EOPayment - Development #26992

Paiement différé à une date arbitraire

04 octobre 2018 14:53 - Emmanuel Cazenave

Statut:	Fermé	**Début:**	04 octobre 2018
Priorité:	Normal	**Echéance:**	
Assigné à:	Emmanuel Cazenave	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	

Description

La partie eopayment de #26914.

Pour le backend systempayv2, accepter une date explicite de remise en banque sur une transaction, qui calcule le vads_capture_delay adéquat à passer au "vrai" backend.

Si une date de remise en banque explicite est passée, et que la backend a été initialisé avec un vads_capture_delay, la valeur initiale est ignorée au profit de la valeur calculée.

Demandes liées:

Lié à Publik - Development #26914: Paiment différé à une date arbitraire	**Fermé**	**02 octobre 2018**

Révisions associées

Révision d383315b - 12 novembre 2018 15:12 - Emmanuel Cazenave

add pytz dependency (#26992)

Révision 980ab967 - 12 novembre 2018 15:18 - Emmanuel Cazenave

allow arbitrary date for deferred payment (#26992)

Révision 5728565a - 13 novembre 2018 14:15 - Emmanuel Cazenave

correct typo in debian control (#26992)

Historique

#1 - 04 octobre 2018 14:53 - Emmanuel Cazenave

- Lié à Development #26914: Paiment différé à une date arbitraire ajouté

#2 - 09 octobre 2018 15:30 - Emmanuel Cazenave

- Fichier 0001-systempayv2-allow-arbitrary-date-for-deferred-paymen.patch ajouté

- Statut changé de Nouveau à Solution proposée

- Patch proposed changé de Non à Oui

Première relecture bienvenue.

#3 - 09 octobre 2018 17:19 - Thomas Noël

Ajouter capture_date dans self.logger.debug (première ligne de request)

Je serais assez pour faire assez vite au début un premier

```
    try:
        capture_date = dt.datetime.strptime(capture_date, "%Y-%m-%d").date()
    except ToutCeQuiPeutSePlanter:
        raise ValueError('capture_date must be yyyy-mm-dd')
```

pour les jours où quelqu'un appelera eopayment en oubliant que capture_date doit être au format yyyy-mm-dd

Un truc relou : VADS_TRANS_DATE va être utcnow, donc dans les pays où y'a un décallage vers l'est, on pourra pas poser de capture_date sur le lendemain en soirée, parce qu'en TU ça sera déjà le lendemain, bref, est-ce qu'on s'en fout un peu, je sais pas. Mais quand même, le calcul de la date sera pas exact autour de minuit en France. Plutôt difficile à gérer, par expérience.

#4 - 10 octobre 2018 16:01 - Emmanuel Cazenave

- Fichier 0001-systempayv2-allow-arbitrary-date-for-deferred-paymen.patch ajouté

Voilà pour les checks sur capture date.
Je l'ai pas mis plus haut parce que plus haut il y a un mic mac sur kwargs, fields, PARAMETERS,
je comprends pas à partir de quel moment j'ai le droit de rajouter un truc dans fields sans qu'il soit trifouillé ensuite
(et j'ai pas envie de chercher à comprendre).

Pour les pays de l'est je suis pas sûr de te suivre. Ma compréhension c'est que comme VADS_TRANS_DATE est en utc et vads_capture_delay en nombre de jour, ils en déduisent un genre de VADS_CAPTURE_DATE en utc, et comme en UTC tout le monde entier est à la même heure ....

#5 - 10 octobre 2018 19:36 - Thomas Noël

Sur l'histoire du fuseau : « dt.datetime.strptime(fields[VADS_TRANS_DATE], "%Y%m%d%H%M%S").date() » va te renvoyer la date UTC (ça ne connait pas les fuseaux horaires). Et en France, entre 0 et 2h, la date UTC est celle de "la veille", donc y'aura un décalage d'une journée. Au final, le résultat de vads_capture_delay va être décalé d'un jour. J'imagine que ça peut être vite super chiant, tendance incompréhensible ("vous deviez me prélever tel jour et paf c'est pas le bon", "fallait pas faire votre demande à 1h du matin monsieur"). Et encore plus pénible dans un pays encore plus décalé par rapport au TU.

