Mode Intégré - Ajax

< Précédent Suivant >

Contenu

Introduction

Objet du document

Ce document décrit la procédure d'intégration de la solution de paiement sécurisé en ligne Payline en mode API AJAX dans votre site commerçant.

Public visé

Ce document est destiné aux commerçants et intégrateurs qui souhaitent utiliser le mode d'intégration « API AJAX » la solution de paiement Payline.

Principe général

Présentation

L'interface AJAX présentée est une extension du mode direct. Le commerçant récupère les données carte sur la page récapitulative de commande hébergée sur ses serveurs. Ce mode permet l'exécution de la demande d'autorisation en mode synchrone ou asynchrone (via stockage temporaire du CVV), l'usage des fonctionnalités de portefeuille et permet de se libérer des obligations PCI-DSS.

L'interface AJAX

Deux modes d'intégration seront proposés :

- Le **mode AJAX** :
  - Réservé aux navigateurs les plus récents, capables d'effectuer des appels AJAX « cross-domain » : IE8+, Chrome, Firefox et Safari ;
  - L'intégration du type JSON-P n'est pas compatible avec PCI-DSS (passage du numéro de carte dans l'URL étant interdit) ;
  - Intégration asynchrone dans les sites et dans les app-mobile (encore plus transparent pour l'acheteur) ;
- Pour les navigateurs non compatibles, le **mode POST** avec redirection temporaire :
  - Le commerçant redirige brièvement l'acheteur sur Payline pour la tokenisation. Après traitement, Payline redirige de nouveau l' acheteur sur le site du commerçant ;
  - Payline n'affiche aucune page Web, la redirection est donc transparente pour l'acheteur ;
  - Ce mode d'intégration est compatible avec tous les navigateurs actuels.

Prérequis et précautions

Périmètre applicable

Le mode d'intégration avec l'API AJAX sur la solution de paiement Payline n'est pas adapté pour traiter les moyens de paiements avec redirection (Paypal, PaysafeCard, …) ou non soumis aux règles de PCI-DSS (vouchers prépayés, comptes eWallet, …).

Actuellement les moyens de paiement pris en charge par l'API AJAX sont :

- Les cartes CB / Visa / Mastercard ;
- Les cartes American Express ;
- Les cartes Maestro;
- Les cartes BCMC;
- Les cartes Aurore.

Pour tous les autres moyens de paiement, le marchand peut utiliser l'API directe.

Sécurité

La collecte des données de paiement par le site marchand implique un risque plus élevé pour le commerçant. Ce mode d'intégration nécessite donc une application rigoureuse des standards de sécurité.

En outre, deux éléments importants sont à mettre en œuvre par le marchand :

1. la page de collecte des données de paiement ne doit stocker (fichiers logs, …) à aucun moment les informations « numéro de carte » et « CVV » saisies par l'acheteur. Ces données ne doivent être transmises qu'à Payline au travers d'un des deux modes préconisés (AJAX cross-domaine ou http POST) ;
2. le numéro de commande fourni lors de l'appel à la fonction *getToken* doit être bien retrouvé à l'identique lors de l'analyse de la réponse. Ce contrôle garantit que les données n'ont pas été détournées dans le but de procéder à un autre paiement que celui prévu initialement.

Le point 1 est particulièrement important : le marchand doit vérifier régulièrement si le script d'appel à la fonction *getToken* a été modifié. Si tel est le cas, il doit s'assurer que les préconisations restent valides.

Prérequis techniques

Pour les serveurs PHP les exemples de code fonctionnent :

- Avec la bibliothèque *gzdecode.php*. qui est optionnelle jusqu'à la version 5.4.0 (disponible en standard pour les versions supérieures) ;
- Avec les modules *mcrypt* et *php_soap*, qui sont à activer.

Pour les serveurs J2E, afin de pouvoir accéder aux fonctions de chiffrement avec des clés supérieures à 128 bits, il faut installer le package *Java Cryptography Extension* (JCE) *Unlimited Strength Jurisdiction Policy* (http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html).

