Arvados - Bug #14236

[WebDAV] Can't delete the last file in a collection

09/19/2018 03:34 PM - Tom Morris

Status:	Resolved	**Start date:**	09/21/2018
Priority:	Normal	**Due date:**	
Assigned To:	Tom Clegg	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	2018-10-03 Sprint		

Description

Subtasks:

Task # 14240: Review 14236-delete-last-file **Resolved**

Associated revisions

Revision 4a16f09d - 09/27/2018 02:03 PM - Tom Clegg

Merge branch '14236-delete-last-file'

fixes #14236

Arvados-DCO-1.1-Signed-off-by: Tom Clegg <tclegg@veritasgenetics.com>

History

#1 - 09/19/2018 03:34 PM - Tom Morris

- Status changed from New to In Progress

#2 - 09/19/2018 03:34 PM - Tom Morris

- Status changed from In Progress to New

#3 - 09/19/2018 03:50 PM - Tom Clegg

- Assigned To set to Tom Clegg

#4 - 09/19/2018 08:22 PM - Tom Clegg

This is caused by "omitempty" in arvados.Collection's json struct tag. After deleting the last file, manifest_text is empty, so manifest_text is omitted from the update request body.

```
        ManifestText               string     `json:"manifest_text,omitempty"`
```

#5 - 09/19/2018 08:22 PM - Tom Clegg

- Status changed from New to In Progress

#6 - 09/19/2018 09:03 PM - Tom Clegg

14236-delete-last-file @ f0af3b1225e5617b0c2635ff8466d6d8b89e50cahttps://ci.curoverse.com/view/Developer/job/developer-run-tests/899/

#7 - 09/21/2018 08:33 PM - Peter Amstutz

Tom Clegg wrote:

> 14236-delete-last-file @ f0af3b1225e5617b0c2635ff8466d6d8b89e50cahttps://ci.curoverse.com/view/Developer/job/developer-run-tests/899/

Instead of removing omitempty, should we convert this into a pointer? I could easily see someone writing code intended to update another field (like name or properties) and accidentally clobbering manifest_text.

It sucks but the least bad way to interoperate Go structs with json seems to be to make everything pointers (that's how the Azure Go SDK works, for example).

#8 - 09/25/2018 06:09 PM - Tom Clegg

IMO that pointer approach is an awkward way to coerce the omitempty tag into implementing selective updates: it adds repetitive new() and nil

checks, and is susceptible to different kinds of bugs, like copying a struct and then accidentally modifying both instead of just the copy. It also doesn't go far enough to address all cases (how do I update to NULL?). I agree it is desirable to offer a selective-update mechanism that prevents callers from accidentally overwriting non-zero values with default zero values, but I'm not convinced pointers will be the answer, and I don't think we should block this bugfix while we figure it out.

(As an aside, the Azure SDK doesn't necessarily do everything wrong, but I wouldn't use it as evidence that something is a good idea.)

FWIW I did look through the code and API test logs for examples of Go-default-zero values being passed in updates but didn't find any.

#9 - 09/27/2018 01:47 PM - Peter Amstutz

Tom Clegg wrote:

> IMO that pointer approach is an awkward way to coerce the omitempty tag into implementing selective updates: it adds repetitive new() and nil checks, and is susceptible to different kinds of bugs, like copying a struct and then accidentally modifying both instead of just the copy. It also doesn't go far enough to address all cases (how do I update to NULL?). I agree it is desirable to offer a selective-update mechanism that prevents callers from accidentally overwriting non-zero values with default zero values, but I'm not convinced pointers will be the answer, and I don't think we should block this bugfix while we figure it out.

I did a quick audit of the code base. Mostly we use the Go SDK for reading, not writing. I did find one place in fs_collection.go where we do a partial update, but that operation includes ManifestText, so it wouldn't be broken by this change. So although I think this has a high potential for future mistakes, I don't think it breaks any existing code right now.

> (As an aside, the Azure SDK doesn't necessarily do everything wrong, but I wouldn't use it as evidence that something is a good idea.)

I use it as evidence that other people have thought about the problem and didn't come up with a more elegant solution.

> FWIW I did look through the code and API test logs for examples of Go-default-zero values being passed in updates but didn't find any.

That's what "omitempty" means, right? Don't send the value if it has Go-default-zero. Since we use that everywhere, that's what we'd expect... except for those fields where the default-zero value is meaningful. Another case that comes to mind is Container Request priority, which has the same ambiguity priority to 0.

So I guess you should merge the quick fix on the expectation we're going to revisit it, we need to decide on a pattern because this is a recurring problem across the whole Go SDK.

#10 - 09/27/2018 02:13 PM - Tom Clegg

- Status changed from In Progress to Resolved

Applied in changeset arvados|4a16f09d3df0c7899fa6367d27b9f66ee9d4582b.

#11 - 11/13/2018 08:49 PM - Tom Morris

- Release set to 14