Pour approcher une valeur peut-être plus correcte, mais sans aller jusqu'à se faire ch**r avec des timezones, je proposerais de partir sur des datetime, un truc genre :

```
    capture_datetime = dt.datetime.strptime(capture_date, "%Y-%m-%d")
    capture_datetime = dt.datetime.combine(capture_datetime, dt.time(12, 0))  # on prélève à midi, zou
    vads_capture_delay = (capture_datetime - dt.datetime.strptime(fields[VADS_TRANS_DATE], "%Y%m%d%H%M%S")).da
ys
```

Ok c'est pas beau. Pour une vraie correction, faut prendre en compte la timezone, et, heu, pfffou.

(J'insiste un tout petit peu, parce qu'eopayment est un des seuls modules potentiellement utilisé ailleurs que dans Publik, voilà)

#6 - 11 octobre 2018 11:58 - Emmanuel Cazenave

Thomas Noël a écrit :

> Sur l'histoire du fuseau : « dt.datetime.strptime(fields[VADS_TRANS_DATE], "%Y%m%d%H%M%S").date() » va te renvoyer la date UTC (ça ne connait pas les fuseaux horaires). Et en France, entre 0 et 2h, la date UTC est celle de "la veille", donc y'aura un décalage d'une journée.

Un exemple quand on est à l'heure d'été en France.

temps local : '2018-07-02T01:00:00+02:00', VADS_TRANS_DATE : '2018-07-01T23:00:00+00:00', (VADS_TRANS_DATE viens de datetime.utcnow())
capture_date : '2018-07-03' qui donnera un vads_capture_delay : 2, qui donnera chez payzen VADS_CAPTURE_DATE : '2018-07-03T23:00:00+00:00'

(et je veux vraiment bien écrire du code si c'est nécessaire, c'est juste que je comprends pas le problème).

#7 - 11 octobre 2018 16:22 - Thomas Noël

Emmanuel Cazenave a écrit :

> Thomas Noël a écrit :
>
> > Sur l'histoire du fuseau : « dt.datetime.strptime(fields[VADS_TRANS_DATE], "%Y%m%d%H%M%S").date() » va te renvoyer la date UTC (ça ne connait pas les fuseaux horaires). Et en France, entre 0 et 2h, la date UTC est celle de "la veille", donc y'aura un décalage d'une journée.
>
> Un exemple quand on est à l'heure d'été en France.
>
> temps local : '2018-07-02T01:00:00+02:00', VADS_TRANS_DATE : '2018-07-01T23:00:00+00:00', (VADS_TRANS_DATE viens de datetime.utcnow())
> capture_date : '2018-07-03' qui donnera un vads_capture_delay : 2

Le soucis est ici : on a mauvais vads_capture_delay, qui devrait être à 1 puisqu'on est déjà le 2018-07-02 en heure locale.

Donc le prélèvement chez systempayv2 sera programmé au 2018-07-02 + 2 = 2018-07-04, le lendemain de ce qu'on veut. (et dans un pays plus à l'Est, la veille, ce qui est sans doute plus génant)

(sur payzen ça marche parce qu'on recalcule la date de prélèvement par rapport au TU)

> (et je veux vraiment bien écrire du code si c'est nécessaire, c'est juste que je comprends pas le problème).

Le soucis c'est que le code, je vois pas comment l'écrire, à part envoyer à eopayment l'info de TZ lorsqu'on envoie un capture_date...?

#8 - 11 octobre 2018 16:51 - Benjamin Dauvergne

Le souci c'est qu'on s'en fout un peu de la timezone locale ou d'UTC, c'est la timezone du serveur de paybox ou payzen qui importe, comme ils demandent explicitement des nombres de jours c'est qu'ils calculent selon leur propre référentiel, et donc je serai pour faire tous les calculs dans une timezone fixe prévue par le backend, ici le temps européen de l'ouest pour payzen et paybox (et on va supposer que paybox belgique leurs serveurs sont en France puisque le service vient de là).

