EOPayment - Development #39377

prise en charge de keyware

29 janvier 2020 13:00 - Frédéric Péters

Statut:	Fermé	**Début:**	29 janvier 2020
Priorité:	Normal	**Echéance:**	
Assigné à:	Valentin Deniaud	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	Non

Description

De : ... <...@be.keyware.com>

2. Le API

- Le API integration manual est en annexe
- Il y a un developers portal: https://dev.online.emspay.eu

Pour créer un test account:

https://boarding.emspay.eu/merchant-portal/create-test-account

Révisions associées

Révision 4eff1c29 - 21 avril 2020 17:08 - Valentin Deniaud

common: add method for checking amount value (#39377)

Révision 18764204 - 21 avril 2020 17:10 - Valentin Deniaud

add keyware payment method (#39377)

Historique

#7 - 26 février 2020 13:59 - Benjamin Dauvergne

Ok il faut utiliser return_url_template pour renvoyer le order_id.

#8 - 27 février 2020 11:50 - Valentin Deniaud

Benjamin Dauvergne a écrit :

> Ok il faut utiliser return_url_template pour renvoyer le order_id.

Je trouve que ça marche bien en utilisant simplement return_url, patch wip incoming.

#9 - 27 février 2020 12:29 - Valentin Deniaud

- Fichier 0001-add-keyware-payment-method-39377.patch ajouté

- Statut changé de Nouveau à Solution proposée

- Patch proposed changé de Non à Oui

Premier jet, avec quelques remarques :

- Je ne fais que du python3, sauf avis contraire.
- J'ai mis keyware comme nom, mais ça devrait à mon avis être emspay, non ?
  - D'ailleurs je me demande bien ce que c'est Keyware par rapport à EMS, si quelqu'un sait.
- La dénomination « transaction_id » présente partout est un joyeux pataquès : ça ne veut pas dire la même chose entre eopayment et lingo, et également à l'intérieur même de ces modules. J'espère avoir réussi à correctement rétro-ingénier le truc, mais je le note pour que ça soit regardé à la relecture.

- Il n'y a pas de gestion des exceptions autour de payment.request dans lingo, ticket ? (payfip_ws lève déjà des exceptions dans cette méthode, c'est pas moi qui ait commencé)

Sinon, ce qu'il reste (au moins) à faire :

- Implémenter l'annulation (quand à la validation, je ne sais ni à quel moment ça sert ni si ça sera utilisé, donc par défaut je ne m'y intéresse pas).
- Tester avec un paiement qui marche : les accès fournis par EMS ne permettent que d'avoir des erreurs. Je leur ai envoyé un mail.

#10 - 27 février 2020 13:47 - Benjamin Dauvergne

Valentin Deniaud a écrit :

> Premier jet, avec quelques remarques :
>
> - Je ne fais que du python3, sauf avis contraire.

Je veux bien mais dans ce cas il faut faire en sorte :

- que les tests ne s'exécutent qu'en python3 (ça c'est fait)
- que le module ne soit déclaré qu'en python3 dans *init*.py (ça non)

> - J'ai mis keyware comme nom, mais ça devrait à mon avis être emspay, non ?
>   - D'ailleurs je me demande bien ce que c'est Keyware par rapport à EMS, si quelqu'un sait.

Aucune idée, je ne comprends pas vraiment le lien entre les deux non plus, si tu finis par trouver tu peux à la l'écrire dans une docstring si ça a un intérêt.

> - La dénomination « transaction_id » présente partout est un joyeux pataquès : ça ne veut pas dire la même chose entre eopayment et lingo, et également à l'intérieur même de ces modules. J'espère avoir réussi à correctement rétro-ingénier le truc, mais je le note pour que ça soit regardé à la relecture.