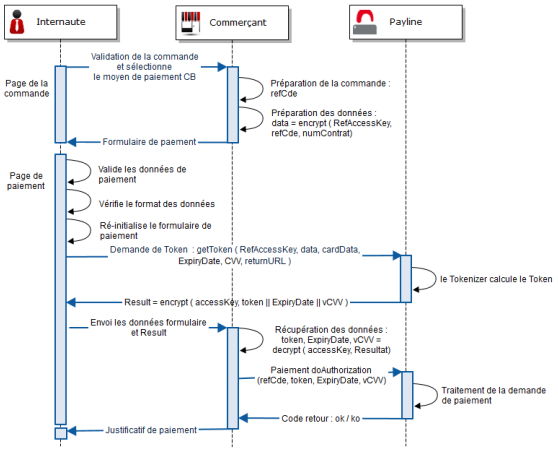
Description fonctionnelle

Cinématique d'un paiement simple

Voici les étapes principales d'un paiement avec cette nouvelle interface :

1. L'acheteur valide son panier, le navigateur envoie une requête http sur le serveur commerçant ;
2. Le serveur commerçant :
    a. génère une référence unique de commande ;
    b. chiffre avec sa clé d'accès la chaine composée de la référence commande et du numéro de contrat ;
    c. construit et renvoie le formulaire au navigateur ;
3. L'acheteur saisit et valide ses données (paiement, adresse de facturation, etc…) ;
4. Le navigateur réalise un appel à la servlet Payline « getToken » ;
5. La servlet Payline traite la requête :
    a. déchiffre les données à partir de la clé commerçant ;
    b. affecte un CVV virtuel à la transaction (valable pour une seule commande) ;
    c. tokenize le numéro de carte ;
    d. chiffre le token, la date d'expiration et le CVV virtuel (+ autres infos sur la carte) à partir de la clé du commerçant ;
    e. retourne les données au navigateur avec éventuellement une redirection http vers la « returnURL » (http 302 en mode synchrone) ;
6. Le navigateur réalise un appel au site du commerçant pour passer les données du formulaire (sans les données bancaires) avec le message chiffré par Payline ;
7. Le serveur commerçant :

Figure 1 : Cinématique d'un paiement simple (sans 3DS)



a. déchiffre le message avec sa clé d'accès pour obtenir : le token, la date d'expiration et le CVV virtuel.
b. vérifie que la référence de commande déchiffrée est bien celle attendue ;
c. appelle le WS Payline « doAuthorization » avec les données déchiffrées ;

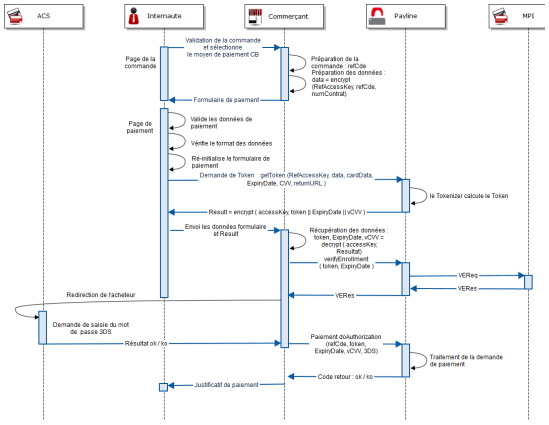
8. Le WS Payline « doAuthorization » :
    a. appelle le tokenizer pour obtenir le numéro de carte à partir du token ;
    b. récupère le CVV à partir du CVV virtuel ;
    c. appelle la banque du commerçant pour réaliser une demande d'autorisation ;
    d. réalise les contrôles anti-fraude ;
    e. retourne le résultat au commerçant.

Cinématique d'un paiement 3DSecure

Voici les étapes principales pour un paiement 3DSecure :

