CN REST - Task #7911

Synchronization allows invalid checksums, preventing corrected sync

2016-10-17 15:16 - Chris Jones

Status:	New	**Start date:**	2016-10-17
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	d1_synchronization	**Estimated time:**	0.00 hour
Target version:			
Story Points:			

Description

Normally, d1_synchronization does checksum validation of objects before registering them in the CN. However, a CHECKSUM_VERIFICATION_SIZE_BYPASS_THRESHOLD flag was introduced into TransferObjectTask that defaults to 10MB. If an object size is greater than this threshold, the checksum won't be verified. In the cn-buildout, this default is not changed in the properties file, but it can be.

As a result, objects below this threshold will throw an exception during sync if the checksum is incorrect, whereas those above the threshold will successfully sync with incorrect system metadata. This becomes a problem later when trying to update the system metadata with the correct checksum because this field is immutable. For example, in the STAGE environment, the following object failed to process the system metadata update:

[ERROR] 2016-10-17 14:54:38,112 (V2TransferObjectTask:call:269)
Task-urn:node:mnTestARCTIC-urn:uuid:a3bfef74-f6e9-4ecc-871e-0a3ea764b471 - UnrecoverableException: Failed to update cn with new valid SystemMetadata! - InvalidRequest - The request is trying to modify an immutable field in the SystemMeta: the new system meta's checksum dee03804421bac149371877d2d366abb7c941fba is different to the orginal one bef6df568ed1c713a83234346943198 94f25a8b9dfa704f7fe2b7d52592b2b40

Since MN.getChecksum() is normally being called to do the heavy lifting of calculating the actual checksum, I'm not sure why this flag was introduced. Even for muti-gigabyte files, the checksum calculation is pretty quick. To prevent the CN from ingesting incorrect system metadata, I'd suggest we consider removing this threshold, or at a minimum, set the property value to be multi-terabyte. Also, this is a case where the MN is authoritative for the system metadata, but the CN update fails because of the immutable status of the checksum. Ultimately, we shouldn't be sync'ing content with invalid checksums, which allows for the MN operator to correct the checksum and then retry the sync. Needs discussion.

History

#1 - 2016-10-17 15:33 - Chris Jones

After troubleshooting this in the TransferObjectTask code, I realized my example above isn't representative of this problem - it's only 17KB. In this case, the stated checksum was an SHA256 sum, which in theory isn't supported. The replacement was an SHA1 sum. Synchronization happens to accept this system metadata because the underlying Java MessageDigest supports the SHA256 algorithm. But technically it probably should have failed.

Anyway, I think this ticket is still an issue - seems like all checksums (even for big objects) need to be correct.

#2 - 2016-10-17 15:52 - Matthew Jones

Why do you say SHA256 is not allowed? It is explicitly in the list of checksums at the Library of Congress. We do require nodes to support MD5 and SHA-1, but others are allowed too. Here's the text from the definition from Checksum Algorithm:

"The cryptographic hash algorithm used to calculate a checksum. DataONE recognizes the Library of Congress list of cryptographic hash algorithms that can be used as names in this field, and specifically uses the madsrdf:authoritativeLabel field as the name of the algorithm in this field. See: Library of Congress Cryptographic Algorithm Vocabulary. All compliant implementations must support at least SHA-1 and MD5, but may support other algorithms as well."

#3 - 2016-10-17 15:58 - Chris Jones

Ah right, my mistake. I thought we ONLY supported SHA1 and MD5. Sounds good!