CyberMUT P@iement **P@iement CIC** **P@iement OBC**

SERVICE SECURISE DE

PAIEMENT INTERNET PAR CARTE BANCAIRE

(Sécurisé par le protocole SSL)

Manuel d'installation

(Document 2/2)

EURO
INFORMATION

Sp.Tech. 1.2.8 Octobre 2001

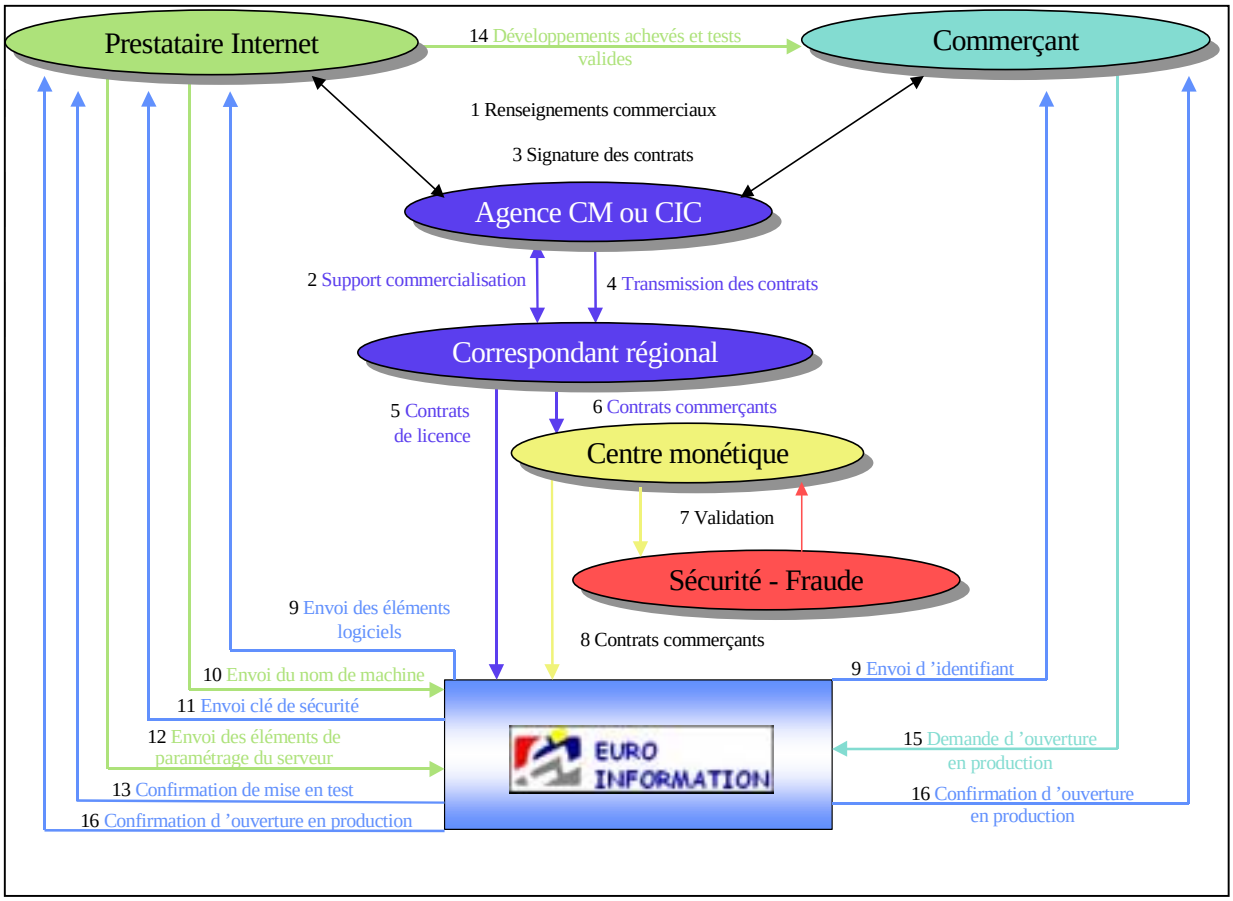
TABLE DES MATIERES

Récapitulatif des étapes de l'installation

3. Détail des éléments fournis.

Les différents éléments nécessaires à la mise en place du paiement sécurisé vous sont fournis par E-mail lors des différentes phases de l'installation.

