base de données et/ou d'envoyer un mél, en plus de la demande de paiement en ligne, sans même mentionner - pour les anciennes plate-formes de ces années-là - les problèmes éventuels en PHP3/CGI à se passer des paramètres à soi-même...

Tel quel, le formulaire avec action="GET" permet néanmoins à golive.php de récupérer directement l'argument de l'appel à la plate-forme de paiement.

Les deux fonctions (présentation du formulaire, validation+ajout+enregistrement+requête) peuvent être mélangées en un seul script, évidemment. Pour la lisibilité, il vaut mieux les séparer :

```
<?php
//
// golive.php : redirection vers paiement en ligne pour somewebsite.com
//
//VGR24012002 Création
//
$qs=$QUERY_STRING; // on va tout bêtement appendre les données requises sous les bons noms de champs

//on envoie un mél
$locdate=date("j-n-Y G:i:s");
mail("$CHAMP008", "somewebsite.com serveur : question de $nom $prenom", "Paiement de $CHAMP201
   $CHAMP202 demandé pour cette question :\n\nauteur : $nom $prenom\ndate : $locdate\nadresse :
   $adresse_ligne1\nVille : $ville\nCPostal : $code_postal\nPays : $pays\ne-mail : $e_mail\nTel. :
   $teleph\n\ncontenu : $question\n\n\nfin",$yourextraheaders);

//envoi effectif requête paiement
Header("Location: https://ecom.cimetz.com/telepaie/cgishell.exe/epaie01.exe?".$qs);
?>
```

On fait difficilement plus simple... Vous noterez que ce vieux code utilise register_globals=On et ne procède à aucune protection sérieuse contre l'injection de code malicieux, mais là n'est pas le propos.

1 voir "user input sanitization" dans le Guide OWASP 2.1 disponible à http://www.edainworks.com/

Un autre script important est la page de retour, appelée après le paiement. Voici celle faite pour CyberPlus en son temps :


```
<?php
//
// pageretour.php : page appelée depuis le serveur de paiement en ligne si paiement positif
//
//VGR24012002 Création
//
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>SomeWebsite : snip</title>
<meta name="description" content="SomeWebsite : avocat snip">
<meta name="keywords" content="consultations juridiques en ligne : snip">
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta name="Generator" content="Wordpad">
<META HTTP-EQUIV='Refresh' CONTENT='10;URL =index.html'>
</head>
<body>
<?
if ($CHAMP904=="") { // incorrect
  echo "<font size=+2>Le paiement a échoué...</font>";
 mail("$CHAMP008", "somewebsite.com serveur : Paiement a échoué pour la question de $CHAMP100
   $CHAMP101", "Paiement de $CHAMP201 $CHAMP202 refusé\n\nfin",$yourextraheaders);
} else {
  echo "<font size=+2>Merci de votre paiement, votre question a bien été prise en compte...</font>";
  mail("$CHAMP008", "somewebsite.com serveur : validation question de $CHAMP100 $CHAMP101",
   "Paiement de $CHAMP201 $CHAMP202 effectué à $CHAMP903\n\nfin",$yourextraheaders);
}
echo "<br><br>patientez svp, vous allez être redirigé vers l'accueil du site...";
?>
</body>
</html>
```

Le but de ce script est de présenter à l'utilisateur/client, de retour sur le site marchand, l'état de la transaction telle qu'elle sera vue depuis le site, en fonction du code de retour envoyé par la plate-forme bancaire. Avec CyberP@yment, c'est le CHAMP904 qui contient cette information.

4. La migration vers Cyber Plus Paiement

Tout d'abord, la bonne nouvelle : le formulaire d'appel ne change pas, même si vous pouvez ajouter un test pour déterminer la bonne image (logo Cyber Paiement) à afficher.

A noter qu'il n'existe pas de logo affichable pour Cyber Plus Paiement compatible (en dimensions) avec le logo pour Cyber P@yment.

