Validating CheckSums of Bitstreams

1 Checksum Checker
- 1.1 Checker Execution Mode
- 1.2 Checker Results Pruning
- 1.3 Checker Reporting
- 1.4 Cron or Automatic Execution of Checksum Checker
- 1.5 Automated Checksum Checkers' Results
- 1.6 Database Query

Checksum Checker

Checksum Checker is program that can run to verify the checksum of every item within DSpace. Checksum Checker was designed with the idea that most System Administrators will run it from the cron. Depending on the size of the repository choose the options wisely.

Command used:	`[dspace]/bin/dspace checker`
Java class:	org.dspace.app.checker.ChecksumChecker
Arguments short and (long) forms):	Description
`-L` or `--continuous`	Loop continuously through the bitstreams
`-a` or `--handle`	Specify a handle to check
`-b <bitstream-ids>`	Space separated list of bitstream IDs
`-c` or `--count`	Check count
`-d` or `--duration`	Checking duration
`-h` or `--help`	Calls online help
`-l` or `--looping`	Loop once through bitstreams
`-p <prune>`	Prune old results (optionally using specified properties file for configuration
`-v` or `--verbose`	Report all processing

There are three aspects of the Checksum Checker's operation that can be configured:

- the execution mode
- the logging output
- the policy for removing old checksum results from the database
  The user should refer to Chapter 5. Configuration for specific configuration beys in the *dspace.cfg* file.

Checker Execution Mode

Execution mode can be configured using command line options. Information on the options are found in the previous table above. The different modes are described below.

Unless a particular bitstream or handle is specified, the Checksum Checker will always check bitstreams in order of the least recently checked bitstream. (Note that this means that the most recently ingested bitstreams will be the last ones checked by the Checksum Checker.)

Available command line options

- **Limited-count mode:** `[dspace]/bin/dspace checker -c` To check a specific number of bitstreams. The *-c* option if followed by an integer, the number of bitstreams to check. Example: `[dspace/bin/dspace checker -c 10` This is particularly useful for checking that the checker is executing properly. The Checksum Checker's default execution mode is to check a single bitstream, as if the option was *-c 1*
- **Duration mode:** `[dspace]/bin/dspace checker -d` To run the Check for a specific period of time with a time argument. You may use any of the time arguments below: Example: `[dspace/bin/dspace checker -d 2h`(Checker will run for 2 hours)

s	Seconds
m	Minutes
h	Hours
d	Days
w	Weeks
y	Years

The checker will keep starting new bitstream checks for the specific durations, so actual execution duration will be slightly longer than the specified duration. Bear this in mind when scheduling checks.

- **Specific Bitstream mode:** `[dspace]/bin/dspace checker -b` Checker will only look at the internal bitstream IDs. Example: `[dspace]/bin/dspace checker -b 112 113 4567` Checker will only check bitstream IDs 112, 113 and 4567.
- **Specific Handle mode:** `[dspace]/bin/dspace checker -a` Checker will only check bitstreams within the Community, Community or the item itself. Example: `[dspace]/bin/dspace checker -a 123456/999` Checker will only check this handle. If it is a Collection or Community, it will run through the entire Collection or Community.
- **Looping mode:** `[dspace]/bin/dspace checker -l` or `[dspace]/bin/dspace checker -L` There are two modes. The lowercase 'el' (-l) specifies to check every bitstream in the repository once. This is recommended for smaller repositories who are able to loop through all their content in just a few hours maximum. An uppercase 'L' (-L) specifies to continuously loops through the repository. This is not recommended for most repository systems. **Cron Jobs**. For large repositories that cannot be completely checked in a couple of hours, we recommend the -d option in cron.
- **Pruning mode:** `[dspace]/bin/dspace checker -p` The Checksum Checker will store the result of every check in the checksum_history table. By default, successful checksum matches that are eight weeks old or older will be deleted when the -p option is used. (Unsuccessful ones will be retained indefinitely). Without this option, the retention settings are ignored and the database table may grow rather large!

Checker Results Pruning

As stated above in "Pruning mode", the checksum_history table can get rather large, and that running the checker with the -p assists in the size of the checksum_history being kept manageable. The amount of time for which results are retained in the checksum_history table can be modified by one of two methods:

1. Editing the retention policies in `[dspace]/config/dspace.cfg` See Chapter 5 Configuration for the property keys. OR
2. Pass in a properties file containing retention policies when using the -p option.To do this, create a file with the following two property keys:

```
checker.retention.default = 10y
checker.retention.CHECKSUM_MATCH = 8w
```

You can use the table above for your time units. At the command line: `[dspace]/bin/dspace checker -p retention_file_name` <ENTER>

Checker Reporting

Checksum Checker uses log4j to report its results. By default it will report to a log called `[dspace]/log/checker.log`, and it will report only on bitstreams for which the newly calculated checksum does not match the stored checksum. To report on all bitstreams checked regardless of outcome, use the *-v* (verbose) command line option:

`[dspace]/bin/dspace checker -l -v` (This will loop through the repository once and report in detail about every bitstream checked.