3.1. Librairie de paiement sécurisé (version 1.2).



La solution de paiement sécurisé par carte bancaire est diffusée sous la forme d'un kit de développement.

Ce kit de développement permet de réaliser les 3 fonctionnalités nécessaires pour réaliser l'interfaçage du serveur du commerçant avec le serveur de paiement sécurisé de la banque. Il se présente sous la forme d'une **librairie de fonctions** (taille : 50 Ko environ).

Cette librairie contient essentiellement des algorithmes assurant la sécurité du système. Pour cette raison, elle vous est fournie **sous forme compilée**, en fonction du type et de la version de votre système d'exploitation ; les programmes sources ne vous sont pas fournis. Le fichier de définitions associé, contenant les prototypes des fonctions présentes dans la librairie, vous est également fourni.

La librairie de fonctions est disponible pour les systèmes d'exploitation suivants :

- **Systèmes UNIX** : librairie de fonctions C :
  fichiers ***« libcm-mac.a »*** et ***« cm-mac.h »***.

- **Systèmes UNIX avec PHP** : librairie de fonctions C + 3 CGI en C à compiler.

- **Systèmes UNIX avec JAVA**: librairie native de fonctions + classe d'appel:
  fichiers ***« libjava-cm-mac.so »*** et ***« JlibCmMac.class »***.

- **Système Windows NT** : DLL WINAPI
  fichiers ***« CMSSL.dll »***, ***« CMSSL.lib »*** et ***« CMSSL.h »***
  Eléments supplémentaires fournis à titre gracieux :
  « ax_cmssl.dll »,
  et « lisezmoi-win.txt»

- **Système MAC OS** : code fragment généré avec le produit Code Warrior PRO3
  fichiers ***« libCM.lib »*** et ***« libCM.h »***

Vous devez utiliser cette librairie de fonctions pour **créer les deux programmes d'interface** : **CGI1** (ou asp1) (génération du formulaire de la demande de paiement) et **CGI2** (ou asp2) (réception de la confirmation du paiement).

A titre d'information, **des exemples en C, PHP, Perl, Cold Fusion, ASP et Java de chacun des deux CGI (ou ASP) à créer** vous sont fournis avec la librairie. Ils sont intégralement commentés et vous présentent de façon précise la démarche à suivre. Vous pouvez utiliser ces exemples comme point de départ, en les modifiant selon les spécificités de votre environnement et de votre application.

La nature du travail à réaliser nécessite impérativement des compétences en programmation dans l'un des langages de programmation suivant :

- C ou C++ (savoir créer un petit programme CGI et le compiler en le liant à la librairie fournie).

- VisualBasic (savoir utiliser l'ActiveX fourni dans des pages ASP)

- PHP (savoir compiler 3 CGI en C sous UNIX , savoir utiliser l'ActiveX fourni sous Windows)

- Perl (savoir utiliser H2XS pour intégrer la librairie de paiement dans Perl sous UNIX, savoir installer Win32 ::API sous NT)

- Cold Fusion

- Java (savoir installer une librairie sous UNIX et savoir appeler les méthodes de la classe JlibCmMac fournie.)

3.2. Clé de sécurité commerçant.

Une clé de sécurité, certifiant les données échangées entre le serveur du commerçant et le serveur de paiement sécurisé de la banque, est attribuée par la banque à chaque commerçant.

Cette clé, associée au TPE virtuel du commerçant**, est indispensable pour utiliser le service de paiement par carte bancaire**. Il s'agit d'un fichier texte, comprenant 4 lignes non modifiables, dont le nom est « *<numéro_TPE>.key* » (exemple : clé commerçant *« 0123456.key »* pour le TPE virtuel *« 0123456 ».*

La clé commerçant est utilisable sur une seule machine : celle dont le nom DNS est renvoyé par le petit utilitaire "host", exécuté sur cette même machine.