J'ai donc utilisé l'unique logo fourni, en contrôlant sa hauteur seule :

Ensuite, il suffit de modifier le second script **golive.php** pour qu'il s'adapte au système appelé :

```
<?php
//
// golive.php : redirection vers paiement en ligne pour somewebsite.com
//
//VGR24012002
//VGR23052010 MOD paiement cyberplus avec "signature"
//
//TODO : Nil
//
//
$cyberplus=TRUE;
$testmodecpp=FALSE;

$qs=$QUERY_STRING; // on va tout bêtement appendre les données requises sous les bons noms de champs

// snip sanitization et extraction _GET[] lorsque register_globals=Off

// envoi mél
$locdate=date("d/m/Y H:i:s");
$services=urldecode($services);
$montant=$_GET['CHAMP201']; //VGR23052010 ADDed pour cpp
$numfact=$_GET['CHAMP200']; // idem

// on est prêts, envoi mél interne
// snip, inchangé

//envoi effectif requête paiement
if ($cyberplus) { //VGR23052010 REM il faut maintenant émettre un POST...
  // REM trois solutions : un formulaire en HTML + DHTML submit() ; un formulaire en DHTML ; cUrl
  //     essayons la première solution d'abord
  // Algorithme :
  // 1 - champs obligatoires entrant dans le calcul de la "signature"
  // 2 - calcul signature
  // 3 - autres champs obligatoires
  // 4 - champs optionnels
  // 5 - envoi

  // 1 - champs obligatoires entrant dans le calcul de la "signature"
  // REM je mémorise aussi le "code" car il sera retourné en cas d'erreur sur un champ
  // init variables user
  $nomprenom="$nom $prenom"; //VGR REM CyberPlus n'a en effet qu'un seul champ pour nom et prénom
  // constantes de site
  $siteid='somesite_id'; $cppver='V1';
  $desturl='https://systempay.cyberpluspaiement.com/vads-payment/';
```

```
  $urlerror='http://somewebsite.com/erreurcpp.php';
  $mode=($testmodecpp)?'TEST':'PRODUCTION';
  if ($testmodecpp) $certif='somewebsite_certif_test'; // (test)
   else $certif='somewebsite_certif_prod'; // production
  // init variables site
  $dummy=''; $devise='978'; $payconfig='SINGLE'; // sert à effectuer des paiements échelonnés si
   MULTI+paramètres
  // init variables transaction
  $tid=date('His'); $tdate=date('Ymd').$tid;
  $montant=100*$montant; // cyber plus veut des cents...
  // let's go
  $formulaire=<<<EOS
<form id="theform" method="POST" action="$desturl">
EOS;
  $oblig=array( /* VGR REM les champs pour signature doivent être dans cet ordre... */
   'version'=>array(1,'cppver','01'), /* format : nom(clef)->signature,variable source,code */
   'site_id'=>array(1,'siteid','02'),
   'ctx_mode'=>array(1,'mode','11'),
   'trans_id'=>array(1,'tid','03'),
   'trans_date'=>array(1,'tdate','04'),
   'validation_mode'=>array(1,'dummy','05'),
   'capture_delay'=>array(1,'dummy','06'),
   'payment_config'=>array(1,'payconfig','07'),
   'payment_cards'=>array(1,'dummy','08'),
   'amount'=>array(1,'montant','09'),
   'currency'=>array(1,'devise','10'),
   'signature'=>array(0,'signature','00'), /* VGR REM le code 00 a été ajouté par moi */
  );
  $optio=array(
   'cust_email'=>'e_mail',
   'cust_name'=>'nomprenom',
   'order_id'=>'numfact',
   'order_info'=>'services',
   'url_error'=>'urlerror'
  );
  // 2 - calcul signature
  $signat='';
  foreach($oblig as $key=>$val) {
    if ($val[0]>0) { // champ requis pour signature
      $signat.=${$val[1]}.'+';
      $formulaire.=<<<EOR
<input type="hidden" name="$key" value="${$val[1]}">
EOR;
    } // else NOP
  }
  $signat.=$certif;
  $signature=sha1($signat);
  // 3 - autres champs obligatoires (dont signature, implicitement)
  foreach($oblig as $key=>$val) {
    if ($val[0]<1) { // champ oblig mais requis pour signature
      $formulaire.=<<<EOR
<input type="hidden" name="$key" value="${$val[1]}">
EOR;
    } // else NOP
  }
  // 4 - champs optionnels
  foreach($optio as $key=>$val) $formulaire.=<<<EOR
<input type="hidden" name="$key" value="${$val}">
EOR;
  // 5 - terminaison HTML
  $formulaire.=<<<EOS
</form>
EOS;
  echo<<<EOS
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>somewebsite : paiement</title>
</head>
<body>
$formulaire
```

```
EOS;

  // 5 - envoi
  echo<<<EOS
<script type="text/javascript">
var fm=document.getElementById('theform');
fm.submit();
</script>
EOS;

  // 5 - terminaison HTML
  echo<<<EOS

</body>
</html>
EOS;
}
else Header("Location: https://ecom.cimetz.com/telepaie/cgishell.exe/epaie01.exe?".$qs);
?>
```

