PayPlug

Mise à jour d’une intégration pour être compatible avec la DSP2

Préambule

L'objectif de cette documentation est de vous partager tous les éléments nécessaires à la mise à jour de votre intégration de l'API de PayPlug afin de la rendre compatible avec avec la DSP2.

Vous trouverez un certain nombre de changements, mais comme vous allez pouvoir le constater il n'y a normalement aucune difficulté. Tous les principaux exemples sont en PHP, mais notre équipe est à votre disposition si vous avez des questions ou si vous avez besoin d'aide pour les adapter dans un autre langage.

Pour plus d'informations n'hésitez pas à consulter notre documentation en ligne.

Sommaire

Mise à jour de l'API PayPlug

Pour commencer vous devez récupérer la dernière version de la bibliothèque PayPlug disponible sur Github et replacer la version que vous utilisez actuellement.

- PHP : https://github.com/payplug/payplug-php/releases
- Python : https://github.com/payplug/payplug-python/releases

Si vous n'utilisez pas les bibliothèques PHP ou Python de PayPlug, les dernières versions de notre API nécessitent de déclarer un numéro de version dans le header HTTP qui est envoyé. La version que vous devez utiliser est la version : **2019-08-06.**

A titre d'exemple, avant pour appeler l'API vous utilisiez un appel de ce type :

```
$ curl -X GET https://api.payplug.com/v1/payments \
   -H "Authorization: Bearer sk_live_43b7e007298f57f732800a52"
```

Désormais vous devrez envoyer :

```
$ curl -X GET https://api.payplug.com/v1/payments \
   -H "Authorization: Bearer sk_live_43b7e007298f57f732800a52" \
   -H "PayPlug-Version: 2019-08-06"
```

Avec nos bibliothèque PHP et Python cette partie est déjà gérée, donc si vous les utilisez vous n'avez pas à vous en occuper.

Intégration des adaptations

Dans le cadre de la création d'un paiement certaines clés ont été remplacées et d'autres ont été ajoutées. Vous devez donc nécessairement faire des changements pour éviter des erreurs lors de vos appels à l'API.

Les informations de facturation

Dans la précédente version de l'API les informations de facturation étaient déclarées sous la clé "customer", cette dernière à été remplacée par les clés "billing" et "shipping".

Dans la partie "billing" les clés suivantes sont recommandées pour favoriser l'authentification sans friction :

first_name	Le prénom du payeur (100 caractères max).
last_name	Le nom de famille du payeur (100 caractères max).
email	L'adresse email du payeur (255 caractères max).
adress1	L'adresse postale du payeur (255 caractères max).
postcode	Le code postal du payeur (16 caractères max).
city	La ville du payeur (100 caractères max).
country	Le pays du payeur, basé sur la norme ISO 3166.
mobile_phone_number	Le numéro de mobile du destinataire ou du payeur (format intrnational standard E.164). Par exemple, un numéro de mobile français sera sous la forme +33611111111.
landline_phone_number	Le numéro fixe du destinataire ou du payeur (format intrnational standard E.164). Par exemple, un numéro fixe français sera sous la forme +33111111111.
language	La langue du destinataire ou du payeur en 2 lettres au format ISO 639-1 code. Langages supportés : fr, en, it.

Dans "shipping", les clés suivantes sont recommandées pour favoriser l'authentification sans friction :

first_name	Le prénom du payeur (100 caractères max).
last_name	Le nom de famille du payeur (100 caractères max).
email	L'adresse email du payeur (255 caractères max).
adress1	L'adresse postale du payeur (255 caractères max).

<table>
<tr><td>postcode</td><td>Le code postal du payeur (16 caractères max).</td></tr>
<tr><td>city</td><td>La ville du payeur (100 caractères max).</td></tr>
<tr><td>country</td><td>Le pays du payeur, basé sur la norme ISO 3166.</td></tr>
<tr><td>mobile_phone_number</td><td>Le numéro de mobile du destinataire ou du payeur (format intrnational standard E.164). Par exemple, un numéro de mobile français sera sous la forme +33611111111.</td></tr>
<tr><td>landline_phone_number</td><td>Le numéro fixe du destinataire ou du payeur (format intrnational standard E.164). Par exemple, un numéro fixe français sera sous la forme +33111111111.</td></tr>
<tr><td>language</td><td>La langue du destinataire ou du payeur en 2 lettres au format ISO 639-1 code. Langages supportés : fr, en, it.</td></tr>
<tr><td>delivery_type</td><td>Le mode de livraison. Pour cette clé vous pouvez choisir l’une des valeurs suivante :
<table><tr><td>BILLING</td><td>L’expédition se fait vers l’adresse de facturation du payeur.</td></tr><tr><td>VERIFIED</td><td>L’expédition se fait vers une adresse vérifiée par le marchand.</td></tr><tr><td>NEW</td><td>L’expédition se fait vers une adresse différente de celle de facturation.</td></tr><tr><td>SHIP_TO_STORE</td><td>La collecte s’effectue dans un point d livraison (l’adresse doit être renseignée dans la variable address de la clé shipping).</td></tr><tr><td>DIGITAL_GOODS</td><td>Tous les biens numériques (sans expédition).</td></tr><tr><td>TRAVEL_OR_EVENT</td><td>Ventes de tickets concernant l’événementiel ou les voyages (pas d’expédition).</td></tr><tr><td>OTHER</td><td>Autre (tous les services en ligne sans expédition).</td></tr></table></td></tr>
</table>