Cette clef est utilisée lors de l'appel des fonctions de la librairie de paiement. Le paramètre TPE des fonctions doit contenir **le chemin d'accès complet** (disque local) au fichier clef (exemple : */home/cybermut/clef/0123456.key* sous UNIX ou *C:\repertoire_clef\0123456.key* sous Windows).

4. Contenu de la librairie de paiement.

La librairie de paiement qui vous est fournie **contient les 3 fonctions à utiliser pour créer vos deux CGI d'interfaçage** avec le serveur de paiement sécurisé de la banque :

- fonction **CreerFormulaireCM ()** : à utiliser dans CGI1
  (fonction **VbCreerFormulaireCM ()** : à utiliser dans ASP1)
  (fonction **JCreerFormulaireCM ()** : à utiliser en Java)

- fonction **TestMAC ()** : à utiliser dans CGI2
  (fonction **VbTestMAC ()** : à utiliser dans ASP2)
  (fonction **JTestMAC ()** : à utiliser en Java)

- fonction **CreerReponseCM ()** : à utiliser dans CGI2
  (fonction **VbCreerReponseCM ()** : à utiliser dans ASP2)
  (fonction **JCreerReponseCM ()** : à utiliser en Java)

4.1. Fonction CreerFormulaireCM ().

- **Présentation :**

  Cette fonction est à utiliser dans le CGI1 (ou asp1) **pour générer le formulaire HTML de la demande de paiement**.

- **Description :**

  Un certain nombre d'informations relatives à la commande sont nécessaires pour le traitement du paiement. Il s'agit des informations telles que le *montant de la commande*, sa *référence*, le *numéro du TPE virtuel* du commerçant, etc.

  Ces informations doivent être associées au bouton *« paiement par carte bancaire »* qui dirige l'acheteur vers le serveur de paiement sécurisé de la banque.
  Elles sont placées dans un formulaire HTML en champ caché (hidden). La seule partie visible de ce formulaire est le bouton de validation (submit) *« paiement par carte bancaire ».* Ce formulaire HTML est à insérer dans le site du commerçant pour proposer le paiement par carte bancaire.

  La fonction CreerFormulaireCM () accepte en entrée les informations nécessaires au paiement, et génère en sortie le source HTML du formulaire de demande de paiement correspondant.

  Vous noterez que dans le formulaire généré, **deux champs ont été calculés par la fonction elle-même**. Il s'agit des champs *« date »* et *« MAC ».*

  Le champ *« date »* est positionné avec la date et l'heure courante du système selon le format JJ/MM/AAAA:HH:MM:SS.

  Le champ *« MAC »* (Message Authentication Code) est calculé par un procédé de cryptographie : il contient un condensé des informations contenues dans le formulaire. Il permet de contrôler que les données n'ont pas été modifiées par l'acheteur lors de la

transmission entre le serveur du commerçant et celui de la banque (intégrité des données) et d'authentifier l'émetteur de la demande de paiement. C'est la raison pour laquelle le formulaire de la demande de paiement doit impérativement être généré par la fonction CreerFormulaireCM ().