Comme vous le voyez, je ne récupère des anciens champs Cyber P@yment du formulaire existant, et aux fins de transmission vers le nouveau système, que le montant (CHAMP201) et le numéro de facture (CHAMP200) ; tout le reste doit être abandonné.

Certains des nouveaux champs, même si obligatoires et même si inclus dans la "signature" (un hachage SHA1), peuvent recevoir une valeur vide, d'où la variable $dummy qui sert en plusieurs endroits. Comme le dit la documentation[2], ce sont alors les valeur configurées dans la boutique en ligne qui s'appliqueront.

Il faut aussi concaténer le nom et le prénom en un seul champ (cust_name) et multiplier le montant (amount) issu du formulaire par 100 car Cyber Plus Paiement veut une valeur entière en cents et non plus une valeur (éventuellement à virgule) en euros. Vous noterez d'ailleurs l'étrange choix du code de devise [3].

Le reste du code est assez standard. Notez l'ordre impératif des champs pour le calcul de la signature à partir de leurs valeurs.

Notez aussi que la signature devant être calculée en fonction du montant et de l'horodatage de la transaction, en plus de valeurs fixes dépendant de la boutique, il est possible

- de la calculer avant de présenter le formulaire au client (choix adopté dans la documentation [2] page 20) si l'on accepte que l'horodatage ne corresponde pas à l'appel à la plate-forme de paiement mais à l'affichage du formulaire,
- ou de construire *a posteriori* la signature comme nous le faisons avec golive.php, ce qui permet d'avoir un horodatage plus conforme à la réalité transactionnelle. **Cette technique est de toute manière obligatoire si le montant de la transaction dépend d'un choix de l'utilisateur.**

2 [2] pages 6 à 11.

3 Le code alphabétique de l'ISO 4217 (ex. 'EUR' au lieu de '978') est beaucoup plus répandu et pratique, et n'aurait pas perturbé le mécanisme de hachage (c'était d'ailleurs la solution retenue par Cyber P@yment). Voir http://www.iso.org/iso/support/currency_codes_list-1.htm

La page de retour :