Les clés ci-dessus ne sont pas obligatoires mais elles sont fortement recommandées pour favoriser l’authentification sans friction (sans 3-D Secure). Plus de détails et d’informations sur les clés disponibles sont listées dans la documentation API.

Exemple complet

Plus concrètement voici un exemple permettant de créer un paiement en tenant compte de ces adaptations :

```php
<?php
$payment = \Payplug\Payment::create(array(
  'amount'           => 3300,
  'currency'         => 'EUR',
  'save_card'        => false,
  'billing'               => array(
    'title'               => 'miss',
    'first_name'          => 'Françoise',
    'last_name'           => 'Dupond',
    'mobile_phone_number' => '+33610000000',
    'email'               => 'francoise.dupond@example.net',
    'address1'            => '13 rue des roses',
    'postcode'            => '03260',
    'city'                => 'Framboisy',
    'country'             => 'FR',
    'language'            => 'fr'
  ),
  'shipping'         => array(
    'title'          => 'miss',
    'first_name'     => 'Françoise',
    'last_name'      => 'Dupond',
    'mobile_phone_number' => '+33610000000',
    'email'          => 'francoise.dupond@example.net',
    'address1'       => '13 rue des roses',
    'postcode'       => '03260',
    'city'           => 'Framboisy',
    'country'        => 'FR',
    'language'       => 'fr',
    'delivery_type' => 'BILLING'
  ),
  'hosted_payment' => array(
    'return_url'    => 'https://example.net/success?id=42',
    'cancel_url'    => 'https://example.net/cancel?id=42'
  ),
  'notification_url' => 'https://example.net/notifications?id=42',
  'metadata'       => array(
    'customer_id' => 42
  )
));
```

```
$payment_url = $payment->hosted_payment->payment_url;
$payment_id = $payment->id;
```

Utilisation des cartes sauvegardées

L'ensemble des actions à réaliser dans le cadre d'une sauvegarde de carte ne change pas, vous devrez juste bien entendu adapter votre code pour intégrer les informations de facturation comme indiqué ci-dessus.

Des exemples et toutes les informations concernant la sauvegarde d'une carte sont disponibles dans notre documentation API.

Les principaux changements interviennent au niveau de la création d'un paiement utilisant une carte enregistrée. Lors de la création de ce paiement vous devez impérativement utiliser la clé "initiator" avec l'une des valeurs suivantes :

PAYER	Le payeur est présent devant la boutique et le paiement s'effectue à son initiative. C'est le cas du paiement en un clic par exemple.
MERCHANT	Le payeur n'est pas présent lorsque le paiement est effectué. C'est le cas d'un paiement par abonnement ou de l'échéance d'un paiement fractionné.

Cette clé permet d'indiquer à la banque du payeur qui est l'initiateur du paiement. Cette information est importante car en fonction de ce qui va être déclaré vous pourriez avoir à gérer du 3-D Secure. Nous allons regarder en détail les deux scénarios possibles.

Le paiement en un clic

Dans le cadre d'un paiement en un clic pour la clé "initiator" vous devez utiliser la variable "PAYER", car le payeur est présent devant la boutique en ligne. Dans le cadre d'un paiement en un clic, la banque émettrice de la carte peut décider de mettre du 3-D Secure, il faut donc que vous puissiez gérer la redirection vers la page d'authentification.