1. L'acheteur valide son panier, le navigateur envoie une requête http sur le serveur commerçant ;
2. Le serveur commerçant génère un formulaire comme dans la cinématique précédente ;
3. L'acheteur saisit et valide ses données (paiement, adresse de facturation, etc…) ;
4. Le navigateur :
    a. appelle la servlet « getToken » ;
    b. appelle le serveur du commerçant avec les données Payline et les autres données du formulaire.
5. Le serveur du commerçant :
    a. déchiffre les données Payline pour récupérer le token, la date d'expiration et le CVV virtuel ;
    b. stocke dans la session de l'acheteur ces données ;
    c. appelle le WS Payline « verifyEnrollment » avec le token et la date d'expiration.
6. Le WS Payline « verifyEnrollment » :
    a. déduit le numéro de carte réel à partir du token carte ;
    b. demande au MPI l'adresse de l'ACS de l'acheteur.
7. L'acheteur :
    a. est redirigé sur l'ACS de sa banque ;
    b. s'identifie ;
    c. est redirigé sur le site du commerçant (cf. TERM_URL).
8. Le serveur commerçant appelle le WS Payline « doAuthorization » avec le token, la date d'expiration, le CVV virtuel et le PARes ;
9. Le WS Payline « doAuthorization » :
    a. déduit le numéro de carte réel à partir du token carte ;
    b. récupère le CVV à partir du CVV virtuel ;
    c. appelle le MPI pour vérifier le PARes ;
    d. appelle la banque du commerçant pour réaliser une demande d'autorisation ;
    e. réalise les contrôles anti-fraude ;
    f. retourne le résultat au commerçant.

Figure 2 : Cinématique d'un paiement simple avec 3DSecure



Cinématique d'un paiement 3DSecure déclenché par le module anti-fraude

Voici les étapes principales pour un paiement 3DSecure déclenché par le module anti-fraude :

1. L'acheteur valide son panier, le navigateur envoie une requête http sur le serveur commerçant
2. Le serveur commerçant génère un formulaire comme dans la cinématique précédente
3. L'acheteur saisit et valide ses données (paiement, adresse de facturation, etc…)
4. Le navigateur :
    a. appelle la servlet Payline « getToken » ;
    b. appelle le serveur du commerçant avec les données Payline et les autres données du formulaire.
5. Le serveur du commerçant :

Figure 3 : Cinématique d'un paiement avec 3DSecure déclenché par le module LCLF

a. décrypte les données Payline pour récupérer le token, la date d'expiration et le CVV virtuel ;
b. stocke dans la session de l'acheteur ses données ;
c. appelle le WS Payline « doAuthorization » avec le token, la date d'expiration et le CVV virtuel ;

6. Le WS Payline « doAuthorization » :
   a. appelle le tokenizer pour obtenir le numéro de carte ;
   b. réalise les contrôles 3DSecure ;
   c. renvoie un code d'erreur 02715 « Authentication3DSecure is mandatory ».
7. Le serveur du commerçant appelle le WS Payline « verifyEnrollment » avec le token et la date d'expiration.

Le traitement reprend à partir de l'étape 6 de la cinématique 3DS précédente :

1. Le WS Payline « verifyEnrollment » :
   a. déduit le numéro de carte réel à partir du token carte ;
   b. demande au MPI l'adresse de l'ACS de l'acheteur.
2. L'acheteur :
   a. est redirigé sur l'ACS de sa banque ;
   b. s'identifie ;
   c. est redirigé sur le site du commerçant (cf. TERM_URL).
3. Le serveur commerçant appelle le WS Payline « doAuthorization » avec le token, la date d'expiration, le CVV virtuel et le PARes ;
4. Le WS Payline « doAuthorization » :
   a. déduit le numéro de carte réel à partir du token carte ;
   b. récupère le CVV à partir du CVV virtuel ;
   c. appelle le MPI pour vérifier le PARes ;
   d. appelle la banque du commerçant pour réaliser une demande d'autorisation ;
   e. réalise les contrôles anti-fraude ;
   f. retourne le résultat au commerçant.