Pour les deux systèmes BPLC, il s'agit de la page appelée lors de l'appui sur "retour à la boutique" ; à ne pas confondre avec la page d'erreur de requête ni avec l'appel "de serveur à serveur". (voir plus bas, section 5)

```
<?php
//
// pageretour.php : page appelée depuis le serveur de paiement en ligne via "retour à la boutique"
//
//VGR24012002 Création
//snip
//VGR23052010 MOD paiement cyberplus
//
$cyberplus=TRUE;
$testmodecpp=FALSE;
// on tire parti de la possibilité de la plate-forme systempay d'appeler différentes URIs
// selon que le mode est TEST ou PRODUCTION (cf ctx_mode) - à configurer dans la boutique -
$lesite=($testmodecpp)?'http://192.168.0.1/somewebsite':'http://somewebsite.com';

echo<<<EOS
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>somewebsite : avocat snip</title>
<meta name="description" content="somewebsite : consultations juridiques en ligne, snip">
<meta name="keywords" content="droit snip">
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta name="Generator" content="Wordpad">
<META HTTP-EQUIV='Refresh' CONTENT='5;URL =$lesite'>
</head>
<body>
EOS;
if ($cyberplus) {
  //VGR23052010 REM retour à la boutique
  if (isset($_POST['site_id'])) { // = valeur émise
    // analyse réponse
    $CHAMP100=$_POST['cust_name']; $CHAMP101='';
    $CHAMP201=$_POST['amount']/100; $CHAMP202='EUR';
    $CHAMP903=$_POST['trans_date'];
    $ok=FALSE;
    switch(@$_POST['result']) {
      case '00' : $ok=TRUE; $message.="paiement réalisé avec succès"; break;
      case '02' : $message.="le commerçant doit contacter la banque du porteur (referral)"; break;
      case '05' : $message.="paiement refusé"; break;
      case '17' : $message.="annulation client"; break;
      case '30' : $message.="erreur de format de la requête : le champ n° {$_POST['extra_result']}
   est en erreur."; break;
      case '96' : $message.="erreur technique lors du paiement"; break;
      default : $message.="erreur inconnue ({$_POST['result']}) lors du paiement"; break;
    }
    if ($ok) {
      echo "<font size=+2>Merci de votre paiement, votre question a bien été prise en
   compte...</font>";
      mail("paul@somewebsite.com", "somewebsite .com serveur : validation question de $CHAMP100
   $CHAMP101", "Paiement de $CHAMP201 $CHAMP202 effectué à $CHAMP903\n\nfin",$yourextraheaders);
    } else {
      echo "<font size=+2>Le paiement a échoué...</font>";
      mail("paul@somewebsite.com", "somewebsite.com serveur : Paiement a échoué pour la question de
   $CHAMP100 $CHAMP101", "Paiement de $CHAMP201 $CHAMP202 refusé\n\nfin",$yourextraheaders);
    }
  } else $message="appel incorrect...";
  echo "<br><br>$message<br>";

} else { // ci-dessous, l'ancien code CyberPlus, inchangé
```

```
  if ($CHAMP904=="") { // incorrect
    echo "<font size=+2>Le paiement a échoué...</font>";
    mail("$CHAMP008", "somewebsite.com serveur : Paiement a échoué pour la question de $CHAMP100
   $CHAMP101", "Paiement de $CHAMP201 $CHAMP202 refusé\n\nfin",$yourextraheaders);
  } else {
    echo "<font size=+2>Merci de votre paiement, votre question a bien été prise en
   compte...</font>";
    mail("$CHAMP008", "somewebsite.com serveur : validation question de $CHAMP100 $CHAMP101",
   "Paiement de $CHAMP201 $CHAMP202 effectué à $CHAMP903\n\nfin",$yourextraheaders);
  }
}
echo "<br><br>patientez svp, vous allez être redirigé vers l'accueil du site...";
?>
</body>
</html>
```

Afin de conserver le même code d'analyse des retours, il suffit de poser que les nouveaux champs correspondent à des anciens :

```
    $CHAMP100=$_POST['cust_name']; $CHAMP101='';
    $CHAMP201=$_POST['amount']/100; $CHAMP202='EUR';
    $CHAMP903=$_POST['trans_date'];
```

Le code d'analyse de la valeur retournée dans result est complet, même si l'erreur 30 n'est pas détaillée. Elle ne devrait d'ailleurs pas se produire, puisqu'une page de retour d'erreur a été définie (voir erreurcpp.php en section 5)