Voici un exemple permettant de gérer ce cas d'usage :

```
<?php

// Création du paiement en utilisant un token de carte.
$payment = \Payplug\Payment::create(array(
```

```
  'amount'          => 1000,
  'currency'        => 'EUR',
  'payment_method' => 'card_e7133426b8de947b37161dfba1897dd1',
  'initiator'       => 'PAYER',
  'billing'       => array(
    'title'       => 'miss',
    'first_name' => 'Françoise',
    'last_name'  => 'Dupond',
    'email'       => 'francoise.dupond@example.net',
    'address1'    => '13 rue des roses',
    'postcode'    => '03260',
    'city'        => 'Framboisy',
    'country'     => 'FR',
    'language'    => 'fr'
  ),
  'shipping'         => array(
    'title'          => 'miss',
    'first_name'     => 'Françoise',
    'last_name'      => 'Dupond',
    'email'          => 'francoise.watson@example.net',
    'address1'       => '13 rue des roses',
    'postcode'       => '03260',
    'city'           => 'Framboisy',
    'country'        => 'FR',
    'language'       => 'fr',
    'delivery_type' => 'BILLING'
  ),
 'hosted_payment' => array(
    'return_url'    => 'https://www.example.net/success?id=42'
  ),
  'notification_url' => 'https://example.net/notifications?id=42',
  'metadata'       => array(
    'customer_id' => 42
  )
));

// Si le paiement est bien passé, le client est redirigé vers la page de
retour. Vous n'êtes pas obligé de faire une redirection, vous pouvez
également traiter le paiement réussi au niveau de cette étape.
if ($payment->is_paid) {
   header('Location: '.$payment->hosted_payment.return_url);
}
// En cas de refus lors de la création du paiement une erreur est
retournée
else if ($payment->failure !== null ) {
```

```
    // Afficher / logger l'erreur venant de failure.message
}
// En cas de 3-D Secure une URL est retournée vous devez donc rediriger
le client dessus
else if ($payment->hosted_payment.payment_url !== null) {
      header('Location: '.$payment->hosted_payment.payment_url);
}
// Enfin en cas de problème une erreur est retournée
else {
     // Afficher / logger une erreur
}
```

Gestion des échéances d'abonnement ou d'un paiement fractionné

Vous avez utilisé l'API de PayPlug pour mettre en place une gestion d'abonnement ou de paiement fractionné. Le client a déjà effectué un premier paiement et par la suite en vous basant sur un échéancier que vous avez mis en place un script permettant de débiter automatiquement la carte.

Lors du paiement le payeur n'est donc pas présent et il n'effectue pas une action pour déclencher le paiement. Les paiements récurrents ou fractionnés avec un montant fixe sont exemptés d'authentification (une fois le 1er paiement effectué) car ils sont considérés comme étant "initiés par le marchand". Si vous êtes dans ce cas, la clé "initiator" doit utiliser la variable "MERCHANT".

Voici un exemple permettant de gérer ce cas d'usage :

```
<?php

// Création du paiement en utilisant un token de carte.
$payment = \Payplug\Payment::create(array(
  'amount'         => 1000,
  'currency'       => 'EUR',
  'payment_method' => 'card_e7133426b8de947b37161dfba1897dd1',
  'initiator'      => 'MERCHANT',
  'billing'      => array(
    'title'      => 'miss',
    'first_name' => 'Françoise',
    'last_name'  => 'Dupond',
    'email'      => 'francoise.dupond@example.net',
    'address1'   => '13 rue des roses',
    'postcode'   => '03260',
```

```
    'city'       => 'Framboisy',
    'country'    => 'FR',
    'language'   => 'fr'
  ),
  'shipping'        => array(
    'title'         => 'miss',
    'first_name'    => 'Françoise',
    'last_name'     => 'Dupond',
    'email'         => 'francoise.dupond@example.net',
    'address1'      => '13 rue des roses',
    'postcode'      => '03260',
    'city'          => 'Framboisy',
    'country'       => 'FR',
    'language'      => 'fr',
    'delivery_type' => 'BILLING'
  ),
 'hosted_payment' => array(
    'return_url'   => 'https://www.example.net/success?id=42'
  ),
  'notification_url' => 'https://example.net/notifications?id=42',
  'metadata'      => array(
    'customer_id' => 42
  )
));
```

Conclusion

Comme nous avons pu le voir, les ajustements pour rendre votre intégration compatible avec la DSP2 ne sont pas si importants. Si vous rencontrez des difficultés ou si vous avez la moindre question, notre équipe est disponible pour vous accompagner.

Garder en tête, la mise en place de la DSP2 vise à permettre aux banques de s'assurer qu'une personne qui paye sur un site e-commerce est bien le propriétaire de la carte. Ainsi plus vous remontez d'informations lors de la création des paiements **plus vous augmentez vos chances de bénéficier d'un paiement sans frictions**.

Retrouvez plus d'informations sur la DSP2 :

- E-commerce : l'essentiel sur la DSP2
- Ce qu'il faut savoir sur la DSP 2