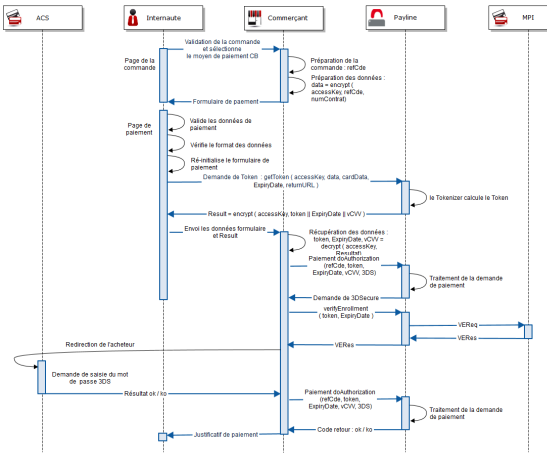


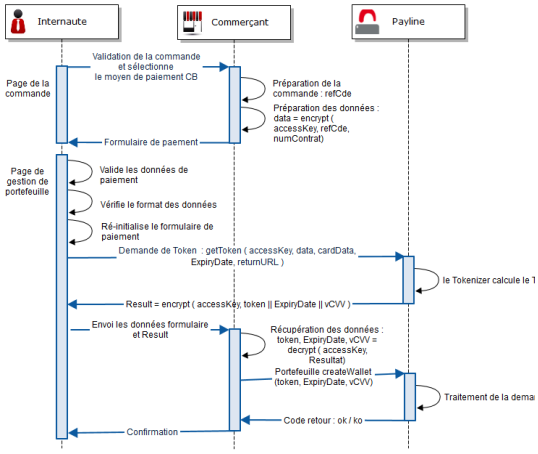
Figure 4 : Cinématique d'un enregistrement de carte dans le portefeuille Payline

Cinématique d'enregistrement d'une carte dans un portefeuille

Dans ce scénario, aucun paiement n'est réalisé.

Lorsque l'acheteur valide :

1. Le navigateur de l'acheteur envoie les données de la carte à Payline (sur le module des pages Web de paiement) ;
2. Payline retourne un token chiffré si les données sont conformes (cf. la première cinématique) ;
3. Le navigateur retourne ces données au serveur commerçant ;
4. Le serveur commerçant :
   a. déchiffre les données pour récupérer : le token, la date d'expiration et le CVV virtuel ;
   b. appelle le WS Payline « createWallet ».
5. Le WS Payline « createWallet » :
   a. appelle le tokenizer pour récupérer le numéro de carte réel et retransforme le CVV virtuel en CVV réel ;
   b. envoie une demande d'autorisation pour scoring à la banque du commerçant (ex : autorisation à 1 euro ou demande d'information selon la banque).

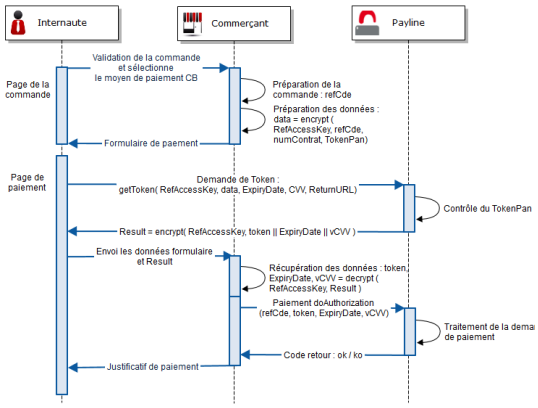


Cinématique d'un second paiement par token

Dans cette cinématique, le commerçant a conservé au préalable le token de la carte (et la date d'expiration) dans sa base de données lors du premier paiement. La page de paiement affiche les cartes associées à ce compte acheteur.
Le commerçant a la possibilité de collecter le CVV auprès de son acheteur et le fournir lors de l'appel du Servelet GetToken pour récupérer un CVV virtuel puis de réaliser un « doAuthorization ».
Le commerçant utilise le même traitement que pour un premier paiement mais avec le TokenPAN.

Lorsque l'acheteur valide la commande :

1. Le serveur commerçant
    a. recherche le token carte associé au client et appelle Payline avec le Servelet GetToken pour récupérer un vCVV ;
    b. appelle WS Payline « doAuthorization » avec le token carte, avec ou sans vCVV et un code action 120 ou 121.
2. Le WS Payline « doAuthorization »
    a. appelle le tokenizer pour récupérer le numéro de carte réel ;
    b. envoie une autorisation à la banque du commerçant.



Pour information : Le TokenPAN en sortie sera identique au TokenPAN utilisé en entrée.