Idem le datetime.date() reçu dans capture_date (je préfèrerai que ce soit un date() et pas une chaîne) on va le considérer dans ce temps local, quand tu fais des transactions à la bourse de New-York, c'est fermé la nuit, et c'est la nuit à New-York qui compte.

#9 - 11 octobre 2018 18:28 - Thomas Noël

Benjamin Dauvergne a écrit :

> Le souci c'est qu'on s'en fout un peu de la timezone locale ou d'UTC, c'est la timezone du serveur de paybox ou payzen qui importe, comme ils demandent explicitement des nombres de jours c'est qu'ils calculent selon leur propre référentiel, et donc je serai pour faire tous les calculs dans une timezone fixe prévue par le backend, ici le temps européen de l'ouest pour payzen et paybox (et on va supposer que paybox belgique leurs serveurs sont en France puisque le service vient de là).
>
> Idem le datetime.date() reçu dans capture_date (je préfèrerai que ce soit un date() et pas une chaîne) on va le considérer dans ce temps local, quand tu fais des transactions à la bourse de New-York, c'est fermé la nuit, et c'est la nuit à New-York qui compte.

Yep. Ca me va bien de considérer la tz Europe/Paris pour payzen/paybox. Quitte plus tard à en faire un paramètre possible dans la régie, mais plus tard seulement en cas de frexit.

#10 - 11 octobre 2018 18:46 - Emmanuel Cazenave

Bon ok je comprends qu'il y a un problème, et dans la foulée je me suis farci ça https://www.w3.org/TR/timezone/.

Je crois pas qu'on "s'en fout un peu de la timezone locale ou d'UTC".
Je crois que le problème c'est qu'on compare capture_date avec transaction_date, que sur capture_date on a pas la timezone et qu'on la compare à transaction_date qui a une timezone (utc).

Le truc propre à mon sens serait d'avoir la timezone associée à capture date, ainsi pouvoir convertir capture_date en UTC et seulement ensuite la comparer à transaction_date.
Et donc d'où on sortirait la bonne timezone aucune idée (vu que prendre celle du serveur ne résout pas le problème).

Je peux essayer faire en sorte que la timezone vienne avec capture date (et lingo enverra celle du serveur), dans l'idée que si un jour lingo ou une autre chose passe la timezone appropriée, eopayment sera déjà ok.

#11 - 11 octobre 2018 18:49 - Emmanuel Cazenave

Thomas Noël a écrit :

> Yep. Ca me va bien de considérer la tz Europe/Paris pour payzen/paybox. Quitte plus tard à en faire un paramètre possible dans la régie, mais plus tard seulement en cas de frexit.

Et donc on trouvera un terrain d'entente sur le fait que ce sera lingo qui décidera de la timezone, Europe/Paris en l'occurrence, all'right ?

#12 - 11 octobre 2018 19:53 - Emmanuel Cazenave

- Fichier Guide_d_implementation_formulaire_Paiement_V2_Abonnements_2.3.pdf ajouté

#13 - 11 octobre 2018 23:03 - Thomas Noël

Emmanuel Cazenave a écrit :

> Et donc on trouvera un terrain d'entente sur le fait que ce sera lingo qui décidera de la timezone, Europe/Paris en l'occurrence, all'right ?

En fait je serais plus radical, comme propose Benjamin : dans ce patch, considérer qu'on est sur Europe/Paris. Parce que sur paybox et payzen a priori ne gèrent que des choses dans ce fuseau.

Et dans un avenir plus lointain (autre ticket, si un jour quelqu'un en Ouzbékistan remonte le pépin), permettre une option "timezone" lors de la création du eopayment.Payment() ; ce qui permettra effectivement de configurer le fuseau au niveau de la régie dans lingo.

Voilà pour mon avis.

#14 - 12 octobre 2018 09:14 - Emmanuel Cazenave

Oubliez mes deux précédents commentaires, ce truc mes mes neurones en surchauffe, je vais faire ce que vous dites.

