Bit Integrity Checker

Overview

The Bit Integrity Checker is intended to provide a simple and easy to use way of assuring that the content stored in your DuraCloud account has maintained bit integrity. The basic idea is that a DuraCloud account administrator provides a listing of expected content IDs and their associated MD5s. This listing is then used by the service as a basis for comparison against MD5s found in DuraCloud.

There is a trade-off between cost (both in time and money) and assurance in the trustworthiness of the MD5 provided by DuraCloud. The service is designed to offer three options to address this balance, see "Levels of trust" below. The fastest and cheapest option is to use the MD5 stored in the metadata of the content item. The underlying storage providers assert that this value which is created on ingest is also checked when the content item is read, and retrieves a mirrored copy of the content if there is a mismatch. As a note, when content is pushed into DuraCloud via the DuraStore REST-API an MD5 can be provided by the user to be automatically checked with the one generated by the underlying storage provider, and if no MD5 is provided the DuraCloud application calculates it for this comparison. If the administrator does not trust the assertion of the underlying storage provider, the Bit Integrity Checker also provides the option of reading the content and recalculating the MD5. Finally, if the administrator does not trust this recalculation, then the Bit Integrity Checker also provides the option of passing in a "salt" character string which will be appended to the content during the recalculation of the MD5.

The input listing is expected to be found as a user-specified content item within DuraCloud, and the resultant output file will be stored to a user-specified location within DuraCloud. The formats of the input and output files are the same, so a previous run's output may be used as a subsequent run's input.

see July 2010 NDIIPP presentation

Requirements

1. User has the option to provide as input the listing of content items with expected MD5s
2. User has the option to provide as input the listing of content items without expected MD5s
3. Service will determine the system MD5 value for each content item according to selected "Level of trust" algorithm
4. Service will store the determined listing in a user-provided DuraCloud location
5. User has the option to specify any two listings of content items / MD5 pairs for comparison
6. Service will report on comparison of provided MD5s against system MD5s values of content items
7. Service will store report output in user-provided DuraCloud location
8. Service will run on a compute instance local to the input storeId - *future*

Design

Levels of trust

Provides the choice of balance amongst cost, time, and assurance

1. Trust in underlying storage providers
2. Trust in DuraCloud and opensource software
3. Trust in requester of service

The three levels of trust above are addressed by three implementations

1. System MD5s are determined by using the stored metadata values
2. System MD5s are determined by DuraCloud re-reading the content bytes and re-calculating the MD5s
3. System MD5s are determined by DuraCloud re-reading the content bytes appended with a 'salt' and re-calculating the MD5s

Operational modes

In order to address both scenarios of allowing the user to have certainty that MD5s are being generated/checked when requested and allowing the user to trust the service and have it execute with a single command, the following modes are available.

1. Single-step interaction
   a. User invokes service with a listing of contentId/**MD5 pairs** (and other options) to check
   b. Bit Integrity Checker generates a listing of contentId/MD5 pairs based on the input options **and performs comparison with input, expected MD5s**

    c. Service generates result report
2. Two-step interaction
    a. User invokes service with a listing of contentIds (and other options) to check
    b. Bit Integrity Checker generates a listing of contentId/MD5 pairs based on the input options
    c. User invokes service with a listing of expected contentId/MD5 pairs to be compared to the generated listing
    d. Service generates result report

Functionality spec

Single-step interaction

1. Service inputs
    a. spaceId & contentId where input listing is stored
        - comma-delimited listing of spaceId, contentId and expected MD5
        - each content item separated by newline character
        - first line in file will be ignored
    b. spaceId & contentId where results file should be written
    c. options (see below)
2. Service outputs
    a. comma-delimited listing of spaceIds, contentIds, expected MD5s, system MD5s, status state
    b. service status state

Two-step interaction

1. Step 1: Service inputs
    a. spaceId & contentId where input listing is stored
        - comma-delimited listing of spaceId, contentId and **without** expected MD5
        - each content item separated by newline character
        - first line in file will be ignored
    b. spaceId & contentId where results file should be written
    c. options (see below)
2. Step 1: Service outputs
    a. comma-delimited listing of spaceIds, contentIds, system MD5s
    b. service status state
3. Step 2: Service inputs
    a. spaceId & contentId where input listing is stored
        - comma-delimited listing of spaceId, contentId and expected MD5
        - each content item separated by newline character
        - first line in file will be ignored
    b. spaceId & contentId where service-generated listing is stored
        - comma-delimited listing of spaceId, contentId and system MD5
        - as generated and stored by the the service in step-1
4. Service outputs
    a. comma-delimited listing of spaceIds, contentIds, expected MD5s, system MD5s, status state
    b. service status state

Service options

1. trust level (stored value, recalculate, salt)
2. salt
    - arbitrary character string which will be appended to content in generating MD5
3. fail-fast boolean
    - service will exit when first error/mismatch found if 'true'
4. complete space(s) boolean
    - indicates if the input listing should be checked against the complete set of items in the space(s)
5. storeId of underlying storage provider
    - default to primary underlying storage provider

Service exceptions

1. Checked
    - Contains the following enum
    a. missing MD5 (expected or found)
    b. MD5 mis-match
    c. unequal content listings
2. Runtime
    a. internal error
    b. salt option set but salt not provided
    c. service level not supported
    d. input content item not exists
    e. output result content item already exists

Options / Mode matrix

The table below shows the possible usage scenarios across the top, and their associated input options.

1. One-step: hash from input list

    - user initiates Bit Integrity Checker as a single operation
    - an input listing of content-ids/hashes is provided
    - service generates hashes one-to-one for each item in the input listing
2. One-step: hash from complete space
    - user initiates Bit Integrity Checker as a single operation
    - an input listing of content-ids/hashes is provided
    - service generates hashes for **all** content-items in spaces found in the input listing
3. Two-step: hash from input list
    - user initiates Bit Integrity Checker as a two part operation, this being the first
    - an input listing of only content-ids is provided
    - service generates hashes one-to-one for each item in the input listing
4. Two-step: hash from complete space
    - user initiates Bit Integrity Checker as a two part operation, this being the first
    - user indicates space(s) of target content to hash
5. Two-step: compare two lists
    - user initiates Bit Integrity Checker as a two part operation, this being the second
    - two input listings of content-ids/hashes are provided

User input option	One-step: hash from input list	One-step: hash from complete space	Two-step: hash from input list	Two-step: hash from complete space	Two-step: compare two lists
hash approach	✓	✓	✓	✓	
salt	✓	✓	✓	✓	
fail-fast	✓	✓			✓
storage provider id	✓	✓	✓	✓	✓
space of provided listing	✓	✓	✓		✓
object-id of provided listing	✓	✓	✓		✓
space of provided listing-B					✓
object-id of provided listing-B					✓
space(s) of target content		✓		✓	
space for output	✓	✓	✓	✓	✓
object-id of result listing	✓	✓	✓	✓	
object-id of report	✓	✓			✓