Implémentation

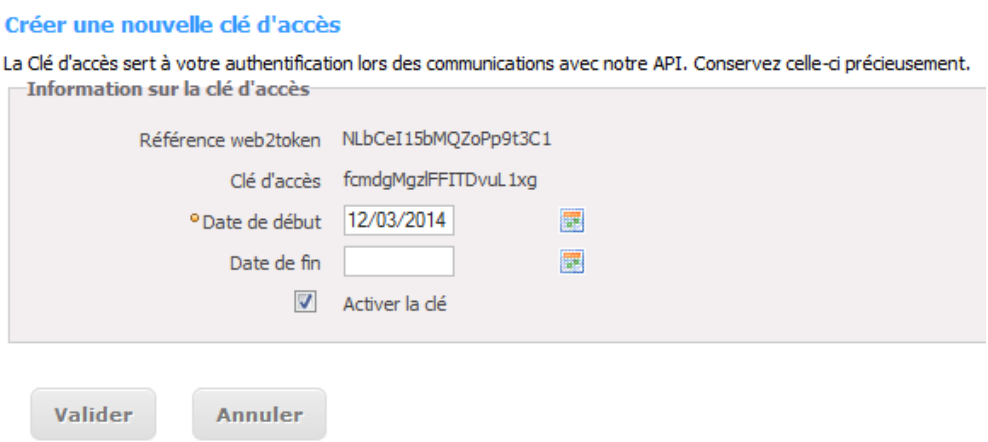
Modification de la gestion des clés commerçant

Les commerçants qui vont utiliser les paiements en mode AJAX doivent générer une clé d'accès à Payline avec le nouveau module de gestion. Ce nouveau module permet d'attribuer une **référence** de clé à chaque clé générée. Il est accessible au travers du centre d'administration dans le menu « Configuration ».

Cette référence est à utiliser dans les appels à la fonction *getToken* décrite ci-après.

La clé elle-même garde les mêmes usages qu'auparavant.

Nouvel écran de génération de clé d'accès :



Fonction de création de token

Description

Ce service est développé sous la forme d'une Servlet nommée « GetToken ». Il est à appeler au niveau de la page du commerçant pour obtenir le token de la carte de l'acheteur (requête AJAX ou requête POST+redirection).

Données en entrée

Nom du champ	Obligatoire	Format	Commentaire

Données à générer par le commerçant			
RefAccessKeyRef	O	AN(20)	Référence de la clé d'accès du commerçant Ex : IWdJoDqyGT6wOQVIf3zM
data	O	AN	Données commerçant : - chiffrées en AES256 - encodées en base64_url. Clé secrète : SHA-256 de la clé d'accès commerçant Contenu : tableau **DATA** ci-dessous
returnURL	F	AN	URL de retour sur le site du commerçant. À utiliser pour les requêtes POST+redirection
Données de la carte saisies par l'acheteur			
cardNumber	O	N(19)	Numéro de la carte
cardExpirationDate	F	MMYY	Date d'expiration Obligatoire pour CB/Visa/Mastercard/Amex
cardCvx	F	N(4)	CVV Obligatoire pour CB/Visa/Mastercard/Amex sauf pour les codes actions récurrents (120 et 121)

Le champ data contient les informations suivantes (valeurs séparées par des points virgules) :

Index	Valeur	Obligatoire	Format
1	Identifiant du commerçant	O	AN(19)
2	Référence unique de commande générée par le commerçant	O	N(50)
3	Numéro de contrat	O	AN(50)
4	Token carte. Il est obligatoirement présent si cardNumber n'est pas renseigné.	F *	AN(19)

* Obligatoire si et seulement si la donnée cardNumber est non renseignée.

Remarque : Le nombre de requêtes *getToken* est limité à 3 tentatives pour chaque référence commande. Si vous obtenez 3 erreurs lors d'une commande, il faudra re-générer une chaine chiffrée qui se base sur une nouvelle référence de commande.

Données en sortie

Le service retourne soit une liste de valeur encodées, soit un code erreur si la fonction ne s'est pas déroulée correctement.