#15 - 12 octobre 2018 09:55 - Serghei Mihai

Thomas Noël a écrit :

> En fait je serais plus radical, comme propose Benjamin : dans ce patch, considérer qu'on est sur Europe/Paris. Parce que sur paybox et payzen a priori ne gèrent que des choses dans ce fuseau.

Dans paybox on ne passe pas de date, mais un délai de débit en nombre de jours.

#16 - 12 octobre 2018 10:45 - Thomas Noël

Serghei Mihai a écrit :

> Thomas Noël a écrit :
>
> > En fait je serais plus radical, comme propose Benjamin : dans ce patch, considérer qu'on est sur Europe/Paris. Parce que sur paybox et payzen a priori ne gèrent que des choses dans ce fuseau.
>
> Dans paybox on ne passe pas de date, mais un délai de débit en nombre de jours.

Dont le calcul est faux, cf plus haut.

#17 - 12 octobre 2018 15:51 - Emmanuel Cazenave

- Fichier 0001-add-pytz-dependency-26992.patch ajouté

- Fichier 0002-systempayv2-allow-arbitrary-date-for-deferred-paymen.patch ajouté

Voilà.

#18 - 12 octobre 2018 16:12 - Benjamin Dauvergne

Je souhaiterai que capture_date soit un datetime.date(), c'est le boulot de lingo ou autre de parser des chaînes.

#19 - 12 octobre 2018 16:16 - Benjamin Dauvergne

Benjamin Dauvergne a écrit :

> Je souhaiterai que capture_date soit un datetime.date(), c'est le boulot de lingo ou autre de parser des chaînes.

Aussi on vient pas de calculer VADS_TRANS_DATE un peu plus haut via datetime.utcnow() ?

```
                dt.datetime.strptime(fields[VADS_TRANS_DATE], "%Y%m%d%H%M%S")
```

#20 - 12 octobre 2018 16:27 - Emmanuel Cazenave

Pour tes deux remarques, tous les arguments de request sont des strings, parce que passés à travers cette moulinette :

```
    def request(self, amount, **kwargs):
        '''Request a payment to the payment backend.

          Arguments:
          amount -- the amount of money to ask
          email -- the email of the customer (optional)
          usually redundant with the hardwired settings in the bank
          configuration panel. At this url you must use the Payment.response
          method to analyze the bank returned values.

          It returns a triple of values, (transaction_id, kind, data):
            - the first gives a string value to later match the payment with
              the invoice,
            - kind gives the type of the third value, payment.URL or
              payment.HTML or payment.FORM,
            - the third is the URL or the HTML form to contact the payment
              server, which must be sent to the customer browser.
        '''
        logger.debug(u'%r' %  kwargs)
        for param in kwargs:
            # encode all input params to unicode
```

```
            kwargs[param] = force_text(kwargs[param])
        return self.backend.request(amount, **kwargs)
```

Fichtre :) Bon ben un ptit coup de isinstance(value, six.string_types), je reconnais que c'est tout pourri.

#21 - 20 octobre 2018 01:01 - Emmanuel Cazenave

- Fichier 0001-add-pytz-dependency-26992.patch ajouté

- Fichier 0002-systempayv2-allow-arbitrary-date-for-deferred-paymen.patch ajouté

Benjamin Dauvergne a écrit :

> Fichtre :) Bon ben un ptit coup de isinstance(value, six.string_types), je reconnais que c'est tout pourri.

Avec capture_date qui échappe à la transformation en unicode.
(je me base sur le nom de l'argument pour l'échappement, pas compris le @isinstance(value, six.string_types))

#22 - 22 octobre 2018 14:20 - Benjamin Dauvergne

À l'usage je me dis que ce sera plus simple d'ignorer capture_date dans ce cas :

```
            if vads_capture_delay <= 0:
                raise ValueError("capture_date needs to be superior to the transaction date")
            fields['vads_capture_delay'] = force_text(vads_capture_delay)
```

Si on a une demande avec un prélèvement aujourd'hui ou hier, et bien on prélève tout de suite. Je ne vois pas de cas où on souhaite recevoir cette erreur.