5. Les nouveaux scripts

5.1) erreurcpp.php : la page appelée en cas d'erreur dans la requête envoyée.

```
<?php
//
// erreurcpp.php : URI appelée en cas d'erreur
//
//VGR23052010 Création
//
//

$testmodecpp=FALSE;
$eoln="\r\n";
$lesite=($testmodecpp)?'http://192.168.0.1/somewebsite':'http://somewebsite.com'; // test ou normal
echo<<<EOS
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>somewebsite : échec du paiement</title>
<meta http-equiv='refresh' content="5; url='$lesite'">
</head>
<body>
EOS;
ob_start();
if (isset($_POST['site_id'])) { // = valeur envoyée
  // analyse réponse
  $message='';
  switch(@$_POST['result']) {
    case '00' : $message.="paiement réalisé avec succès";
```

```
      if (@$_POST['validation_mode']==0) $message.=$eoln."mode de validation : automatique ( = défaut
   boutique)";
       else  $message.=$eoln."mode de validation : manuel";
      if (@$_POST['capture_delay']==0) $message.=$eoln."délai de rétention avant remise en banque : 0 ( = défaut
   boutique)";
       else  $message.=$eoln."délai de rétention avant remise en banque : {$_POST['capture_delay']}";
      break;
    case '02' : $message.="le commerçant doit contacter la banque du porteur (referral)"; break;
    case '05' : $message.="paiement refusé"; break;
    case '17' : $message.="annulation client"; break;
    case '30' : $message.="erreur de format de la requête : ";
      switch($_POST['extra_result']) {
       case '00' : $message.="la signature est fausse"; break;
       case '01' : $message.="version"; break;
       case '02' : $message.="site_id"; break;
       case '03' : $message.="trans_id"; break;
       case '04' : $message.="trans_date"; break;
       case '05' : $message.="validation_mode"; break;
       case '06' : $message.="capture_delay"; break;
       case '07' : $message.="payment_config"; break;
       case '08' : $message.="payment_cards"; break;
       case '09' : $message.="amount"; break;
       case '10' : $message.="currency"; break;
       case '11' : $message.="ctx_mode"; break;
       case '12' : $message.="language"; break;
       case '13' : $message.="order_id"; break;
       case '14' : $message.="order_info"; break;
       case '15' : $message.="cust_email"; break;
       case '16' : $message.="cust_id"; break;
       case '17' : $message.="cust_title"; break;
       case '18' : $message.="cust_name"; break;
       case '19' : $message.="cust_address"; break;
       case '20' : $message.="cust_zip"; break;
       case '21' : $message.="cust_city"; break;
       case '22' : $message.="cust_country"; break;
       case '23' : $message.="cust_phone"; break;
       case '24' : $message.="url_success"; break;
       case '25' : $message.="url_refused"; break;
       case '26' : $message.="url_referral"; break;
       case '27' : $message.="url_cancel"; break;
       case '28' : $message.="url_return"; break;
       case '29' : $message.="url_error"; break;
       default : $message.="champ inconnu (code={$_POST['extra_result']})"; break;
      }
      $message.=$eoln;
      break;
    case '96' : $message.="erreur technique lors du paiement"; break;
    default : $message.="erreur inconnue ({$_POST['result']}) lors du paiement"; break;
  }
  // génération du message à envoyer à qui de droit
  echo "$message{$eoln}";
  // ajout raw
  echo 'Raw Data received :'.$eoln;
  print_r($_POST);
  echo $eoln;
  echo 'terminé.';
  $value=ob_get_contents();
  ob_end_clean();
  mail("webmaster@somewebsite.com", "somewebsite.com serveur : erreur requête", "$value\n\nfin",
    $yourextraheaders);

  echo nl2br($message);
  echo<<<EOS
<br>
Patientez, vous allez être redirigé vers le site...
</body>
</html>
EOS;
}
?>
```

Ce script construit un message contenant le détail utile de l'erreur détectée par la plate-forme systempay, et l'envoie au client pour affichage (d'où l'appel à nl2br()). Le maximum de détails sont envoyés au ouaibemestre pour analyse, toujours par mél dans cet exemple (d'où l'usage de \n comme terminateur de ligne et l'appel à nl2br() pour affichage).

Cette information aurait dû être complète mais la plate-forme systempay ne renvoie pas la valeur en erreur. L'information transmise au ouaibemestre est donc - malheureusement - purement informative. On sait quel est le champ fautif, mais pas la valeur qui avait été transmise. La documentation[4] pêche à ce sujet.

Dans un système où les requêtes seraient enregistrées dans une base de données, cela ne poserait pas de problème de retrouver la valeur fautive, mais sinon (mode HTTP classique, *stateless*) rend le retour d'erreur reçu sans intérêt.