Nom du champ	Obligatoire	Format	Commentaire
Données à générer par le commerçant			
data	F	AN	Retourné si la fonction se déroule correctement. Données : - compressées avec l'algorithme gzip - chiffrées en AES256 - encodées en base64_url Clé secrète : la même qu'en entrée Contenu : tableau ci-dessous Exemple : 497910Aztyqdeidn1234;1113;v456…
errorCode	F	N(5)	Fourni en cas d'erreur, cf. tableau suivant

Le champ data contient les informations suivantes (valeurs séparées par des points virgules) :

Index	Valeur	Obligatoire	Format
1	Token associé au numéro de de carte Ex : 497910AztyqdEGdn123	O	AN (19)
2	Date d'expiration de la carte (même donnée qu'en entrée)	F	MM YY
3	CVV virtuel, il devra être restitué dans la demande d'autorisation sans être modifié Ex. : v456	F	AN (5)

4	Référence commande identique à celle passée en entrée. Pour éviter un éventuel rejeu, **le commerçant doit contrôler que cette référence est bien celle attendue**. Si tel n'est pas le cas, il doit refuser la transaction et envoyer une annulation à Payline	O	AN (50)
5	Type de carte Ex. : VISA	O	AN
6	Indicateur « isCVD » (carte virtuelle)	O	Y ou N
7	Code pays de la carte Ex. : FR	F	AN (2)
8	Code produit de la carte Ex. : L (pour Electron)	F	AN (3)
9	Code de la banque émettrice de la carte Ex : 30003	F	AN (11)

Codes d'erreur

Code	Message court	Message long
02303	ERROR	Numéro de contrat invalide
02539	ERROR	Expiration date is mandatory for this token format.
02540	ERROR	No card found for this token.
02623	REFUSED	Nombre d'essai maximal atteint
02624	REFUSED	Carte expirée
02625	ERROR	Format du numéro de carte incorrect
02626	ERROR	Format de la date d'expiration incorrect ou date non fournie
02627	ERROR	Format du CVV incorrect ou CVV non fourni
02628	ERROR	Format de l'URL de retour incorrect
02631	REFUSED	Délai d'attente expiré
02703	ERROR	Action non autorisée
02713	ERROR	The token field is invalid. Need to be alphanumeric (13-19)
09101	ERROR	Accès non autorisé
09102	ERROR	Compte commerçant bloqué ou désactivé

API webservice

Les webservices SOAP de Payline ont évolué de façon à être compatible avec le paiement en mode Ajax.

Pour cela le champ « card.token » a été ajouté dans les paramètres d'entrée des webservices suivants :

- doAuthorization
- createWallet : voir le service pour les précisions d'utilisation avec le mode Ajax
- updateWallet
- verifyAuthentication
- verifyEnrollment
- doCredit
- doDebit

Afin de bénéficier de cette évolution, veillez à bien reprendre la dernière version du WSDL Payline.

Les URL des serveurs Payline à utiliser sont :

Homologation : https://homologation-webpayment.payline.com/webpayment/getToken

Production : https://secure.payline.com/webpayment/getToken

Exemples d'implémentation

Trois étapes sont nécessaires :

1. Préparation de la page de collecte des données carte
2. Mode Intégré - Ajax#Page de paiement
3. Traitement de la réponse

Préparation de la page de collecte des données carte

Avant de présenter la page à ses acheteurs, le commerçant doit préparer les éléments qui serviront à envoyer la demande de token à Payline :

1. Préparer un hash de sa clé d'accès Payline (méthode SHA-256) ;
2. Affecter une référence de commande unique au panier de l'acheteur ;
3. Récupérer les informations de son compte Payline : identifiant commerçant et numéro de contrat sur lequel va porter le paiement ;
4. Chiffrer ces données ;
5. Générer le formulaire de paiement présenté à l'acheteur en incluant les données chiffrées à l'étape précédente.

À noter que la chaîne de caractères chiffrée (obtenue à l'étape 5) doit être encodée en base64url (cf. https://fr.wikipedia.org/wiki/Base64#base64url).

Code serveur PHP

Préparer un hash de la clé d'accès :

```
$aes256Key = hash("SHA256", $accessKey, true);
```

La donnée accessKey représente la clé d'accès du commerçant

Pour chiffrer les données, il faut d'abord concaténer les données avant de chiffrer la chaine de caractères obtenue :

```
messageUtf8 = utf8_encode($merchantId.";".$orderRef.";".$contractNumber);
$crypted crypted = getEncrypt($messageUtf8, $aes256Key);

function getEncrypt($message, $key){
     $block = mcrypt_get_block_size('rijndael_128', 'ecb');
     $pad = $block - (strlen($message) % $block);
     $message .= str_repeat(chr($pad), $pad);
     return base64_url_encode(mcrypt_encrypt(MCRYPT_RIJNDAEL_128, $key, $message, MCRYPT_MODE_ECB));
}
```

Avec :

merchantId : identifiant Payline du commerçant
orderRef : référence unique de la commande en cours
contractNumber : numéro de contrat Payline sur lequel va porter le paiement.