#23 - 28 octobre 2018 05:34 - Emmanuel Cazenave

Benjamin Dauvergne a écrit :

> ... Je ne vois pas de cas où on souhaite recevoir cette erreur.

Dans mon cas d'usage capture_date est calculé dans un workflow sur la base d'une autre date + x jours.Il se peut que ce calcul soit mal paramétré : donc planter tout de suite pour faire remonter ce problème plutôt que la laisser passer et s'en rendre compte au bout de n paiements foireux ....

#24 - 02 novembre 2018 11:13 - Thomas Noël

- Statut changé de Solution proposée à Solution validée

> donc planter tout de suite pour faire remonter ce problème plutôt que la laisser passer et s'en rendre compte au bout de n paiements foireux

D'accord avec ça.

Détails que tu peux voir (ou pas) avant de pousser :

- dans __init__.py le commentaire « # encode all input params to unicode » → « # encode all input params to unicode, except capture_date (datetime.date) »
- et pour bien clarifier le type attendu dans le raise de systempayv2.py : "capture_date should be a date object" → "capture_date should be a datetime.date object"

#25 - 02 novembre 2018 11:58 - Emmanuel Cazenave

- Statut changé de Solution validée à En cours

De jabber :

Fred : le calcul d'un délai à partir d'une date ça n'a rien à voir avec systempay ça aurait eu sa place dans du code commun.
....
Benj : convertissons donc capture_date en capture_delay dans Payment.request() et capture_delay et l'API native pour tous les backends, capture_date est la surcouche fournit par eopayment

#26 - 05 novembre 2018 18:46 - Emmanuel Cazenave

- Fichier 0001-add-pytz-dependency-26992.patch ajouté

- Fichier 0002-allow-arbitrary-date-for-deferred-payment-26992.patch ajouté

- Statut changé de En cours à Solution proposée

La version générique.

#27 - 05 novembre 2018 19:24 - Frédéric Péters

```
+        capture_day = kwargs['capture_day'] if 'capture_day' in kwargs else self.capture_day
```

<thomas>...</thomas>. → capture_day = kwargs.get('capture_day', self.capture_day).

```
-        if 'captureDay' in kwargs:
-            data['captureDay'] == kwargs.get('captureDay')
+        if 'capture_day' in kwargs:
+            data['captureDay'] = kwargs.get('capture_day')
```

À la marge mais ça me semble casser l'API, pour ne pas prendre de risque je garderais les deux. (avec éventuellement un deprecation warning posé sur le premier).

> \+ # capture_date can be used for deffered_payment

deferred.

```
+            delay_param = None
+            for parameter in self.backend.description['parameters']:
+                if parameter['name'] in ('capture_day', 'vads_capture_delay'):
+                    delay_param = parameter['name']
+                    break
+
+            if delay_param is None:
+                raise ValueError('capture_date is not supported by the backend.')
```

J'aurais préféré un peu de code dans le module systempayv2, qui fasse que celui-ci accepte capture_day, plutôt qu'avoir à chercher quel est le bon attribut.

#28 - 05 novembre 2018 22:50 - Benjamin Dauvergne

Emmanuel Cazenave a écrit :

> Benjamin Dauvergne a écrit :
>
> > ... Je ne vois pas de cas où on souhaite recevoir cette erreur.
>
> Dans mon cas d'usage capture_date est calculé dans un workflow sur la base d'une autre date + x jours.Il se peut que ce calcul soit mal paramétré : donc planter tout de suite pour faire remonter ce problème plutôt que la laisser passer et s'en rendre compte au bout de n paiements foireux ....

Pour moi ça va juste empêcher un fonctionnement normal, si la date calculée c'est par exemple j-5 d'un évènement mais qu'on permet une réservation jusqu'à j-1 (avec prélèvement immédiat) ça va demander un traitement particulier alors que sans rien faire tout ça marcherait normalement, et si ton erreur de calcul est dans le futur plutôt que dans le passé tu ne le sauras jamais non plus.