To change the location of the log, or to modify the prefix used on each line of output, edit the `[dspace]/config/templates/log4j.properties` file and run `[dspace]/bin/install_configs`.

Cron or Automatic Execution of Checksum Checker

You should schedule the Checksum Checker to run automatically, based on how frequently you backup your DSpace instance (and how long you keep those backups). The size of your repository is also a factor. For very large repositories, you may need to schedule it to run for an hour (e.g. `-d 1h` option) each evening to ensure it makes it through your entire repository within a week or so. Smaller repositories can likely get by with just running it weekly.

Unix, Linux, or MAC OS. You can schedule it by adding a cron entry similar to the following to the crontab for the user who installed DSpace:

```
0 4 * * 0 [dspace]/bin/dspace checker -d2h -p
```

The above cron entry would schedule the checker to run the checker every Sunday at 400 (4:00 a.m.) for 2 hours. It also specifies to 'prune' the database based on the retention settings in *dspace.cfg*.

Windows OS. You will be unable to use the checker shell script. Instead, you should use Windows Schedule Tasks to schedule the following command to run at the appropriate times:

```
[dspace]/bin/dspace checker -d2h -p
```

(This command should appear on a single line).

Automated Checksum Checkers' Results

Optionally, you may choose to receive automated emails listing the Checksum Checkers' results to the email address specified in the `mail.admin` configu ration property. Schedule it to run **after** the Checksum Checker has completed its processing (otherwise the email may not contain all the results). As of DSpace 4.1, an email is only generated if the selected report contains at least one bitstream needing attention.

Command used:	`[dspace]/bin/dspace checker-emailer`
Java class:	org.dspace.checker.DailyReportEmailer
Arguments short and (long) forms):	Description
`-a` or `--All`	Send all the results (everything specified below)
`-d` or `--Deleted`	Send E-mail report for all bitstreams set as deleted for today.
`-m` or `--Missing`	Send E-mail report for all bitstreams not found in assetstore for today.
`-c` or `--Changed`	Send E-mail report for all bitstreams where checksum has been changed for today.
`-u` or `--Unchanged`	Send the Unchecked bitstream report.
`-n` or `--Not Processed`	Send E-mail report for all bitstreams set to longer be processed for today.
`-h` or `--help`	Help

You can also combine options (e.g. `-m -c`) for combined reports.

Cron. Follow the same steps above as you would running checker in cron. Change the time but match the regularity. Remember to schedule this **after** Checksum Checker has run. For an example cron setup, see Scheduled Tasks via Cron.

Database Query

A query like the following can be used to check the results of the checker (Postgres):

```
SELECT *
FROM checksum_history
WHERE date_trunc('day', process_start_date) = CURRENT_DATE
AND result != 'CHECKSUM_MATCH'
AND result != 'BITSTREAM_MARKED_DELETED';
```

Example of a more detailed query:

```
SELECT
    ch.process_start_date,
    ch.process_end_date,
    ch.result,
    ch.checksum_expected,
    ch.checksum_calculated,
    b.bitstream_id,
    bfr.short_description,
    b.store_number,
    substring(b.internal_id for 2) || '/' || substring(b.internal_id from 3 for 2) || '/' || substring(b.
internal_id from 5 for 2) || '/' || b.internal_id AS bitstream_path,
    hi.handle AS item_handle,
    hc.handle AS collection_handle
FROM checksum_history ch
JOIN bitstream b
ON ch.bitstream_id = b.uuid
JOIN bitstreamformatregistry bfr
ON b.bitstream_format_id = bfr.bitstream_format_id
LEFT JOIN bundle2bitstream bb
ON b.uuid = bb.bitstream_id
LEFT JOIN item2bundle ib
ON bb.bundle_id = ib.bundle_id
LEFT JOIN item i
ON ib.item_id = i.uuid
LEFT JOIN handle hi
ON i.uuid = hi.resource_id
AND hi.resource_type_id = 2
LEFT JOIN handle hc
ON i.owning_collection = hc.resource_id
AND hc.resource_type_id = 3
WHERE ch.result != 'CHECKSUM_MATCH'
AND date_trunc('day', process_start_date) = CURRENT_DATE
ORDER BY ch.check_id DESC;
```