Passerelle - Development #21465

connecteurs SMS : file pour envois asynchrones

29 janvier 2018 09:59 - Frédéric Péters

Statut:	Fermé	**Début:**	29 janvier 2018
Priorité:	Normal	**Echéance:**	
Assigné à:	Nicolas Roche	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	Non

Description

Aujourd'hui le connecteur fait l'envoi du SMS de manière synchrone et du coup timeouts, pertes de message, etc. Il faudrait gérer une file des messages, que l'envoi se passe de manière asynchrone.

Demandes liées:

Lié à Passerelle - Development #42090: Avoir des logs par job	**Nouveau**	**25 avril 2020**
Lié à Passerelle - Bug #42187: Ne pas faire dépendre les tests ProxyLogger d'...	**Fermé**	**28 avril 2020**
Lié à Passerelle - Development #42230: Uniformiser les retours des connecteur...	**Rejeté**	**28 avril 2020**

Révisions associées

Révision 115b0bb7 - 29 mai 2020 11:32 - Nicolas Roche

sms: send SMS asynchronously (#21465)

Historique

#1 - 29 janvier 2018 11:26 - Benjamin Dauvergne

Est-ce qu'on en profite pour ajouter un support asynchrone général dans passerelle ? L'idée de base ce serait au niveau de la ressource on met un flag "queue_on_error = True" et en cas d'erreur on créé un objet queue avec le contenu du POST et on se débrouille pour qu'une tâche cron rejoue ça, parce que des connecteurs "fire and forget" on n'aura pas que celui là. Plus tard on pourra faire évoluer ça pour permettre une notification de w.c.s. quand ça finit par passer.

Un modèle comme cela:

```
class Job:
   ressource = FK(ContentType)
   endpoint = string
   data = json : None
   result = json : None
   status = enum(TODO, ERROR, UNRECOVERABLE_ERROR, DONE) : TODO
   tries = int : 0
   next_try = datetime : None
   time_between_tries = int : 60
   exponential_retry_timeout_factor = decimal : 1
   max_tries = int :
```

C'est vaguement inspiré du système de job que j'ai dans zoo.

#2 - 29 janvier 2018 12:47 - Frédéric Péters

À noter aussi #3090 et #12469.

#3 - 18 novembre 2018 16:30 - Frédéric Péters

- Sujet changé de connecteurs SMS : fil pour envois asynchrones à connecteurs SMS : file pour envois asynchrones

#4 - 10 avril 2020 18:32 - Nicolas Roche

- Fichier 0001-sms-add-endpoint-to-send-SMS-asynchronously-21465.patch ajouté

- Statut changé de Nouveau à Solution proposée

- Assigné à mis à Nicolas Roche

- Patch proposed changé de Non à Oui

> Aujourd'hui le connecteur fait l'envoi du SMS de manière synchrone et du coup timeouts...

Via les Jobs, un éventuel timeout lors de l'envoi du SMS n'aura pas de répercussion sur l'appelant.

> ... , pertes de message, etc...

Les messages d'erreur sont logués dans la table des jobs.

> Il faudrait gérer une file des messages, que l'envoi se passe de manière asynchrone.

Je précise que ce patch ne gère pas l'envoie d'un lot de messages de façon assynchrone.

(ma branche est basée sur #19663 pour pouvoir tester via twilio)

#5 - 14 avril 2020 10:06 - Frédéric Péters

Il faudrait que ça soit tout le temps asynchrone, et qu'il y ait plusieurs tentatives sur l'envoi des messages en cas d'échec.

#6 - 24 avril 2020 18:28 - Nicolas Roche

- Fichier 0003-sms-add-endpoint-to-send-sms-asynchronously-21465.patch ajouté

- Fichier 0002-sms-factorise-sms-code-on-parameters-21465.patch ajouté

- Fichier 0001-sms-add-SMS-class-to-store-sms-parameters-21465.patch ajouté

J'ai ajouté une classe SMS pour stocker les paramètres du Job (en m'inspirant du connecteur mdel_ddpacs).
Un Job est créé pour chaque SMS puis d'autres le sont (et s'exécuteront 5 minutes plus tard) en cas d'erreur de transport (requests.RequestException).

#7 - 24 avril 2020 18:43 - Frédéric Péters

Il y a un job.parameters qui permet de ne pas créer de modèle.

#8 - 24 avril 2020 20:22 - Nicolas Roche

Oui, je l'ai utilisé dans le premier patch.
Ensuite on peut relancer les Jobs avec SkipJob, mais j'ai l'impression qu'alors on ne peut pas garder trace des erreurs.

#9 - 25 avril 2020 07:14 - Benjamin Dauvergne

Nicolas Roche a écrit :

> Oui, je l'ai utilisé dans le premier patch.
> Ensuite on peut relancer les Jobs avec SkipJob, mais j'ai l'impression qu'alors on ne peut pas alors garder traces des erreurs.