Vraiment je serai pour accepter n'importe quelle date et si c'est passé, l'ignorer.

#29 - 08 novembre 2018 10:44 - Emmanuel Cazenave

Benjamin Dauvergne a écrit :

> Emmanuel Cazenave a écrit :
>
> > Benjamin Dauvergne a écrit :
> >
> > > ... Je ne vois pas de cas où on souhaite recevoir cette erreur.
> >
> > Dans mon cas d'usage capture_date est calculé dans un workflow sur la base d'une autre date + x jours.Il se peut que ce calcul soit mal paramétré : donc planter tout de suite pour faire remonter ce problème plutôt que la laisser passer et s'en rendre compte au bout de n paiements foireux ....
>
> Pour moi ça va juste empêcher un fonctionnement normal, si la date calculée c'est par exemple j-5 d'un évènement mais qu'on permet une réservation jusqu'à j-1 (avec prélèvement immédiat) ça va demander un traitement particulier alors que sans rien faire tout ça marcherait

> normalement, et si ton erreur de calcul est dans le futur plutôt que dans le passé tu ne le sauras jamais non plus.

Ton exemple est différent de mon cas d'usage où capture_date doit être j+7 avec j="date où l'utilisateur occupera une salle", indépendamment du moment où la réservation est effectuée.

Toujours dans mon cas d'usage, une erreur de calcul dans le futur serait beaucoup moins grave qu'une erreur dans le passé qui pourrait donner la situation suivante : le mec a tout cassé dans la salle qu'il a occupé et on peut pas lui encaisser sa caution parce que capture_date s'est retrouvé à j-2 (et comme le paiement est aussi à "validation manuelle", et bien à j-2 ya plus de paiement plus de caution plouf dans le trou).

#30 - 08 novembre 2018 11:07 - Benjamin Dauvergne

Emmanuel Cazenave a écrit :

> Toujours dans mon cas d'usage, une erreur de calcul dans le futur serait beaucoup moins grave qu'une erreur dans le passé qui pourrait donner la situation suivante : le mec a tout cassé dans la salle qu'il a occupé et on peut pas lui encaisser sa caution parce que capture_date s'est retrouvé à j-2 (et comme le paiement est aussi à "validation manuelle", et bien à j-2 ya plus de paiement plus de caution plouf dans le trou).

capture_delay ne permet pas la validation manuelle, j'ai l'impression qu'on mélange n'importe quoi, validation manuelle il n'y a pas de délai normalement. Faut lister les capacités de chaque backend, les cas d'usage et voir si ça rentre dans notre API générique parce que sinon ça va pas le faire. Je sais que Serghei qui suit le ticket et Victor qui m'a évoqué le besoin de filer une date pour eux c'est du prélèvement automatique, pas du manuel.

#31 - 08 novembre 2018 11:10 - Benjamin Dauvergne

Et donc après discussion avec Serghei sur le cas Paybox/Arles c'est bien du déclenchement manuel mais dans ce cas il n'y a pas de délai, c'est quand on veut, sauf que si on dépasse le délai de validité de l'autorisation (c'est un délai fixe à ma connaissance) alors le prélèvement se fera sans autorisation (ça passe, mais la personne peut s'y opposer ensuite).

#32 - 08 novembre 2018 11:35 - Emmanuel Cazenave

Benjamin Dauvergne a écrit :

> capture_delay ne permet pas la validation manuelle, j'ai l'impression qu'on mélange n'importe quoi, validation manuelle il n'y a pas de délai normalement.

Je mélange rien, je vais utiliser capture_date sur une régie où l'option de validation manuelle sera activée, ce qui donne la feature 'caution' qui motivé l'ouverture de ce ticket.

#33 - 08 novembre 2018 12:46 - Emmanuel Cazenave

- Fichier 0002-allow-arbitrary-date-for-deferred-payment-26992.patch ajouté

En mettant pour l'instant de coté le débat "capture_date" dans le passé, j'ai tenu compte de toutes les remarques de Thomas et Frédéric, sauf :

Frédéric Péters a écrit :

```
-        if 'captureDay' in kwargs:
-            data['captureDay'] == kwargs.get('captureDay')
+        if 'capture_day' in kwargs:
+            data['captureDay'] = kwargs.get('capture_day')
```

> À la marge mais ça me semble casser l'API, pour ne pas prendre de risque je garderais les deux. (avec éventuellement un deprecation warning posé sur le premier).