Notez qu'il est néanmoins possible en l'état :

- de définir dans l'interface de gestion de la boutique qu'il faut, en cas d'erreur dans la requête, retourner au formulaire d'envoi plutôt qu'à la page d'erreur spécifique ci-dessus ;
- de modifier cette page de formulaire pour ré-afficher les valeurs transmises ET la valeur en erreur, si :

```
erreur de format de la requête : amount

Raw Data received :
Array
(
    [version] =>
    [site_id] =>
    [ctx_mode] =>
    [trans_id] =>
    [trans_date] =>
    [validation_mode] =>
    [capture_delay] =>
    [payment_config] =>
    [card_brand] =>
    [card_number] =>
    [amount] =>
    [currency] =>
    [auth_mode] =>
    [auth_result] =>
    [auth_number] =>
    [warranty_result] =>
    [payment_certificate] =>
    [result] => 30
    [extra_result] => 09
    [card_country] =>
    [language] =>
    [signature] => d0c9d738812819snip7fc35
)

terminé.

fin
```

exemple de mél reçu de ce script

- on utilise a bon escient le code de champ en erreur retourné dans extra_result ;
- le serveur est capable de renvoyer au client les valeurs transmises dans la requête initiale (via des variables de session ou un cookie client). Le client dispose en effet de la même session, liée à l'instance du navigateur, que lors de l'appel au formulaire.

C'est au serveur de maintenir le jeu de données transmis, car la plate-forme systempay ne le renvoie pas complètement.

Notez que la documentation[5] est fautive car elle n'indique pas que le champ signature devrait avoir le code '00' ; en effet, si la signature est fausse, le message affiché "la signature est fausse" correspond à un code d'erreur result=30 avec extra_result='00'.

4 page 13 de "descriptif de l'interface vers la page de paiement", "retour vers le site marchand" : "les champs optionnels de la requête sont renvoyés tels quels dans la réponse" : dommage que les champs obligatoires ne suivent pas le même traitement. Pour le champ "amount", il est indiqué "idem requête" ce qui est inexact. Essayez en passant "-10".

5 page 6 de "descriptif de l'interface vers la page de paiement"