La fonction **mcrypt_** (mcrypt_get_block_size, mcrypt_encrypt, mcrypt_decrypt) a été supprimé à partir de la version PHP 7.1.0.
La fonction **mcrypt_enc_get_block_size** retourne la taille de bloc de l'algorithme ouvert.

L'utilisation de cette fonction est fortement déconseillée.
Veuillez utiliser la fonction **openssl_** en vous référant à la documentation PHP : https://www.php.net/manual/fr/book.openssl.php

Code serveur J2E

Préparer un hash de la clé d'accès :

```
MessageDigest sha = MessageDigest.getInstance("SHA-256");
aes256Key = sha.digest(accessKey.getBytes("UTF-8"));
```

La donnée accessKey représente la clé d'accès du commerçant.

Chiffrer les données. Il faut d'abord concaténer les données avant de chiffrer la chaîne de caractères obtenue :

```
byte[] msgUtf8 = (merchantId+orderRef+ContractNumber).getBytes("UTF-8");

SecretKeySpec secretKeySpec = new SecretKeySpec(accessKeyBytes, "AES");
Cipher cipher = Cipher.getInstance("AES");
cipher.init(Cipher.ENCRYPT_MODE, secretKeySpec);
byte[] ciphered = cipher.doFinal(msgUtf8);

String crypted = Base64.encodeBase64URLSafeString(ciphered);
```

Avec :
merchantId : identifiant Payline du commerçant ;
orderRef : référence unique de la commande en cours ;
contractNumber : numéro de contrat Payline sur lequel va porter le paiement.

Page de paiement

La page de paiement doit implémenter un fonctionnement qui garantit que le numéro de carte saisit par l'acheteur ne sera jamais stocké (ni par le navigateur de l'acheteur, ni par le serveur web du commerçant).

L'appel à la fonction getToken() peut se faire en mode AJAX cross-domain ou via une requête http post (fournir une URL de retour dans ce cas).

En retour un traitement côté serveur web commerçant doit être appelé afin de prendre en compte la réponse et retourner la page adéquate à l' acheteur (ticket/confirmation de paiement ou page d'erreur selon les cas).

Exemple de script d'appel AJAX :

```
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script>
  // Requête AJAX pour appeler la fonction getToken de Payline
  $(document).ready( function () {
    $("#paymentForm").submit( function() {  // à la soumission du formulaire
      jQuery.support.cors = true;  // activer les requêtes ajax cross-domain
      $.ajax( {
        type: "POST",
        url: "https://homologation-webpayment.payline.com/webpayment/getToken",
        data: "data="+$("#data").val() + "&accessKeyRef=" +
$("#accessKeyRef").val() + "&cardNumber=" + $("#cardNumber").val() +
 "&cardExpirationDate=" + $("#cardExpirationDate").val() +
"&cardCvx=" + $("#cardCvx").val(),
        success: function(msg){ // si l'appel a bien fonctionné
          $.ajax({  // fonction permettant de faire de l'ajax
            type: "POST", // methode de transmission au site marchand
            url: "paymentAjax.php",  // traitement serveur (appel local)
            data: "resultPayline="
            success: function(result){  // si l'appel a bien fonctionné
     // traitement du résultat OK (afficher les parametres dans cet exemple)
              var divMsg = $(result);
              divMsg.hide();
              $("#result").append(divMsg);
              divMsg.slideDown();
            }
          });
        },
        error:function (xhr, status, error){
          console.log("Erreur lors de l'appel de Payline : " + xhr.responseText + " (" + status + " - " +
error + ")");
```

```
        }
      });
      return false; // pour rester sur la même page à la soumission du formulaire
    });
  });
</script>
```

