**PROCESS USED TO REPLACE BLESSFILES**

*Replacement of all blessfiles that were created before September 14 2014.*

**Description:**

The code that generates the blessfiles (Split.pl) changed when we added precision to the calculation of the channel, trigger and error values. The cosmic upload module uses the blessfiles to determine whether the split file should be blessed or unblessed. Therefore these files needed to be replaced with the correct calculation for those values/fields. The following diagram shows the cosmic upload process. The db (database) is the VDS used by the e-Labs to store the metadata associated with all the files used by the e-Labs (e.g., data, plots, posters, etc.). Legacy data collected before the Blessing functionality was added was grandfathered in by running Autobless.

|  |
| --- |
|  |

**Summary:**

* **Step 1: Recreate and replace the actual files**
  + Work in an isolated area in the server where we could run the split process for all the affected source files and get the new blessfiles with the corrected precision.
  + Run validation of the newly created blessfiles to make sure they are correct replacements.
  + Replace the actual files.
* **Step 2:** **Re-run the autoBless program**
  + Determine the new status of legacy data based on the new blessfiles.
  + Apply changes to database.
* **Step 3:** **Re-run the blessing routine**
  + Determine the new status of data uploaded using the benchmark module.
  + Apply changes to database.

**Details:**

**STEP 1: Recreate and replace the actual files**

1. We ran queries from the postgres vds database to retrieve the following:
   1. All blessfiles that needed to be recreated and their source files. We found a total of **17712** files.
   2. The list of source files that we needed to pass to Split.pl.
   3. A list of the source files in the order they were uploaded.
2. We set a working/scratch area in data2.i2u2.org to work.
3. We copied all the zipped raw files from data4.i2u2.org.
4. We unzipped all the raw source files.
5. We copied all the original \*.raw.meta files corresponding to these source files and renamed them as \*.raw.meta.old.
6. We wrote a Python program that:
   1. Read the input file with the ordered source files.
   2. Launched Split.pl with each source file and no special option.
7. Results from the **first run** of this program:

Split.pl’s output:

* 1. Split files (which we discarded).
  2. Blessfiles (which we needed to analyze if they were good replacements).
  3. \*.raw.meta file (which we compared to the \*.raw.meta.old for relevant fields).

