RPM Support - Issue #9364

feature request: include/exclude/filter packages in remotes

09/09/2021 11:17 AM - wheezer

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

Description

Hi all,

I'm using reposync to create a local copy of remote rpm repositories, fetching only the latest package release and excluding packages that conflict with other repositories, or including only a few packages I need from that source, i.e.:

- repository **foo** provides nicepackage-1.2.3-1 with is compiled with --nifty-feature but repository **bar** provides nicepackage-1:1.2.3-1 which was compiled without --nifty-feature. Because of the epoch the version without --nifty-feature would be installed. I don't want to fetch that rpm at all, so I exclude nicepackage from remote **bar**.
- repository **baz** provides lots of packages, most of them are huge disk-space-wasting X-applications (e.g. gnome desktop, firefox browser, stuff you don't need on a headless server), but there are also some nice cli-tools I'd like to mirror locally. Including only niceclitool from that remote is the way to go.

Now I'd like to replace it with pulp. Unfortunately a *remote* cannot be filtered. So here's my feature request: Could you please implement filters in rpm remotes?

Thanks :)

History

#1 - 09/09/2021 05:56 PM - dalley

```
repository foo provides nicepackage-1.2.3-1 with is compiled with --nifty-feature but repository bar provi
des nicepackage-1:1.2.3-1 which was compiled without --nifty-feature. Because of the epoch the version wit
hout --nifty-feature would be installed. I don't want to fetch that rpm at all, so I exclude nicepackage f
rom remote bar.

repository baz provides lots of packages, most of them are huge disk-space-wasting X-applications (e.g. gn
ome desktop, firefox browser, stuff you don't need on a headless server), but there are also some nice cli
-tools I'd like to mirror locally. Including only niceclitool from that remote is the way to go.
```

I'm not sure that the suggested workflow is the best way to acheive these goals.

Disk space usage: There are a few different ways to address this

- Use "yum download $package" to download only the packages you want, and then upload them into Pulp (or use a CLI tool like createrepo_c to create repository metadata)
- use "on_demand" syncing, which doesn't download any of the packages until a client requests the file. If no client ever downloads the package then the disk space won't be wasted, and don't need to worry about manually filtering packages.

Alternative versions of a package: I'm assuming these are your own RPMs? Using epochs for this is a really bad idea, because you can never turn back. Higher epochs always rank as higher versions, so if you want to switch back to the version compiled with the original options, then you must bump the epoch again, and vice versa, every single time.

There's two ways you could do it:

- Have two differently named packages that both "provide" the same dependency, but conflict with each other. For example the "docker" command can be provided by either the "podman-docker" package or by the "moby" package. libpulse can be provided by either "pulseaudio" or "pipewire-pulse"
- Use RPM modules. You can have multiple different "streams" of the same package, and which one is installed depends on which stream is enabled on the system. e.g. you could install "nodejs:8" or "nodejs:10" https://docs.fedoraproject.org/en-US/modularity/installing-modules/

The trouble is that filtering the upstream repositories as you suggest could be very problematic. For one, it would conflict with the "mirror" option brutally, but additionally it could result in totally broken repositories. If you filter everything out of a repo except for a few packages, you might also be filtering out some or all of the dependencies.

#2 - 09/13/2021 11:21 PM - dalley

- Status changed from NEW to CLOSED - WONTFIX

@wheezer This is still open for discussion btw even though I'm closing it, if there are bits of the experience that could be improved we'd love to do that.

#3 - 09/14/2021 11:40 AM - wheezer

Hi dalley,

sorry for the late response, I didn't get any notification from plan.io about new replies. Thanks for your explanation.

It's not about only own repositories but about mirrors of official repos, too.

Let's forget about the epoch and just have two repos (not mine, so I have no control over conflicts/obsoletes) provide nicepackage – one with --nifty-feature, the other without it. Both repos try to keep the version up to date so we have nicepackage-1.2.3-1 in repo foo **and** in repo bar. Afaik that cannot be solved by using modules, also iirc modules are not really a thing in rhel/centos/… 7.

Take ecryptfs-utils as an example. Both **epel** and **elrepo** provide the package:

```
$ dnf --showduplicates list ecryptfs-utils
Available Packages
ecryptfs-utils.x86_64            111-1.el7.elrepo        elrepo
ecryptfs-utils.x86_64            111-5.el7               epel
```

From **elrepo** I only need some kmods.

Even worse, in my setup I merge repositories. I have centos-base, centos-updates, epel, elrepo, mariadb, postgresql, mongodb, and many more merged together into a "super"-repository using mergerepo_c so that the clients will only have to include this single repo. This makes it impossible to add includepkgs and/or exclude in the repo file.

Regarding the dependency solving – normally, when including/excluding packages from a repository, I tested the dependencies beforehand – so I could do without an automatism. But I see your point.

The only way to solve this issue seems to be to mirror all of these repositories using reposync having includepkgs/exclude options in the corresponding repo files and then injecting all retrieved packages into a pulp repo. Looks quite redundant to me, I could just keep the packages and run createrepo_c (and modifyrepo_c for the the updateinfo) over the target folder. Guess I'll stick to that method for remote repos then :).

Thanks again for your feedback.