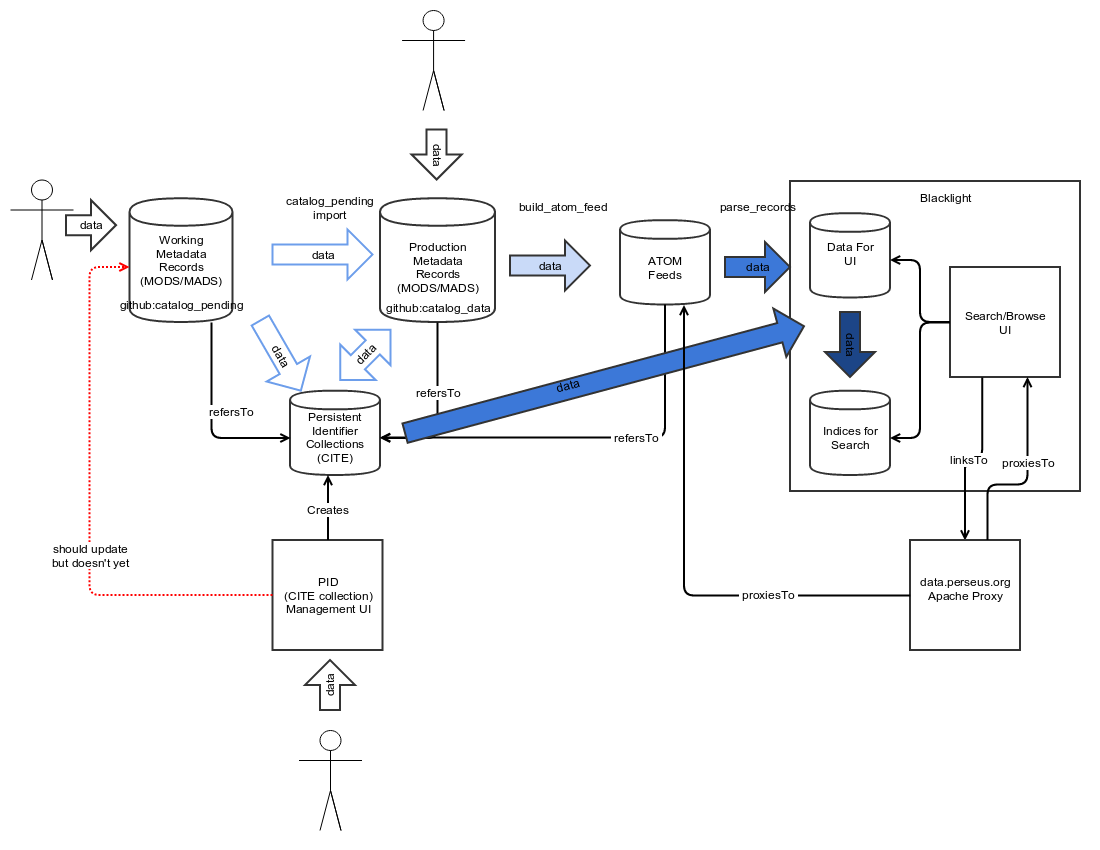
Architecture Overview

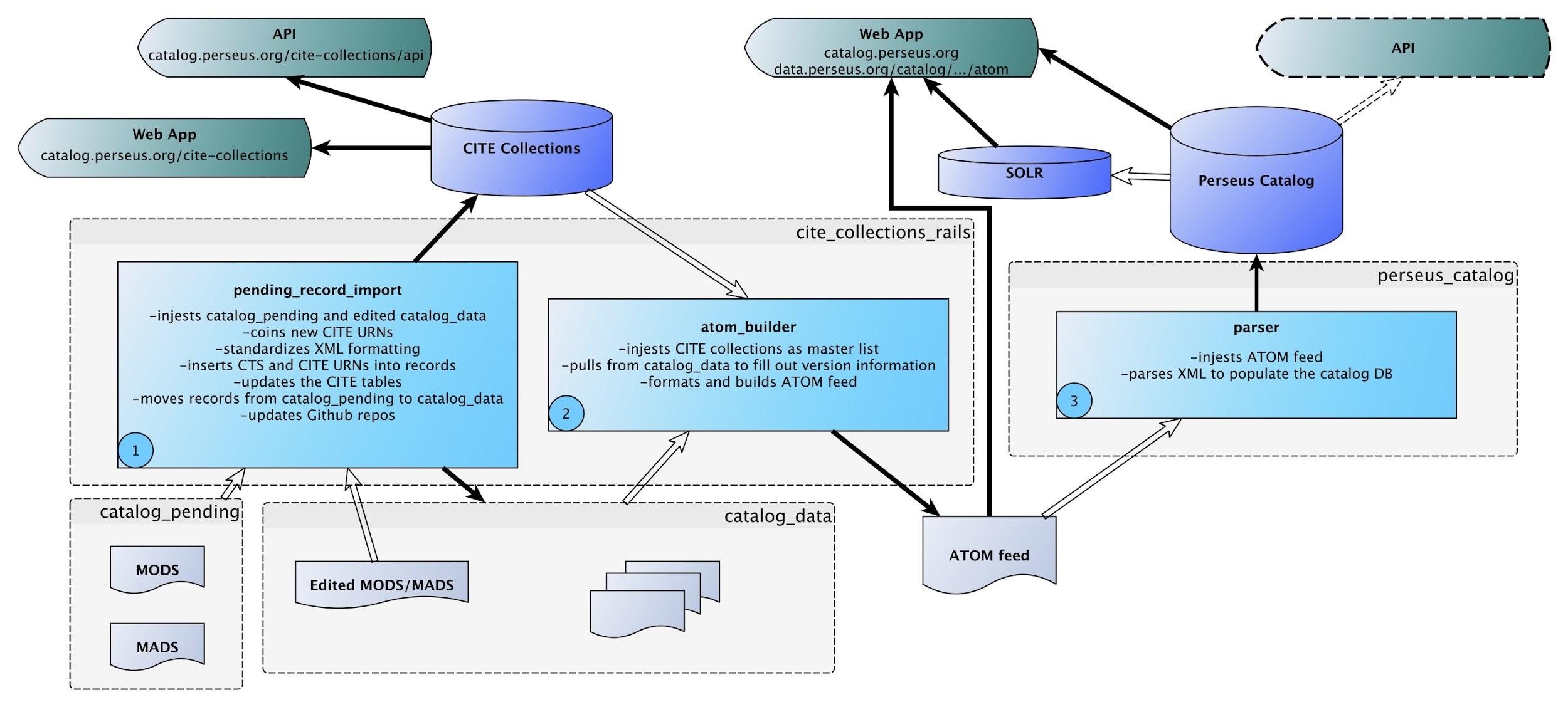
For additional details and history, please see:

* Catalog User Documentation: <http://sites.tufts.edu/perseuscatalog/documentation>
* Cataloging Process : http://catalog.perseus.tufts.edu/PerseusCatalogDataCreation.html



Perseus Catalog Update Process

The following image is an illustration of how the update works. Inputs are marked with white arrows, outputs with black arrows. The larger, dotted border boxes represent github repos where the data or code are located.



This document is going to delve pretty deeply into the steps that the software takes, it can be read as documentation of the software itself and supplements the comments in the code. Various levels of detail are linked via formatting and correspond to the above illustration, read as deep as you’d like.

Pseudo scripts covering these steps, and meant to be a first step towards a fully automated continuous integration environment for the catalog are at:

<https://github.com/PerseusDL/perseus_catalog/blob/master/catalog-update.setup.sh>

<https://github.com/PerseusDL/perseus_catalog/blob/master/catalog-update.sh>

0 -- Catalog records are added to catalog\_pending and/or records in catalog\_data are edited.

1 -- A catalog pending import process is triggered (either manually or automatically, ideally once a week).

-Process catalog\_pending\_import:

-(rake catalog\_pending\_import)

-Update from catalog\_data

-finds recently changed files in catalog\_data

-for each recently changed file:

-parses xml to find ids, particularly the ctsurn

-if MADS file, update author row if any changes to authority\_name, canonical\_id, alt\_ids, or related\_works

-if MODS file

-update textgroup row if any changes to groupname\_eng

-update work row if any changes to title\_eng or orig\_lang

-update version row if any changes to label\_eng or desc\_eng

-MADS importer

-parses the catalog\_pending MADS xml to pull out important parts, any ids, the author name, etc.

-searches the CITE tables for authors that match the information

-if a row for the author already exists in the authors CITE:

-attempt to make sure the canonical\_id is correct, add any new alternate ids, check the name and related\_works

-if it doesn’t exist

-add a CITE row for the author

-add the newly created CITE id for the author to the MADS file

-generate the CITE id (increments from last id in the table)

-move the MADS file to catalog\_data

-MODS importer

-adds “mods” namespace to the record if it doesn’t already have it

-parses catalog\_pending MODS xml to find ids, author, title, etc.