Tu auras deux types d'erreurs, les récupérables et les non récupérables[1]:

- pour les récupérables tu les logs et tu raises SkipJob,
- pour les non récupérables tu les raises et ce sera enregistré dans le job par handle_job_error()

En v1 ça me parait bien suffisant, il suffira de chercher un numéro de téléphone dans les logs pour savoir ce qui lui est arrivé (pourvu que ce soit loggé correctement).

Dans un seconds temps si tu veux ajouter une gestion des logs par job tu pourras le faire :

- différencier dans GenericViewJobsConnectorView les jobs qui n'ont jamais été exécutés des jobs en retry (visible via creation_timestamp != update_timestamp),
- trouver un moyen de passer un ProxyLogger particulier à la méthode exécutée construit via resource.log.context(job_id=job.id) pour pouvoir afficher les logs d'un job en particulier (via @ResourceLog.objects.filter(extra__job_id=job.id)).

Mais c'est un travail plus global à penser pour bénéficier à tous les connecteurs.

[1] Ici l'absence d'accès à l'objet job depuis la méthode appelée rend les choses simples et empêche de faire des trucs sioux ; mais c'est vrai que pour gérer une politique de retry un peu complexe ça nous limite (genre essayer 10 fois, en attendant 5, puis 10, 30, 60, 120, 240, etc.. minutes, sans savoir quand on a commencé ou combien de fois on a essayé c'est pas simple). On pourrait prévoir un attribute nees_job = True ajouté à la méthode et dans ce cas on lui passe le job en premier paramètre.

#10 - 25 avril 2020 07:23 - Benjamin Dauvergne

- Lié à Development #42090: Avoir des logs par job ajouté

#11 - 25 avril 2020 07:25 - Benjamin Dauvergne

Un truc que tu peux faire aussi c'est alloué un natural_id à tes jobs pour pouvoir retrouver les jobs qui étaient ensembles et plus tard pouvoir récupérer leur statut groupé; natural_id que tu peux renvoyer dans ta réponse à l'appelant.

#12 - 25 avril 2020 14:07 - Frédéric Péters

De manière pas générique, à travailler avec un modèle dédié, il y a le modèle SMSLog introduit dans #39651 qui pourrait être étendu, avec les informations de contenu/destinataire de message, et statut de l'envoi, on pourrait alors avoir les choses découpées :

- job qui reçoit l'id d'un enregistrement "SMSLog", tente l'envoi,
- cron hourly qui prend tous les SMSLog pas réussis à être transmis, vieux entre 5 minutes et ?, et retente,
- sur succès d'envoi les infos destinataire et contenu du message seraient vidées.

#13 - 26 avril 2020 11:04 - Nicolas Roche

- Fichier 0002-sms-add-endpoint-to-send-sms-asynchronously-21465.patch ajouté

- Fichier 0001-sms-factorise-sms-code-on-parameters-21465.patch ajouté

Voici la version 1, avec renvoie à l'appelant d'un natural_id.
(les envois sont retentés toutes les heures)

#14 - 26 avril 2020 11:21 - Benjamin Dauvergne

Pour natural_id j'aurai juste mis un str(uuid.uuid4()) là on va avoir le même id pour un même envoi (et encore ça dépendra de l'ordre des numéros dans destinations, enfin ça se discute).

#15 - 26 avril 2020 11:22 - Frédéric Péters

0002 me semble dupliquer tout le code d'envoi.

Aussi je notais plus haut :

> Il faudrait que ça soit tout le temps asynchrone [...]

Je voulais dire par là que l'endpoint send devait être asynchrone, qu'il ne devait pas y avoir de nouvel endpoint.

#16 - 26 avril 2020 11:44 - Frédéric Péters

Très basiquement, quand j'ai réévoqué ce ticket, je pensais juste à zéro modif côté connecteurs, quelque chose de global, approchant ceci :

```
@@ -962,9 +962,15 @@ class SMSResource(BaseResource):
         logging.info('sending message %r to %r with sending number %r',
                      data['message'], data['to'], data['from'])
         stop = not bool('nostop' in request.GET)
-        result = {'data': self.send_msg(data['message'], data['from'], data['to'], stop=stop)}
+        self.add_job('send_job', stop=stop, **data)
+        return {'err': 0}
+
+    def send_job(self, **kwargs):
+        try:
+            self.send_msg(**kwargs)
+        except whatever:
+            # deal with it
+            return
         SMSLog.objects.create(appname=self.get_connector_slug(), slug=self.slug)
-        return result
```

qui correspond assez à ton premier patch; modulo mon commentaire "tout le temps async".