➢ **Paramètres en entrée :**

- **url_banque** : URL du serveur de paiement sécurisé de la banque
- **version** : version du système de paiement de la banque
  version actuelle : 1.2
- **TPE** : Chemin complet du fichier clef correspondant au TPE virtuel du commerçant fourni par la banque.
  ex : /home/repertoire_clef/0123456.key
  [sur MAC OS, ne spécifier que le numéro de TPE. exemple: 0123456]
- **montant** : montant TTC de la commande formaté de la façon suivante :
  - un nombre entier
  - un point décimal (optionnel)
  - un nombre entier (optionnel sur 2 décimales)
  - une devise sur 3 caractères alphabétiques majuscule selon codification internationale ISO 4217 (cf. annexes). **La devise doit impérativement être indiquée**.
    ex : 10EUR ; 65.25USD
- **reference** : référence de la commande
  chaîne alpha-numérique (A…Z, a…z, 0…9) unique sur 12 caractères maximum permettant d'identifier la commande (par exemple, un numéro séquentiel incrémenté à chaque commande)
  ex : 000000000145
- **texte_libre** : paramètre optionnel
  zone de texte libre (40 lignes de 80 caractères chacune)
  il peut contenir une description sommaire de la commande
- **url_retour** : **URL de retour pour l'acheteur** : retour sur la page d'accueil de la boutique (attention : à ne pas confondre avec l'URL de confirmation des paiements sur laquelle sera placée le CGI2 (ou asp2))
- **url_retour_ok** : **URL de retour pour l'acheteur** : page de retour sur le site commerçant après un paiement accepté (attention : à ne pas confondre avec l'URL de confirmation des paiements sur laquelle sera placée le CGI2 (ou asp2)
- **url_retour_err** : **URL de retour pour l'acheteur** : page de retour sur le site commerçant après un paiement refusé (attention : à ne pas confondre avec l'URL de confirmation des paiements sur laquelle sera placée le CGI2 (ou asp2))

- **langue** : langue dans laquelle les pages seront affichées sur le serveur de paiement de la banque
  valeurs possibles :
  francais, anglais , allemand , espagnol ou italien

- **code_société** : code à usage interne uniquement (non affiché) qui permet à un commerçant d'utiliser le même TPE virtuel pour des sites différents (paramètres différents).
  ex : vetement

- **texte_bouton** : phrase affichée sur le bouton de validation du formulaire de demande de paiement (type « submit »)
  ex : Paiement par carte bancaire

NB : les paramètres "url_retour", "url_retour_ok" et "url_retour_err" sont optionnels (pointeurs "NULL" en langage C). Dans ce cas de figure, ce sont les paramètres stockés sur le serveur de la banque qui seront utilisés.

➢ **Paramètre en sortie :**

- **formulaire** : buffer contenant le source HTML du formulaire de demande de paiement. La taille de ce buffer doit être allouée avec une taille suffisante pour pouvoir y stocker le formulaire de sortie (en particulier, tenir compte de la taille du champ texte-libre).
  Ce formulaire est à insérér dans la page HTML du serveur web du commerçant, sur laquelle le commerçant veut proposer le paiement par carte bancaire à l'acheteur.

NB : Si un problème survient dans le calcul du champ MAC, un message d'erreur s'affiche dans le champ caché "MAC".

➢ **Prototype de la fonction CreerFormulaireCM () pour les systèmes UNIX :**

```
extern void CreerFormulaireCM (char *url_banque,
                               char *version,
                               char *TPE,
                               char *montant,
                               char *reference,
                               char *texte_libre,
                               char *url_retour,
                               char *url_retour_ok,
                               char *url_retour_err,
                               char *langue,
                               char *code_societe,
                               char *texte_bouton,
                               char *formulaire);
```

➢ **Prototype de la fonction CreerFormulaireCM () pour le système WINDOWS NT :**

```
extern void WINAPI CreerFormulaireCM (char *url_banque,
                                      char *version,
                                      char *TPE,
                                      char *montant,
                                      char *reference,
                                      char *texte_libre,
                                      char *url_retour,
                                      char *url_retour_ok,
                                      char *url_retour_err,
                                      char *langue,
                                      char *code_societe,
                                      char *texte_bouton,
                                      char *formulaire);
```

➢ **Prototype de la fonction CreerFormulaireCM () pour le système MAC OS :**

La fonction possède deux paramètres supplémentaires en entrée :

- **key_file** : chemin absolu du fichier contenant la clé commerçant
  ex : /u/keys/0123456.key
- **hostname** : nom DNS de la machine sur laquelle est installée la librairie de paiement sécurisé
  ex : www.server.com

```
extern void CreerFormulaireCM (char *url_banque,
                               char *version,
                               char *TPE,
                               char *montant,
                               char *reference,
                               char *texte_libre,
                               char *url_retour,
                               char *url_retour_ok,
                               char *url_retour_err,
                               char *langue,
                               char *code_societe,
                               char *texte_bouton,
                               char *key_file,
                               char *host_name,
                               char *formulaire);
```

Après avoir traité la demande de paiement, **le serveur de la banque informe directement le système informatique du commerçant du résultat de la demande de paiement en émettant une requête HTTP on-line**, contenant le résultat de la demande de paiement, sur l'URL de confirmation des paiements. Cette URL correspond à l'URL du CGI2 (ou asp2) (URL que vous devez nous indiquer au moment de la mise en place du système).