Ce truc n'a jamais marché (noter le double '='), c'est pour ça que je me suis permis de le shooter.

#34 - 12 novembre 2018 11:19 - Benjamin Dauvergne

> @pytest.mark.freeze_time('2018-10-02 23:50:00')

Il ne faut plus faire comme cela, c'est incompatible avec la manière dont fonctionne freezegun désormais, il faut utiliser la fixture freezer et faire un freezer.move_to('2018-10-02 23:50:00').

#35 - 12 novembre 2018 15:19 - Emmanuel Cazenave

- Fichier 0002-allow-arbitrary-date-for-deferred-payment-26992.patch ajouté

Benjamin Dauvergne a écrit :

> Il ne faut plus faire comme cela, c'est incompatible avec la manière dont fonctionne freezegun désormais, il faut utiliser la fixture freezer et faire un freezer.move_to('2018-10-02 23:50:00').

Ils documentent pas ça les salopiots, voilà.

#36 - 13 novembre 2018 11:49 - Benjamin Dauvergne

- Statut changé de Solution proposée à Solution validée

Ok.

#37 - 13 novembre 2018 14:07 - Emmanuel Cazenave

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

```
commit 980ab967ba9eb3ddbb67f10b0cc574c4a12bdb47
Author: Emmanuel Cazenave <ecazenave@entrouvert.com>
Date:   Fri Oct 12 15:35:23 2018 +0200

    allow arbitrary date for deferred payment (#26992)

commit d383315b34639421d6ef29dcdb603c41f07dde28
Author: Emmanuel Cazenave <ecazenave@entrouvert.com>
Date:   Fri Oct 12 15:33:47 2018 +0200

    add pytz dependency (#26992)
```

#38 - 13 novembre 2018 14:31 - Emmanuel Cazenave

```
commit 5728565a507fb82f687125f93b6455581974bb41
Author: Emmanuel Cazenave <ecazenave@entrouvert.com>
Date:   Tue Nov 13 14:15:35 2018 +0100

    correct typo in debian control (#26992)
```

#39 - 30 novembre 2018 12:18 - Emmanuel Cazenave

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

Fichiers

0001-systempayv2-allow-arbitrary-date-for-deferred-paymen.patch	4,32 ko	09 octobre 2018	Emmanuel Cazenave
0001-systempayv2-allow-arbitrary-date-for-deferred-paymen.patch	5,42 ko	10 octobre 2018	Emmanuel Cazenave
Guide_d_implementation_formulaire_Paiement_V2_Abonnements_2.3.pdf	1,03 Mo	11 octobre 2018	Emmanuel Cazenave
0001-add-pytz-dependency-26992.patch	1,19 ko	12 octobre 2018	Emmanuel Cazenave
0002-systempayv2-allow-arbitrary-date-for-deferred-paymen.patch	6,78 ko	12 octobre 2018	Emmanuel Cazenave
0001-add-pytz-dependency-26992.patch	1,19 ko	19 octobre 2018	Emmanuel Cazenave
0002-systempayv2-allow-arbitrary-date-for-deferred-paymen.patch	8,1 ko	19 octobre 2018	Emmanuel Cazenave
0001-add-pytz-dependency-26992.patch	1,19 ko	05 novembre 2018	Emmanuel Cazenave
0002-allow-arbitrary-date-for-deferred-payment-26992.patch	12,6 ko	05 novembre 2018	Emmanuel Cazenave
0002-allow-arbitrary-date-for-deferred-payment-26992.patch	13,3 ko	08 novembre 2018	Emmanuel Cazenave
0002-allow-arbitrary-date-for-deferred-payment-26992.patch	13,3 ko	12 novembre 2018	Emmanuel Cazenave