Debian Support - Issue #7756

Content upload has different method-definintion than rpm/file/etc.

10/27/2020 04:58 PM - mbucher

Status:	CLOSED - WONTFIX	**Start date:**	
Priority:	Low	**Due date:**	
Assignee:		**Estimated time:**	0:00 hour
Category:			
Sprint/Milestone:			
Severity:	2. Medium	**Groomed:**	No
Version - Debian:		**Sprint Candidate:**	No
Platform Release:		**Tags:**	API Bindings, Katello
Target Release - Debian:		**Sprint:**	
OS:		**Quarter:**	
Triaged:	No		

Description

Looking at the generated API-Bindings for ruby, the method for creating/uploading a new deb-package differs from the one used for e.g. rpm.

```
module PulpDebClient
  class ContentPackagesApi
...
    def create(opts = {})
```

While in the RPM-bindings it looks like this:

```
module PulpRpmClient
  class ContentPackagesApi
...
    def create(relative_path, opts = {})
```

The fact that relative_path is specified as parameter in the latter case seems to be due to the fact that in the API, relative_path is marked as required in pulp_rpm, but not in pulp_deb.

https://pulp-deb.readthedocs.io/en/latest/restapi.html#operation/content_deb_packages_create

https://pulp-rpm.readthedocs.io/en/latest/restapi.html#operation/content_rpm_packages_create

THis came up during the katello integration of pulp_deb for pulpcore.

History

#1 - 10/28/2020 10:04 PM - mbucher

Currently solved this the katello-PR by using a wrapper-method, which should be fine.

If there is a good reason for pulp_deb diverging from how the API works for the other content-types, I am ok with keeping it that way.

Discussion welcome :-)

#2 - 11/02/2020 03:47 PM - quba42

I can provide at least some initial background for this:

pulp_deb currently generates the relative path for packages using various control file fields, to produce the default pool structure that upstream Debian repos use. (Something like pool/<component>/<first_letter_of_package_name_or_lib_plus_fourth_letter_of_package_name>/<package_name>/<file_name>, where <file_name> also follows some default structure).

If I am not mistaken the current implementation runs into trouble if the relative path for a package does not use this default format (requires further testing for details), therefore it is very much "recommended" not to have users provide a relative_path (and rely on the default path generating

mechanism instead).

Of course we could consider changing the implementation to permit (more or less) arbitrary relative paths for packages. The question is whether it is worth the effort if there is a simple workaround on the Katello side.

#3 - 04/27/2021 05:49 PM - quba42

- Sprint/Milestone set to Katello

Still not sure if we want or should do anything about this. I will add it to the Katello milestone for now.

#4 - 04/28/2021 09:32 AM - quba42

- Priority changed from Normal to Low

Setting this to Low priority since it is unclear if anything really needs doing.

The way I see it there is a good reason for the differences in code, leading to a difference in API bindings.

There is a workaround within Katello, so the only argument for doing anything is that this Katello based workaround might be liable to breakage in the future...

#5 - 11/02/2021 11:02 AM - quba42

- Status changed from NEW to CLOSED - WONTFIX

#6 - 11/02/2021 11:03 AM - quba42

- Sprint/Milestone deleted (Katello)