5.2) retourcpp.php : appel "de serveur à serveur"

```
<?php
//
// retourcpp.php : URI appelée "de serveur à serveur", seul moyen fiable de savoir le devenir de la
   transaction bancaire en mode sécurisé
//
//VGR23052010 Création
//VGR30052010 ADDed validation du retour (signature, origine)
//

$testmodecpp=FALSE;
$eoln="\r\n";
ob_start();
if (isset($_POST['site_id'])) { // = valeur envoyée
  // analyse réponse
  $message="<!--from={$_SERVER['HTTP_REFERER']}-->"; //VGR REM debug stuff
  //VGR30052010 ADDed analyse réponse
  // vérification origine
  if (TRUE) { //VGR REM referer non positionné par Systempay... $_SERVER['HTTP_REFERER']=='https://systempay.cyberpluspaiement.com') { //
   c'est bien Systempay qui renvoie le message
    // vérification signature
    if ($testmodecpp) $certif='58somecertif351'; // test
     else $certif='923somecertif0'; // production
    $oblig=array( /* VGR REM les champs pour signature doivent être dans cet ordre... */
     'version'=>array(1,'cppver','01'),
     'site_id'=>array(1,'siteid','02'),
     'ctx_mode'=>array(1,'mode','11'),
     'trans_id'=>array(1,'tid','03'),
     'trans_date'=>array(1,'tdate','04'),
     'validation_mode'=>array(1,'dummy','05'),
     'capture_delay'=>array(1,'dummy','06'),
     'payment_config'=>array(1,'payconfig','07'),
     'payment_cards'=>array(1,'dummy','08'),
     'amount'=>array(1,'montant','09'), /* format : nom(clef)->signature,variable source,code retour */
     'currency'=>array(1,'devise','10'),
    );
    $signat='';
    foreach($oblig as $key=>$val) $signat.=@$_POST[$key].'+';
    $signat.=$certif;
    $signature=sha1($signat);
    if ($signature==$_POST['signature']) {
      // proceed to next level
      switch(@$_POST['result']) {
        case '00' : $message.="paiement réalisé avec succès";
          if (@$_POST['validation_mode']==0) $message.=$eoln."mode de validation : automatique ( = défaut
   boutique)";
           else  $message.=$eoln."mode de validation : manuel";
          if (@$_POST['capture_delay']==0) $message.=$eoln."délai de rétention avant remise en banque : 0 ( = défaut
   boutique)";
           else  $message.=$eoln."délai de rétention avant remise en banque : {$_POST['capture_delay']}";
          break;
        case '02' : $message.="le commerçant doit contacter la banque du porteur (referral)"; break;
        case '05' : $message.="paiement refusé"; break;
        case '17' : $message.="annulation client"; break;
        case '30' : $message.="erreur de format de la requête : le champ n° {$_POST['extra_result']} est en erreur."; break;
        case '96' : $message.="erreur technique lors du paiement"; break;
        default : $message.="erreur inconnue ({$_POST['result']}) lors du paiement"; break;
      }
      // autorisation
      $message.=$eoln.'Autorisation : '.$eoln;
      if (@$_POST['auth_result']<>'') {
       switch($_POST['auth_result']) {
        case '00' : $message.="transaction approuvée";
          if (@$_POST['auth_mode']=='FULL') $message.=" en totalité"; //VGR REM attention si paiements en MULTI, ce n'est que
   l'autor. du 1er versement
           else  $message.=" avec prise d'empreinte";
          switch(@$_POST['warranty_result']) {
           case 'YES' : $message.=" - le paiement est garanti"; break;
           case 'NO' : $message.=" - le paiement n'est pas garanti"; break;
```

```
           case 'UNKNOWN' : $message.=" - le paiement n'est pas garanti suite à une erreur technique"; break;
           case '' : $message.=" - garantie de paiement non applicable"; break;
           default : $message.=" - code de retour de garantie de paiement inconnu ({$_POST['warranty_result']})"; break;
          }
          $message.="{$eoln}{$eoln} IMPORTANT : certificat de paiement - {$_POST['payment_certificate']}";
          break;
        case '02' : $message.="contacter émetteur"; break;
        case '03' : $message.="accepteur invalide"; break;
        case '04' : $message.="conserver la carte"; break;
        case '05' : $message.="ne pas honorer"; break;
        case '08' : $message.="approuver après identification"; break;
        case '12' : $message.="transaction invalide"; break;
        case '14' : $message.="numéro de porteur invalide"; break;
        case '33' : $message.="date de validité dépassée"; break;
        case '34' : $message.="suspicion de fraude"; break;
        case '41' : $message.="carte perdue"; break;
        case '43' : $message.="carte volée"; break;
        default : $message.="autre cause de refus ({$_POST['auth_result']})"; break;
       }
      } else $message.="L'autorisation a échoué";

      $message.=<<<EOS
{$eoln}
{$eoln}
Informations sur la carte utilisée : {$eoln}
type de carte : {$_POST['card_brand']}{$eoln}
numéro de carte : {$_POST['card_number']}{$eoln}
n° autorisation : {$_POST['auth_number']}{$eoln}
EOS;
    } else $message.="Le message est contrefait";
  } else $message.="La requête ne vient pas du système attendu";
  //EoAdd

  $message.=<<<EOS
{$eoln}
{$eoln}
Informations sur la carte utilisée : {$eoln}
type de carte : {$_POST['card_brand']}{$eoln}
numéro de carte : {$_POST['card_number']}{$eoln}
n° autorisation : {$_POST['auth_number']}{$eoln}
EOS;
  // génération du message à envoyer à qui de droit
  echo "$message{$eoln}";
  // ajout raw
  echo 'Raw Data received :<pre>';
  print_r($_POST);
  echo '</pre><hr>';
  echo 'terminé.';
  $value=ob_get_contents();
  ob_end_clean();
  mail("paul@somewebsite.com", "somewebsite.com serveur : retour banque", "$value\n\nfin",
   $yourextraheaders);
}
?>
```

Ce dernier script envoie (encore et toujours) un mél de confirmation au propriétaire de la boutique, à partir des informations fournies depuis la plate-forme systempay.