Exemple utilisable dans un formulaire du type :

```
<form id="paymentForm" action="#" method="post">
  <input type="hidden" name="data" id="data" size='255' value="<?php echo $crypted ?>" />
  <input type="hidden" name="accessKeyRef" id="accessKeyRef" value="<?php echo $accessKeyRef ?>" />
  <label for="">Numéro de carte</label>
  <input type="text" name="cardNumber" id="cardNumber"/>
  <label for="">Date d'expiration</label>
  <input type="text" name="cardExpirationDate" id="cardExpirationDate"/>
  <label for="">Cryptogramme</label>
  <input type="text" name="cardCvx" id="cardCvx"/>
  <br/>
  <input type="submit" class="btn btn-primary" value="Payer" />
</form>
```

Avec :
crypted : données chiffrées préparées à l'étape précédente
accessKeyRef : référence de la clé d'accès commerçant

Traitement de la réponse

La réponse contient des données qu'il faut déchiffrer et décompresser, le tout étant codé en base64url.

La chaine peut ensuite être découpée pour récupérer les valeurs séparées par les points-virgules.

Une fois cette étape effectuée, il est possible de procéder à toute opération sur la carte au travers de l'API Webservice Payline.

Généralement le marchand va effectuer une demande d'autorisation à Payline (avec token carte, date d'expiration et CVV virtuel) ou bien une vérification d'enrôlement 3DSecure.

Pour plus d'information sur l'API webservice, vous pouvez consulter la documentation associée.

Serveur PHP

Déchiffrer les données reçues :

```
$zippedData = getDecrypt($data, $aes256Key);

function getDecrypt($message, $key){
     $message = base64_url_decode($message);
     $message = mcrypt_decrypt(MCRYPT_RIJNDAEL_128, $key, $message, MCRYPT_MODE_ECB);
     $block = mcrypt_get_block_size(MCRYPT_RIJNDAEL_128, MCRYPT_MODE_ECB);
     $len = strlen($message);
     $pad = ord($message[$len-1]);
     return substr($message, 0, $len-$pad);
}
```

La donnée data représente la valeur du paramètre ($_POST['data']) reçu en retour (cas où l'appel ne retourne pas une erreur).

Décompresser les données :

```
$uncompressedData = gzdecode($zippedData);
```

Découper le résultat pour récupérer le résultat de l'appel :

```
$paylineDataResponse=explode(';', $uncompressedData);

$cardToken = $paylineDataResponse[0];
$cardExpirationDate = $paylineDataResponse[1];
$cardVirtualCVV = $paylineDataResponse[2];
$orderReference = $paylineDataResponse[3];
…
```

Serveur J2E

Déchiffrer les données reçues :

```
byte[] decryptedMessage = new byte[0];
zippedData = AESEncryption.decrypt(aes256Key, data);

public static final byte[] decrypt(final String key, final String message) {
  byte[] decrypt = new byte[0];
  MessageDigest sha = MessageDigest.getInstance("SHA-256");
  keyBytes = sha.digest(key.getBytes("UTF-8"));
  SecretKeySpec secretKeySpec = new SecretKeySpec(keyBytes, "AES");
  Cipher cipher = Cipher.getInstance("AES");
  cipher.init(Cipher.DECRYPT_MODE, secretKeySpec);
  decrypt = cipher.doFinal(Base64.decodeBase64(message.getBytes("UTF-8")));
  return finalDecrypt;
 }
```

La donnée data représente la valeur du paramètre (request.getParameter("data")) reçu en retour (cas où l'appel ne retourne pas une erreur).

Décompresser les données :

```
final StringBuffer outStr = new StringBuffer();
final ByteArrayInputStream gzipedStr = new ByteArrayInputStream(zippedData);
final GZIPInputStream gis = new GZIPInputStream(gzipedStr);
final BufferedReader bf = new BufferedReader(new InputStreamReader(gis));
String line;
while ((line = bf.readLine()) != null) {
  outStr.append(line);
}
gis.close();
String uncompressedData = outStr.toString();
```

Découper le résultat pour récupérer le résultat de l'appel :

```
String[] paylineDataResponse = uncompressedData.split(";");

String cardToken = paylineDataResponse[0];
String cardExpirationDate = paylineDataResponse[1];
String cardVirtualCVV = paylineDataResponse[2];
String orderReference = paylineDataResponse[3];
…
```