-sees if the record already has a ctsurn

-if record has ctsurn:

-search the version CITE table for an existing row

-if the row exists, it isn’t invalid, and matches the current ctsurn perfectly

-find the CITE work

-if “has\_mods” true (assumed cite row or correction)

-update label and description

-move file to catalog\_data

-if “has\_mods” false

-update “has\_mods” to true, update label and description

-move file to catalog\_data

-if there is a ctsurn but no version CITE row

-check for constituents within the record

-split into different files and go through the steps again for each

-if no constituents

-check that there are textgroup and work CITE rows, add them if not

-generate the CITE id (increments from last id in the table)

-add new version to version CITE table

-move file to catalog\_data

-if record does not have a ctsurn:

-check for constituents within the record

-split into different files and go through the steps again for each

-if no constituents

-check that there are textgroup and work CITE rows, add them if not

-generate the CITE id (increments from last id in the table)

-add new version to version CITE table

-move file to catalog\_data

-delete any now empty directories in catalog\_pending

-catalog\_pending\_import finishes, any files that hit errors remain in catalog\_pending and noted in the error log (products of this process are new files in catalog\_data and new rows in the CITE tables [and therefore newly generated CITE urns])

1a -- Both catalog\_pending and catalog\_data need their changes to be committed and pushed to Github.

2 -- Atom feed creation is triggered.

-Process build\_atom\_feed (types possible, all or latest):

(rake build\_atom\_feed type=[all | latest])

-if type = all, dumps all rows in the works CITE table and works through that list

-if type = latest, pulls files in catalog\_data that have been changed recently (set to files less than a week old) and uses that to pull a works list from the CITE table

-for each row in the work list that is “published”

-parse the xml to find the needed ids, language info, names, etc.

-set up the FRBR.feeds directory structure

-create the textgroup directory and feed file, if they don’t already exist, and open the file

-the head of all the atom file levels (textgroup, work, and version) and setup for the text inventory are all the same, initial feed creation populates the header and starts the text inventory by adding the textgroup

-add a node for the current work to the opened textgroup feed’s text inventory

-create the work feed file and open it

-add node for the work into the work feed’s text inventory

-find all potential valid author rows in the CITE tables and save for later

-pull all MODS files for the current work and for each:

-find ids

-find version CITE row

-create version feed and directory

-add work node to text inventory in version feed

-add the version node to the text inventory of all three feed files

-build header for MODS file

-add MODS file to all three feed files

-add MADS (if there is one) to the feed files

-save the feed files

-build\_atom\_feed finishes, it produces a directory named FRBR.feed.[date] in the directory where cite\_collection\_rails is located, can be relocated if desired

3 -- Catalog import proper is triggered

-Process parse\_records:

-(rake parse\_records file\_type=’atom’ rec\_file=’[atom feed file path]’)

-import textgroups

-get all rows in the textgroup CITE table

-for each textgroup:

-if no textgroup exists in the catalog database, create a new row and populate information from the CITE row

-if there is an existing textgroup row, check that the urn and group\_name are the same, replace if not

-save the row

-import authors

-get all rows in author CITE table

-for each author:

-get the MADS record

-find the corresponding textgroup CITE row, based on the canonical\_id

-if there is a CITE textgroup, the CITE textgroup has\_mads=true, and there is an author

-search the catalog database for the CITE author, first by ids, secondly by authority\_name

-if there is no author in the catalog database, create an author row from the CITE author

-if there is an author from the catalog database, check the unique\_id (the CITE URN), ids, and name, change to the CITE values if they don’t match

-move related\_works to the catalog db author

-parse the MADS record to pull out information for the fields not saved in the CITE author table

-save the catalog db author

-import the atom feeds, work level

-use the atom id to extract the various CTS ids (work, textgroup, author) and save them separately for ease of using them later

-pull from the CITE tables the rows for the current work, textgroup, and author

-search the catalog db for all potential authors

-check the catalog db for the current work

-if no work row, create a new work row

-use the CITE work to fill in/double check the catalog db work’s fields

-if there are no potential authors, create a stub author using the CITE textgroup

-if there are authors, choose the correct author by looking for the work in the related\_works

-associate the work, textgroup, and author together in a separate table

-find all the MODS records in the work level atom file and for each:

-get the CITE version row (errors out if there is no CITE row)

-check if the version exists in the catalog db row using the cts\_urn provided by the CITE version

-parse the MODS record or pull from the CITE version to fill out the catalog db fields

4 -- Solr import is triggered

-curl http://localhost:8983/solr/db/dataimport?command=full-import&clean=false

5 -- Server is restarted

-This might not be strictly necessary if it is just a data import, however if any code is added or debugged a restart is needed for the new code to be used

6 -- Atom Feeds are deployed

-This is manual process at the moment, an automated method needs to be developed, the current process is:

-ssh [username]@pollux.perseus.tufts.edu

-ssh [username]@10.246.176.215 (ip for static0.perseus.tufts.edu -- dns doesn’t work)

-cd /perseus/servers/static/catalog

-rsync -av -catalog1.perseus.tufts.edu:/var/www/cite\_tables/FRBR.feeds.YYYYMMDD/\*

-N.B. concerning the atom feed path, at the moment the feeds will be generated and saved in the same base directory as cite\_collections\_rails (/var/www), the location of the directory is liable to change