Ça ne veut rien dire du tout du coté des systèmes de paiement non plus :/ Ce sont tous des mots creux comme order_id, request_id, transaction_id, etc... Dans ems il y a aussi une notion de transaction (le paiement lui même je suppose) et d'order (une commande/facture demandant un paiement), mais jamais les informations utile, genre est-ce qu'ils sont capable de bloquer la création d'un ordre pour un merchant_order_id existant ou payé et/ou de nous renvoyer le même order_id (et est-ce qu'on peut relancer le paiement sur un même order_url). Fonctionnellement la doc est super légère mais c'est un peu toujours pareil.

> - Il n'y a pas de gestion des exceptions autour de payment.request dans lingo, ticket ? (payfip_ws lève déjà des exceptions dans cette méthode, c'est pas moi qui ait commencé)

Tu peux oui.

> Sinon, ce qu'il reste (au moins) à faire :
>
> - Implémenter l'annulation (quand à la validation, je ne sais ni à quel moment ça sert ni si ça sera utilisé, donc par défaut je ne m'y intéresse pas).

A priori les cas d'utilisation de la validation/annulation venaient initialement d'IMIO donc il y a des chances que ça devienne nécessaire mais je laisserai Fred dire.

> - Tester avec un paiement qui marche : les accès fournis par EMS ne permettent que d'avoir des erreurs. Je leur ai envoyé un mail.

Ok, faudra attendre ça pour vraiment valider.

#11 - 27 février 2020 16:42 - Valentin Deniaud

Merci pour le coup d'œil.

Benjamin Dauvergne a écrit :

> que le module ne soit déclaré qu'en python3 dans *init*.py (ça non)

Yep, je fais ça.

> et est-ce qu'on peut relancer le paiement sur un même order_url

Non et c'est assez logique : un order_id est généré à chaque tentative de paiement, et est présent dans order_url. Donc une fois une tentative de

paiement effectuée, une requête sur order_url ne réaffiche pas la page de paiement mais renvoie la même réponse que l'issue de la tentative.

> est-ce qu'ils sont capable de bloquer la création d'un ordre pour un merchant_order_id existant ou payé et/ou de nous renvoyer le même order_id

Je ne comprends pas bien d'où on sort ce merchant_order_id et à quoi il sert. Ce serait un identifiant généré par nous ? Qui identifie quoi ?
En fait, c'est lié à une autre histoire d'ids que j'ai le sentiment de ne pas avoir comprise : dans le modèle Transaction de lingo, il y a order_id et bank_transaction_id. Donc en extrapolant sur la terminologie, bank_transaction_id doit être le order_id fourni par EMS. Et Transaction.order_id généré par nous. Pourquoi en avoir deux, pourquoi ne pas se baser seulement sur le order_id de EMS ?

#12 - 27 février 2020 18:31 - Benjamin Dauvergne

Valentin Deniaud a écrit :

> Merci pour le coup d'œil.
>
> Benjamin Dauvergne a écrit :
>
> > que le module ne soit déclaré qu'en python3 dans *init*.py (ça non)
>
> Yep, je fais ça.
>
> > et est-ce qu'on peut relancer le paiement sur un même order_url
>
> Non et c'est assez logique : un order_id est généré à chaque tentative de paiement, et est présent dans order_url. Donc une fois une tentative de paiement effectuée, une requête sur order_url ne réaffiche pas la page de paiement mais renvoie la même réponse que l'issue de la tentative.

Ouaip mais il y a l'autre endpoint payment links qui fait ça et c'est pas bien clair la différence entre ces deux URLs de paiement à part qu'une permet de revenir pour payer plusieurs fois jusqu'à ce que ça aboutisse et pas l'autre.

> est-ce qu'ils sont capable de bloquer la création d'un ordre pour un merchant_order_id existant ou payé et/ou de nous renvoyer le même order_id
>
> Je ne comprends pas bien d'où on sort ce merchant_order_id et à quoi il sert. Ce serait un identifiant généré par nous ? Qui identifie quoi ?

Un numéro de facture par exemple.