1. We ran a series of analysis after the first run:
   1. For each source file, we compared the new meta file to the old meta file.
      1. These are the relevant fields we looked at and the decisions we made:
         1. blessfile (name): if the blessfile name differed by the index we needed to create dummies and rerun Split.pl with the source file where the dummies were present so the code would create the correct blessfile name. The tags for these files were either ‘Combo:4’ or Combo:5’.
         2. startdate: if this field was different, then we flagged this file as having a problem with a tag called ‘Combo:1’.
         3. enddate: if this field was different, we flagged it with a tag called ‘Combo:2’.
         4. chan1, chan2, chan3, chan4 and triggers: if any of these fields was different, then we tagged the file with ‘Combo:3’.
         5. We tagged ‘Combo:0’ all those meta files that differed in the resulting number of splits.
         6. If there were problems in several fields, we combined the tags. For example: the startdate and enddate were different and the tag was ‘Combo:12’.
   2. After comparing the meta files, we also compared the new blessfile to the old blessfile.
      1. Rules:
         1. The number of lines had to match in both.
         2. The first column had to have exactly the same values.
         3. The columns that now have precision had to have an absolute difference that was less than 1.
2. Results after running after **26 more runs** of Split.pl with option 2 (no need to create the split file, we supplied the dummies so Split.pl would name the blessfile appropriately):
   1. **14020** files passed all the tests.
      1. We prepared a test by selecting and moving a few files.
      2. We took a screenshot of the before and after moving the new files and we agreed that these files were good replacements.
      3. We moved the new files to production.
      4. We tagged these files in the database with a new field in the metadata: **blessfilecomment**.
   2. **83** files could not be recreated because the source file did not exist in our server.
   3. **224** files were not created because the new version of Split.pl did not create them from the corresponding source file (the older version of Split.pl did).
   4. The remaining **3385** files failed some of the tests and we needed to examine where they failed and if there was something that could be done to fix/recreate these files.
3. We started working with the Combo tags (totals at the end):
   1. **‘Combo:0’**: This tag meant that the resulting number of files in the new meta file differed from the number of files in the old meta file. We parsed and compared them in more detail:
      1. Some of the files could be compared and they went through all the tests that we ran before (by combing them out and comparing them).
      2. Some of the files in the old meta had the same exact root and different indexes. This did not happen in the new meta files. These files could not be compared.
      3. Some of the files existing in the old meta were never created by the new version of Split.pl.
   2. **‘Combo:1’**: This meant that the new and old meta file description differed only on the ‘startdate’. We agreed that if this was the only test not met then the resulting new file was a good replacement.
   3. **‘Combo:2’**: This meant that the new and old meta file description differed only on the ‘enddate’. We agreed that if this was the only test not met then the resulting new file was a good replacement.
   4. **‘Combo:3’**: This meant that the new and old meta file description differed only on chan1 and/or chan2 and/or chan3 and/or chan4 and/or triggers. We agreed that if these were the only tests not met and the file was not a ‘benchmark’ file then the resulting new file was a good replacement.
   5. **‘Combo:4’** or **‘Combo:5’:** This was an index problem and this could be easily fixed by changing the name of the file provided it passed all the other tests.
4. We also analyzed the files that failed when comparing the actual new blessfile to the old blessfile and we decided the following:
   1. If the files differed in the line count and the difference was just **1 more line,** then the new blessfiles were good replacements.
5. For every new group of files that were good replacements, we chose a sample and took screenshots of the before and after before actually moving them to production and then we added a new field to the metadata to indicate when the blessfile was replaced and with which group to make it easier to track.
6. Final totals:
   1. **14020** files passed all the tests.
   2. **2025** extra files were moved because of analyzing combos and creating new rules. Total moved files so far: **16045**.
   3. **83** files could not be recreated because the source file did not exist in our server.
   4. **224** files were not created because the new version of Split.pl did not create them from the corresponding source file (the older version of Split.pl did).
   5. **102** files were deleted by the user since we started the process and did not need to be replaced.
   6. **349** files were never created by the new Split.pl.
   7. **223** files have the same root and multiple indexes within the same split.
   8. The remaining **686** files failed the actual blessfile comparison. We parsed the newly created blessfile and the existing blessfile and a ran two different types of comparisons:
      1. We counted lines in both and did not accept files that different by count >= 2.
      2. The first column (time) had to be an exact match otherwise the file was not accepted.
      3. For the columns with increased precision we only accepted those that differed by an absolute value < 1.
7. All the files that were not replaced have a tag in the metadata indicating that we were not able to recreate them.

**STEP 2: Re-run the autoBless program**

At the time of the implementation of the Benchmark module, we ran a program called autoBless.pl to determine the blessing status for the already existing files by analyzing the actual blessfile.

1 – We used the same input list that was created in February 2014 (same files).

2 – Obtained a new report from the autoBless program.

3 – Compared the results of the new report to the results of the original report and determined how many files changed their status from blessed to unblessed and from unblessed to blessed.

4 – We prepared update queries for these files in the database with two checks:

* The legacy files have NOT been chosen as benchmarks
* The legacy files have NOT been re-blessed by the user using a benchmark.

5 – Totals:

* A total of 1521 files were updated:
  + 484 changed from unblessed to blessed
  + 1037 changed from blessed to unblessed
* A total of 10654 files remained the same:
  + 7283 remained blessed
  + 3371 remained unblessed
* Files that could not be updated:
  + 7 are now benchmark files
  + 417 files have gone through the blessing routine and have a benchmark reference.

**STEP 3: Re-run the blessing routine**

These were files that have gone through the blessing routine at the time of split. This module determines whether the file is blessed or unblessed by comparing it to the user selected benchmark.

1 – We wrote a java program that would run the blessing routine outside the cosmic upload module.

2 – The input to this program was the output file of a query to the database in which we retrieved all those files that had a benchmark reference plus all the necessary database fields the program needs to determine whether the file should be blessed or unblessed.

3 – The output of running this program was a report with the new status of these files plus the error message if the check resulted with an unblessed status.

4 – We analyzed this report to find how many files changed their status.

5 – We prepared update queries to the database for only those files with different status.

6 – Totals:

* 3450 files were analyzed
* 2546 files stayed the same
* 904 changed their status
  + 469 changed from true to false
  + 435 changed from false to true