Le CGI2 (ou asp2) est chargé **de recevoir la requête de confirmation du paiement, d'en extraire les différentes informations**, dont le résultat du paiement**, et d'y répondre par un accusé de réception** : pour cela **il doit impérativement faire appel à la fonction *TestMAC ()*** (prise en compte des aspects de sécurisation des échanges) puis **à la fonction *CreerReponseCM ()*** (génération de l'accusé de réception à renvoyer au serveur de la banque).

Le CGI2 (ou asp2) sera appelé par le serveur de la banque avec la méthode GET ou POST et recevra un formulaire contenant les champs suivants :
(notons que si vous travaillez en VIRTUAL HOSTING, vous devez nous le préciser et dans ce cas vous devez utiliser la méthode POST).
Dans tous les cas il faut nous préciser que si vous êtes en méthode POST, pour que nous puissions correctement paramètrer nos bases .

- **TPE** : TPE virtuel du commerçant
  ex : 0123456
- **date** : date de la commande
  ex : `06/06/1999_a_17:30:42`
- **montant** : montant de la commande
  ex : 10EUR ; 65.25USD
- **référence** : référence de la commande
  ex : 000000000145
- **MAC** : de façon analogue au formulaire de demande de paiement à l'aller, le formulaire de retour contient également un champ MAC : condensé calculé à partir des données placées dans le formulaire, garantissant l'intégrité des données reçues et permettant d'authentifier son émetteur.
  ex : `d0f7e7d748420395df1ff18e50f598d616058fdb`

- **texte-libre** : zone de texte libre placée dans la demande de paiement

- **code-retour** : résultat du paiement
  valeurs possibles :
  payetest si la carte bancaire est **acceptée sur le serveur de test**
  paiement si la carte bancaire est **acceptée sur le serveur de production**
  Annulation si la carte bancaire est **refusée**

La Query String recue est de la forme suivante :

"TPE=<numero_TPE>&date=<date>&montant=<montant>&reference=<reference>&MAC=<code_MAC>&texte-libre=<texte_libre>&code-retour=<code_retour>"

Exemple :

"TPE=1234567&date=09%2f10%2f1996%5fa%5f15%3a51%3a46&montant=10%2e75&reference=0000234&MAC=e4359a2c18d86cf2e4b0e646016c202e89947b04&texte-libre=test&code-retour=paiement"

La confirmation par fax peut être mis en place, C'est une option payante (5F/fax). Cette confirmation ne remplace pas celle on-line obligatoire.

Dans le cas d'une confirmation par fax, le commerçant sera informé du montant (TTC) de la transaction ainsi que du détail de la commande (correspondant au champ texte-libre de la confirmation par requête on-line).

4.2. Fonction TestMAC ().

➢ **Présentation :**

Cette fonction est à utiliser dans le CGI2 (ou asp2) **pour s'assurer qu'il n'y a pas eu de falsification des données contenues dans le message de confirmation du paiement reçu.**

➢ **Description :**

Le message de confirmation reçu comporte un code MAC qui a été calculé par le serveur de paiement de la banque.
L'objet de la fonction TestMAC () est de recalculer le code MAC associé au message et de le comparer à celui transmis dans le message : **si les deux codes MAC sont identiques, l'information reçue est fiable** (intégrité et authentification émetteur).

La fonction TestMAC () accepte en entrée le code MAC (calculé par le serveur de paiement de la banque) transmis dans le message, ainsi que les données permettant de le recalculer. Elle indique en sortie si le message reçu est correct ou a été falsifié.