> En fait, c'est lié à une autre histoire d'ids que j'ai le sentiment de ne pas avoir comprise : dans le modèle Transaction de lingo, il y a order_id et bank_transaction_id. Donc en extrapolant sur la terminologie, bank_transaction_id doit être le order_id fourni par EMS. Et Transaction.order_id généré par nous. Pourquoi en avoir deux, pourquoi ne pas se baser seulement sur le order_id de EMS ?

Certains backend demandent de recevoir un numéro (la plupart d'entre eux) et eopayment le génère, d'autres le génèrent eux même (PayFiP WS, EMS), dans les deux cas c'est le premier élément renvoyé par payment.request(...) et c'est conservé dans Transaction.order_id. bank_transaction_id selon le connecteur c'est la même chose que order_id mais dans d'autres cas c'est le numéro d'autorisation du réseau CB (sips2) ou une combinaison d'autres trucs (dans payfip_ws c'est order_id + refdet) en fait ça ne sert pas à grand chose si dans les BO on peut toujours chercher par order_id.

#13 - 02 mars 2020 15:59 - Valentin Deniaud

Benjamin Dauvergne a écrit :

> Valentin Deniaud a écrit :
>
> > Je ne comprends pas bien d'où on sort ce merchant_order_id et à quoi il sert. Ce serait un identifiant généré par nous ? Qui identifie quoi ?
>
> Un numéro de facture par exemple.

De ce que je lis du code j'en déduis que tu disais ça pour info mais que ça ne concerne pas ce backend de paiement.

> Ouaip mais il y a l'autre endpoint payment links qui fait ça et c'est pas bien clair la différence entre ces deux URLs de paiement à part qu'une permet de revenir pour payer plusieurs fois jusqu'à ce que ça aboutisse et pas l'autre.

Ça me paraît assez clair, d'un côté il y a /orders/ qui pour des infos te donne un lien de paiement, de l'autre il y a /paymentlinks/ qui pour des infos te donne un lien qui permet d'être redirigé vers un lien de paiement, dans les cas où tu ne peux pas redonner toi-même les infos à chaque fois (cas

d'usage qui ne nous concerne pas, donc). En tout cas pour les deux on a un order_id qui change à chaque tentative.

> si dans les BO on peut toujours chercher par order_id

Là il y a peut-être un truc à faire, le BO de EMS ne permet pas de chercher par order_id, seulement par... merchant_order_id. Donc on pourrait générer un id à chaque requête, le passer par merchant_order_id, puis le mettre dans Transaction.order_id, et mettre le order_id de EMS dans Transaction.bank_transaction_id ? Enfin bon, tout ça pour contourner la limitation d'un BO...
Sinon je mets order_id pour les deux et roulez jeunesse.

#14 - 02 mars 2020 16:04 - Benjamin Dauvergne

Valentin Deniaud a écrit :

> Benjamin Dauvergne a écrit :
>
> > Valentin Deniaud a écrit :
> >
> > > Je ne comprends pas bien d'où on sort ce merchant_order_id et à quoi il sert. Ce serait un identifiant généré par nous ? Qui identifie quoi ?
> >
> > Un numéro de facture par exemple.
>
> De ce que je lis du code j'en déduis que tu disais ça pour info mais que ça ne concerne pas ce backend de paiement.

Ça concerne ce backend, mais ce n'est pas exploitable via eopayment/lingo.

> Là il y a peut-être un truc à faire, le BO de EMS ne permet pas de chercher par order_id, seulement par... merchant_order_id. Donc on pourrait générer un id à chaque requête, le passer par merchant_order_id, puis le mettre dans Transaction.order_id, et mettre le order_id de EMS dans Transaction.bank_transaction_id ? Enfin bon, tout ça pour contourner la limitation d'un BO...
> Sinon je mets order_id pour les deux et roulez jeunesse.

Pfiou.. oui si ça marche.