En particulier, le n° d'autorisation (auth_number) et (très important) le "certificat" de paiement (payment_certificate) sont à conserver précieusement pour tout contact ultérieur avec le client ou la banque.

L'analyse du retour me semble complète, même si les messages de refus ont été simplifiés.

Voici un exemple du message envoyé :

```
paiement réalisé avec succès
mode de validation : automatique ( = défaut boutique)
délai de rétention avant remise en banque : 0 ( = défaut boutique)
Autorisation :
transaction approuvée en totalité - le paiement n'est pas garanti

 IMPORTANT : certificat de paiement = b0e1216afsnipc7846381d5bc2c10991e



Informations sur la carte utilisée :

type de carte : CB

numéro de carte : 497010XXXXXX0007

n° autorisation : 740001

Raw Data received :<pre>Array
(
    [validation_mode] => 0
    [capture_delay] => 0
    [card_brand] => CB
    [card_number] => 497010XXXXXX0007
    [auth_mode] => FULL
    [auth_result] => 00
    [auth_number] => 740001
    [warranty_result] => NO
    [payment_certificate] => b0e1216snipb2937f6bc7846381d5bc2c10991e
    [result] => 00
    [extra_result] =>
    [card_country] => FR
    [language] => fr
    [hash] => p$BFd$EV$lA5yMYnoSJuc$snip263UvNDiPfMqU4sA-
    [url_check_src] => PAY
    [cust_title] =>
    [user_info] =>
    [cust_city] =>
    [order_info2] =>
    [order_id] => 999999
    [order_info3] =>
    [cust_zip] =>
    [currency] => 978
    [version] => V1
    [amount] => 4500
    [cust_name] => Graux vince
    [cust_address] =>
    [trans_id] => 140509
    [cust_email] => vgr@edainworks.com
    [theme_config] =>
    [site_id] => 72snip52
    [payment_config] => SINGLE
    [cust_id] =>
    [cust_phone] =>
    [ctx_mode] => TEST
    [trans_date] => 20100523140509
    [cust_country] =>
    [order_info] => Droit au quotidien
    [contrib] =>
    [payment_src] =>
    [signature] => e010f13snip3ed122389851f5ebf9c9bbf8snip4
)
</pre><hr>terminé.

fin
```

Vous noterez que cette fois-ci les valeurs sont bien toutes retransmises de la plate-forme systempay vers le serveur ouaibe du site, ce qui autorise à améliorer les messages (exemple "paiement de X euros autorisé"...). Attention au montant qui est en cents.

Il est évidemment possible de modifier tous ces scripts pour ne plus envoyer de méls mais plutôt écrire dans une base de données.

6. AUTRES ÉLÉMENTS MAL COUVERTS PAR LA DOCUMENTATION

6.1 Le logo de la boutique

Attention aux dimensions, un logo vaguement carré et de hauteur 100 px est recommandé.

6.2 la mise en place

Ne sont intéressantes dans le document [1] que les pages 13 (étapes 3 et 4) et 15 ("boutiques medium"). Il ne vous faut que l'identifiant et le mot de passe à l'"interface de caisse" (https://systempay.cyberpluspaiement.com/vads-merchant/) que doit vous envoyer le propriétaire de la boutique.

Il vous faut ensuite récupérer deux informations essentielles : l'ID de site (menu paramétrage/boutiques/.../configuration générale) et le "certificat" de test et/ou de production (menu paramétrage/boutiques/.../certificats), comme indiqué page 15, § 3.3 .

6.3 les tests

Veuillez vous référer au document [2] page 22, § 6.1.2 pour avoir les n° de cartes de tests ; le PV de recette qu'il faut ensuite renvoyer pour passer en production (§ 6.2) se trouve, et la documentation ne le dit pas, à https://systempay.cyberpluspaiement.com/html/Doc/Proces_Verbal_Recette.doc