➢ **Paramètres en entrée :**

- **MAC** : code calculé par le serveur de paiement de la banque et transmis dans le message de confirmation du paiement (différent de celui de la phase aller)
  ex : `d0f7e7d748420395df1ff18e50f598d616058fdb`
- **version** : version du système de paiement utilisé
  version actuelle : 1.2
- **TPE** : Chemin complet du fichier clef correspondant au TPE virtuel du commerçant fourni par la banque.
  ex : /home/repertoire_clef/0123456.key
  [sur MAC OS, ne spécifier que le numéro de TPE. exemple: 0123456]
- **date** : date de la commande
  ex : `06/06/1996_a_17:30:42`
- **montant** : montant TTC de la commande formaté de la façon suivante :
  - un nombre entier
  - un point décimal (optionnel)
  - un nombre entier (optionnel)
  - une devise sur 3 caractères alphabétiques majuscule selon codification internationale ISO 4217 (cf. annexes)

  ex : 10EUR ; 65.25USD
- **reference** : référence de la commande
  chaîne alpha-numérique (A…Z, a…z, 0…9) unique permettant d'identifier la commande
  ex : 00000014500
  Pour des raisons pratiques, nous demandons une longueur de 12 caractères alphanumériques pour la référence
- **texte_libre** : zone de texte libre fournie lors de la demande de paiement
- **code-retour** : résultat du paiement
  valeurs possibles :
  payetest si la carte bancaire est **acceptée sur le serveur de test**
  paiement si la carte bancaire est **acceptée sur le serveur de production**
  Annulation si la carte bancaire est **refusée**

➢ **Paramètre en sortie :**

- **cdr_test** : résultat de l'authentification
  valeurs possibles : **1** si le message reçu est **authentifié**
  0 sinon

- **Prototype de la fonction TestMAC () pour les systèmes UNIX :**

```
extern int TestMAC (char *MAC,
                    char *version,
                    char *TPE,
                    char *date,
                    char *montant,
                    char *reference,
                    char *texte_libre,
                    char *code_retour) ;
```

- **Prototype de la fonction TestMAC () pour le système WINDOWS NT :**

```
extern void WINAPI TestMAC (char *MAC,
                            char *version,
                            char *TPE,
                            char *date,
                            char *montant,
                            char *reference,
                            char *texte_libre,
                            char *code_retour,
                            char *cdr_test) ;
```

- **Prototype de la fonction TestMAC () pour le système MAC OS :**

La fonction possède deux paramètres supplémentaires en entrée :

- **key_file** : chemin absolu du fichier contenant la clé commerçant (exemple : "Localdisk:keys:0123456.key")
- **hostname** : nom DNS de la machine sur laquelle est installée la librairie de paiement sécurisé

```
extern void TestMAC (int *cdr_test ,
                     char *MAC,
                     char *version,
                     char *TPE,
                     char *date,
                     char *montant,
                     char *reference,
                     char *texte_libre,
                     char *code_retour,
                     char *key_file,
                     char *host_name) ;
```

4.3. Fonction CreerReponseCM ().

➢ **Présentation :**

Cette fonction est à utiliser dans le CGI2 (ou asp2), après l'appel à la fonction TestMAC(), **pour générer l'accusé de réception de la confirmation de paiement.**

Le CGI2 (ou asp2) doit retourner un accusé de réception positif au serveur de paiement de la banque pour lui signifier qu'il a bien reçu le message de confirmation du paiement (et ce, que la demande de paiement ait été précédemment acceptée ou refusée).

➢ **Description :**

La fonction CreerReponseCM () génère la totalité du message d'accusé de réception de la confirmation du paiement, que le CGI2 (ou asp2) doit retourner au serveur de paiement de la banque.

Le message d'accusé de réception généré est un document de type MIME « text/plain » au format suivant :

Sur Windows NT :
```
Content-type : text/plain<CR><LF>
Version: 1 <LF>
OK<LF>
```

Autre :
```
Content-type : text/plain<LF><LF>
Version: 1 <LF>
OK<LF>
```

➢ **Paramètres en entrée :**

- **phrase** : corps du message d'accusé de réception

Valeurs possibles	*Signification*
"OK"	Le message de confirmation du paiement a été correctement reçu et authentifié
Autre	Le message de confirmation du paiement n'a pas été correctement authentifié

➢ **Paramètre en sortie :**

- **reponse** : buffer contenant le message d'accusé de réception (entête et corps du message). Ce buffer doit être affiché sur la sortie standard pour que le serveur de la banque reçoive la réponse.