Et puis je ne comprends pas comment ce commentaire "tout le temps async" amène à la production de la nouvelle série, qui commence à modifier partout, ajouter des modèles, etc. et une discussion qui part là-dessus, alors que non, c'est à la simplicité d'origine qu'il faut retourner.

#17 - 26 avril 2020 14:23 - Nicolas Roche

Désolé, c'est moi qui ai mal interprété.

Mon commentaire été lié au sujet du ticket :

> Il faudrait gérer une file des messages, que l'envoi se passe de manière asynchrone.

Mon commentaire n'étais vraiment pas (du tout) clair.

> Je précise que ce patch ne gère pas l'envoie d'un lot de messages de façon asynchrone.

Je voulais dire en fait 2 choses :

- il n'y a pas de reprise sur exception
- les messages ne sont pas envoyés de façon unitaire

J'ai sur-interprété ta réponse avec la pensée implicite que les SMS devaient être envoyés de façon unitaire : si l'on envoie tous les SMS (un même texte peut être adressé à plusieurs destinataires via un unique appel au endpoint), et que seul le n-ième SMS plante, alors on va renvoyer tous les autres SMS une seconde fois.

#18 - 26 avril 2020 14:54 - Frédéric Péters

Je suis pour ne pas vouloir en faire trop, abysse, déjà 17 commentaires, etc. Limitons ce ticket à l'envoi aynchrone, qui n'offrira pas plus de gestion d'erreurs que la situation présente.

On verra ensuite dans un autre ticket pour s'occuper de la gestion des erreurs, là on pourra avoir la question "quid si un message dans une série échoue ?", là on pourra s'interroger sur le moment ou les moments où tenter la réexpédition, etc.

#19 - 26 avril 2020 18:30 - Nicolas Roche

- Fichier 0001-sms-send-SMS-asynchronously-21465.patch ajouté

Maintenant que c'est simplifié, je ne m'en sort pas pour autant :

- J'ai du inventer une erreur libellée "connection timed-out" pour les tests, parce que les connecteurs ne retranscrivent pas le nom de l'exception dans leurs messags d'erreur :

```
except requests.RequestException as e:
    results.append('XXX error: %s' % e)
pdb> str(e)
(vide)
```

- J'ai du retirer 2 tests sur "parameters.responses_max_size" parce que le 'body' n'est pas logué dans les Jobs.
  Est-ce qu'il faut que je réécrive test_api_access.py en me basant sur un autre connecteur ?

#20 - 26 avril 2020 19:00 - Frédéric Péters

> - J'ai du inventer une erreur libellée "connection timed-out" pour les (...)

Parce que tu négliges "On verra ensuite dans un autre ticket pour s'occuper de la gestion des erreurs.". Laisse ça à un autre ticket, où ça pourra être approché spécifiquement, où on pourra se dire qu'il y a à uniformiser des bouts entre connecteurs plutôt qu'essayer de matcher des bouts de chaines.

> J'ai du retirer 2 tests sur "parameters.responses_max_size" parce que le 'body' n'est pas logué dans les Jobs.

C'est peut-être un problème en soit, par quelle mécanique celui-ci devrait être loggué, pourquoi ne l'est-il pas ici, ça peut faire l'objet d'un ticket séparé, etc.

> Est-ce qu'il faut que je réécrive test_api_access.py en me basant sur un autre connecteur ?

J'aurais volontiers pointé que la gestion des accès ne devrait pas changer.

Mais à regarder le patch, elle ne change pas, il y a juste zéro raison à la présence d'un "test_logged_requests_and_responses_max_size" dans test_api_access.py.

Ça me semble plutôt relever de tests/test_proxylogger.py.

#21 - 28 avril 2020 11:59 - Nicolas Roche

- Lié à Bug #42187: Ne pas faire dépendre les tests ProxyLogger d'un connecteur spécifique. ajouté

#22 - 28 avril 2020 19:27 - Nicolas Roche

- Lié à Development #42230: Uniformiser les retours des connecteurs SMS ajouté

#23 - 28 avril 2020 19:28 - Nicolas Roche

- Statut changé de Solution proposée à En cours

#24 - 29 avril 2020 15:10 - Nicolas Roche

- Fichier 0001-sms-send-SMS-asynchronously-21465.patch ajouté

- Statut changé de En cours à Solution proposée

Basé sur #42230

#25 - 25 mai 2020 10:17 - Frédéric Péters

J'ai noté dans #42230 qu'il ne me semble pas opportun. Et k'ai noté ici, plus haut, "On verra ensuite dans un autre ticket pour s'occuper de la gestion des erreurs.".

Vraiment : d'abord l'envoi asynchrone, ensuite la gestion des erreurs.