#15 - 03 mars 2020 16:41 - Valentin Deniaud

- Fichier 0001-add-keyware-payment-method-39377.patch ajouté

J'ai corrigé quelques trucs et ajouté l'annulation. Je voulais ajouter la validation pour finir, et il se trouve que l'endpoint est cassé, en gros ça se finit par une 500.
Je n'ai toujours pas eu de réponse à ma demande de numéros de CB test, mais parmi les autres méthodes de paiement qui sont proposées il y en a une qui retourne « payé » sur un malentendu, j'ai donc pu avoir ce retour.
Pour résumer l'API est nulle, la doc aussi, l'environnement de test ne vaut pas mieux, et il n'y a aucun support (probablement parce qu'il n'y a aucun utilisateur). Je pense qu'il n'y a pas grand chose d'autre à faire en l'état, donc bon pour relecture.

#16 - 03 mars 2020 21:11 - Benjamin Dauvergne

C'est un retour qu'on peut faire à IMIO qui l'entendra certainement.

PS: pour être clair le patch ne pourra pas intégrer eopayment tant qu'on ne pourra pas tester ça de bout en bout en vrai (ça ne sert un peu à rien), on peut bien sûr leur annoncer fièrement qu'on a fini mais qu'il est impossible de tester.

#17 - 03 mars 2020 21:22 - Benjamin Dauvergne

Relecture :

- mettre les liens vers la doc en commentaire dans le code, pour un truc aussi obscure ça fait toujours plaisir
- [redacted]
  ```
  def _clean_amount(amount):
  ```
  devrait aller dans PaymentCommon je pense (en classmethod).

#18 - 04 mars 2020 11:58 - Valentin Deniaud

- Fichier 0001-common-add-method-for-checking-amount-value-39377.patch ajouté

- Fichier 0002-add-keyware-payment-method-39377.patch ajouté

Remarques prises en compte.

tant qu'on ne pourra pas tester ça de bout en bout en vrai

Je note explicitement ce que permet de faire leur environnement de test. Pas de problème pour entamer la procédure de paiement, et être redirigé vers la page présentant les différentes méthodes. C'est là que commencent les problèmes, il y a le paiement par carte bleue et d'autres méthodes random. La carte bleue n'est pas testable, on a pas de numéro de test et ceux trouvés sur internet ne marchent pas. La plupart des autres méthodes permettent de finaliser le paiement, mais sur un statut « échec ». Il y en a une qui finit on ne sait pourquoi sur un statut « payé » (youpi) et même une sur un statut « en cours ».
Donc le truc est quasi testé de bout-en-bout, c'est juste que niveau robustesse et confiance on est pas beaucoup au dessus de 0.

#19 - 21 avril 2020 16:30 - Frédéric Péters

- Statut changé de Solution proposée à Solution validée

Je dirais de pousser ainsi, ça permettra peut-être d'être utilisé et à ce moment-là il y aura de vrais tests.

#20 - 21 avril 2020 17:10 - Valentin Deniaud

- Statut changé de Solution validée à Résolu (à déployer)

```
commit 18764204cd1e4a3b78417cba357c091de6ce9a41
Author: Valentin Deniaud <vdeniaud@entrouvert.com>
Date:   Wed Feb 26 15:57:52 2020 +0100

    add keyware payment method (#39377)

commit 4eff1c29f35794df733af9fb16556b9902e6e99b
Author: Valentin Deniaud <vdeniaud@entrouvert.com>
Date:   Wed Mar 4 11:12:22 2020 +0100

    common: add method for checking amount value (#39377)
```

#21 - 18 mai 2020 21:17 - Frédéric Péters

- Statut changé de Résolu (à déployer) à Solution déployée

Fichiers

EMS Online - API integration manual.docx	44,3 ko	29 janvier 2020	Frédéric Péters
0001-add-keyware-payment-method-39377.patch	14,5 ko	27 février 2020	Valentin Deniaud
0001-add-keyware-payment-method-39377.patch	15,7 ko	03 mars 2020	Valentin Deniaud
0001-common-add-method-for-checking-amount-value-39377.patch	8,32 ko	04 mars 2020	Valentin Deniaud
0002-add-keyware-payment-method-39377.patch	15,1 ko	04 mars 2020	Valentin Deniaud