➢ **Prototype de la fonction CreerReponseCM () pour les systèmes UNIX et MAC OS :**

extern void CreerReponseCM (char *phrase, char *reponse)

➢ **Prototype de la fonction CreerReponseCM () pour le système WINDOWS NT :**

extern void WINAPI CreerReponseCM (char *phrase, char *reponse)

5. Opérations à effectuer dans les CGI (ou asp)

5.1. Opérations à effectuer dans le CGI 1 (ou asp 1)

- Positionner la variable TPE (systèmes Unix et Windows)
  Cette variable doit indiquer le chemin complet de la clé de sécurité.

- Récupérer les variables à passer à la fonction CreerFormulaireCM
  Celles-ci peuvent être déduites d'un identifiant ou d'une référence passé en argument au CGI.

- Lancer l'appel à CreerFormulaireCM

- Afficher le formulaire sur la sortie standard

5.2. Opérations à effectuer dans le CGI 2 (ou asp 2)

- Positionner la variable TPE (systèmes Unix et Windows)
  Cette variable doit indiquer le chemin complet de la clé de sécurité.

- Récupérer les variables envoyées par le serveur de la banque.
  Selon la configuration souhaitée, ceux-ci sont passés avec la méthode GET ou POST.

- Tester l'intégrité des données avec la fonction TestMAC.

- Générer et envoyer l'accusé de réception avec la fonction CreerReponseCM.
  NB : L'accusé de réception doit contenir le message OK, que le paiement ait été accepté ("paiement" sur le serveur de production et "payetest" sur le serveur de test) ou non ("Annulation"). Par contre, il doit contenir autre chose si l'intégrité des données est altérée.

6. Annexes.

6.1. Contraintes.

- **sur la taille des champs.**

version	maxi = 10
TPE	maxi = 10
date	maxi = 30
montant	maxi = 20
reference	maxi = 12
MAC	maxi = 50
url_retour	maxi = 255
url_retour_ok	maxi = 255
url_retour_err	maxi = 255
lgue	maxi = 50
code-societe	maxi = 50
texte-libre	maxi = 3200 (uuencoded)

- **sur le contenu des champs.**

  Les champs « *version, TPE, date, montant, référence, MAC, url_retour, url_retour_ok, url_retour_err* » ne doivent contenir ni chr(10) ni chr(13).

6.2. URLs du serveur de la banque.

Ces URL sont celles que vous devez passez en paramètre "url_banque" dans la fonction CreerFormulaireCM.

- **URL de test**

Vous pourez valider vos développements dans l'environnement de test, disponible à l'adresse suivante :

Solution ***Cyber*MUT P@iement** :
https://www.creditmutuel.fr/telepaiement/test/paiement.cgi

Solution P@iement CIC :
https://ssl.paiement.cic-banques.fr/test/paiement.cgi

Solution P@iement OBC :
https://ssl.paiement.banque-obc.fr/test/paiement.cgi

L'accès à ces URL est protégé par un identifiant / mot-de-passe

- **URL de production**

Après avoir validé vos développements, vous pourrez vous adresser au serveur de production, disponible à l'adresse suivante :

Solution ***Cyber*MUT P@iement** :
https://www.creditmutuel.fr/telepaiement/paiement.cgi