#21465#note-18

#26 - 25 mai 2020 15:40 - Nicolas Roche

- Fichier 0001-sms-send-SMS-asynchronously-21465.patch ajouté

Sans la gestion des erreurs.
Testé avec Twilio :

```
        $ passerelle-server
$ curl https://passerelle.dev.publik.love/twilio/test/send -d '{"message": "coucou", "from": "+12015849510", "
to": ["+330695036993"]}'
{"err": 0}

        endpoint POST /twilio/test/send ('{"message": "coucou", "from": "+12015849510", "to": ["330695036993"]
}')
        sending SMS to ['330695036993'] from '+12015849510'

$ passerelle-manage tenant_command cron -v2 --connector twilio jobs --domain passerelle.dev.publik.love
https://api.twilio.com:443 "POST /2010-04-01/Accounts/ACe1e8c7acefe6c0a76d1e68d815e460f9/Messages.json HTTP/1.
1" 201 808
POST https://api.twilio.com/2010-04-01/Accounts/ACe1e8c7acefe6c0a76d1e68d815e460f9/Messages.json (=> 201)
```

#27 - 25 mai 2020 15:45 - Frédéric Péters

```
    def send_job(self, *args, **kwargs):
        try:
            self.send_msg(**kwargs)
        except APIError:
            raise  # not recoverable
```

Ça rend l'erreur silencieuse, faut pas.

#28 - 25 mai 2020 16:21 - Nicolas Roche

- Fichier 0001-sms-send-SMS-asynchronously-21465.patch ajouté

En fait j'ai oublié de retirer le try/except qui ne sert plus à rien (puisque je re-raise derrière).

```
$ curl https://passerelle.dev.publik.love/twilio/test/send -d '{"message": "coucou", "from": "+12015849510", "
to": ["+3369503699300"]}'
$ passerelle-manage tenant_command cron -v2 --connector twilio jobs --domain passerelle.dev.publik.love
...
Traceback (most recent call last):
  File "/home/nroche/src/passerelle/passerelle/base/models.py", line 545, in jobs
    getattr(self, job.method_name)(**job.parameters)
  File "/home/nroche/src/passerelle/passerelle/base/models.py", line 972, in send_job
    self.send_msg(**kwargs)
  File "/home/nroche/src/passerelle/passerelle/apps/twilio/models.py", line 87, in send_msg
    raise APIError(
passerelle.utils.jsonresponse.APIError: Twilio error: some destinations failed

# select * from base_resourcelog;
```

```
...
931 | 2020-05-25 16:02:20.158987+02 | twilio        | test                                    |      40 |
  | POST https://api.twilio.com/2010-04-01/Accounts/ACe1e8c7acefe6c0a76d1e68d815e460f9/Messages.json (=> 400)
| ...
932 | 2020-05-25 16:02:20.176716+02 | twilio        | test                                    |      40 |
  | error running send_job job (Twilio error: some destinations failed)  | ...
```

#29 - 25 mai 2020 16:40 - Frédéric Péters

- Statut changé de Solution proposée à Solution validée

Ok ainsi, à noter en évolution, ça pourrait retourner l'id du job (ou une url de récupération de statut de celui-ci, j'ai créé #43278 pour cette suite).

#30 - 29 mai 2020 11:33 - Nicolas Roche

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

```
commit 115b0bb78c953ad5c3c29f46e5de4b7143a42699
Author: Nicolas ROCHE <nroche@entrouvert.com>
Date:   Sun Apr 26 15:54:47 2020 +0200

    sms: send SMS asynchronously (#21465)
```

#31 - 29 mai 2020 14:16 - Frédéric Péters

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

Fichiers

0001-sms-add-endpoint-to-send-SMS-asynchronously-21465.patch	6,99 ko	10 avril 2020	Nicolas Roche
0002-sms-factorise-sms-code-on-parameters-21465.patch	5,97 ko	24 avril 2020	Nicolas Roche
0003-sms-add-endpoint-to-send-sms-asynchronously-21465.patch	18 ko	24 avril 2020	Nicolas Roche
0001-sms-add-SMS-class-to-store-sms-parameters-21465.patch	24,2 ko	24 avril 2020	Nicolas Roche
0001-sms-factorise-sms-code-on-parameters-21465.patch	5,98 ko	26 avril 2020	Nicolas Roche
0002-sms-add-endpoint-to-send-sms-asynchronously-21465.patch	17,9 ko	26 avril 2020	Nicolas Roche
0001-sms-send-SMS-asynchronously-21465.patch	12,8 ko	26 avril 2020	Nicolas Roche
0001-sms-send-SMS-asynchronously-21465.patch	11,4 ko	29 avril 2020	Nicolas Roche
0001-sms-send-SMS-asynchronously-21465.patch	9,5 ko	25 mai 2020	Nicolas Roche
0001-sms-send-SMS-asynchronously-21465.patch	9,41 ko	25 mai 2020	Nicolas Roche