Solution P@iement CIC :
https://ssl.paiement.cic-banques.fr/paiement.cgi

Solution P@iement OBC :
https://ssl.paiement.banque-obc.fr/paiement.cgi

6.3. Exemple de formulaire généré par la fonction CreerFormulaireCM ().

```
<form action="https://serveur_de_la_banque/test/paiement.cgi"
method="post" target="_top">
<input type="hidden" name="version" value="1.2">
<input type="hidden" name="TPE" value="0123456">
<input type="hidden" name="date" value="04/11/1999:11:13:42">
<input type="hidden" name="montant" value="10.75EUR">
<input type="hidden" name="reference" value="000001234">
<input type="hidden" name="MAC" value="05df8c76beff135f8aabe44eea07918d8b831dc1">
<input type="hidden" name="url_retour" value="http://www.home.fr/">
<input type="hidden" name="url_retour_ok" value="http://www.home.fr/paiement_ok.html">
<input type="hidden" name="url_retour_err" value="http://www.home.fr/paiement_err.html">
<input type="hidden" name="lgue" value="francais">
<input type="hidden" name="societe" value="vetement">
<input type="hidden" name="texte-libre" value="informations diverses">
<input type="submit" value="Paiement par carte bancaire">
</form>
```

6.4. Codification internationale des devises ISO 4217

DEVISE	CODE ISO 4217
Franc français	FRF
Franc suisse	CHF
Livre Sterling	GBP
Dollar américain	USD
Euro	EUR

6.5. Messages d'erreur

- "Le site de votre commerçant n'a pas été identifié par notre serveur. Nous ne sommes pas en mesure de traiter la demande de paiement relative à votre commande."

  Ce message signifie que les informations transmises par le CGI1 (ou asp1) ne sont pas reconnues par le serveur de la banque.
  Vérifiez que vous avez bien répondu aux questions qui sont posées dans le mail de fourniture de la clé.
  Vérifiez que les paramètres suivants, transmis par le CGI1 (ou asp1), correspondent aux informations que vous nous avez envoyées par mail :
  - Numéro de TPE
  - Code société (importance de la casse)
  - Langue

- "Les informations transmises par votre commerçant ont une signature non valide : Le niveau de sécurité exigé n'est pas atteint. Notre serveur n'est pas en mesure de traiter la demande de paiement relative à votre commande."

  Le champ MAC généré par le premier CGI n'est pas valide ou n'a pas pu être calculé. Veuillez afficher le source du formulaire généré par la fonction CreerFormulaireCM et regarder le contenu du champs caché "MAC".

  - "Erreur : Clé non trouvée"
    Vérifiez que la variable TPE est bien positionnée et pointe sur le chemin complet du fichier clé. Vérifiez que ce fichier est accessible en lecture.

  - "Erreur : Clé falsifiée"
    Vérifiez que le fichier clé n'a pas été altéré durant le transfert par mail (pas de lignes blanches, pas de caractères spéciaux).
    Si vous avez reçu plusieurs versions d'une même clé, veillez à utiliser la dernière version.

  - "Erreur : Host non autorisé"
    Vous essayez de faire fonctionner le système de paiement sécurisé sur une machine autre que celle sur laquelle il peut s'exécuter. Le nom de machine inscrit dans la clé de sécurité ne correspond pas au nom de machine renvoyé par le programme host. Une nouvelle clé de sécurité doit être générée pour cette machine.

- "Erreur : Host non défini"
  Cette erreur survient quand le système de paiement sécurisé ne parvient pas à trouver le nom DNS de la machine sur laquelle il s'exécute. Vérifiez le paramétrage DNS de la machine.

- "Erreur : Mémoire saturée"
  Votre programme alloue de la mémoire sans la libérer et sature le système.

- "Votre commande a déjà été traite."
  Ce message signifie que vous essayez de passer une commande sur une référence de commande déjà passée.

- "La date de validité de votre commande est dépassée"
  Ce message indique qu'il y a un trop grand décalage entre la date envoyée par votre CGI1 (ou asp1) et la date système du serveur de la banque.
  Recommencez le test avec un formulaire mis à jour.
  Vérifiez la date système de votre machine.

- "Votre paiement a été effectué. Mais le commerçant n'a pu être informé car une erreur de communication est survenue. Veuillez contacter directement le commerçant pour vérifier que votre commande sera bien prise en compte."
  Ce message signifie que la phase aller du paiement s'est bien déroulée mais que le CGI2 ne nous a pas renvoyé un accusé de réception valide.
  Vérifiez que le contenu du message renvoyé par le cgi2 (ou asp2) correspond aux spécifications techniques (cf. CreerReponseCM).