code{4}lib JOURNAL

Issue 52, 2021-09-22

Building and Maintaining Metadata Aggregation Workflows Using Apache Airflow

PA Digital is a Pennsylvania network that serves as the state's service hub for the Digital Public Library of America (DPLA). The group developed a homegrown aggregation system in 2014, used to harvest digital collection records from contributing institutions, validate and transform their metadata, and deliver aggregated records to the DPLA. Since our initial launch, PA Digital has expanded significantly, harvesting from an increasing number of contributors with a variety of repository systems. With each new system, our highly customized aggregator software became more complex and difficult to maintain. By 2018, PA Digital staff had determined that a new solution was needed. From 2019 to 2021, a cross-functional team implemented a more flexible and scalable approach to metadata aggregation for PA Digital, using Apache Airflow for workflow management and Solr/Blacklight for internal metadata review. In this article, we will outline how we use this group of applications and the new workflows adopted, which afford our metadata specialists more autonomy to contribute directly to the ongoing development of the aggregator. We will discuss how this work fits into our broader sustainability planning as a network and how the team leveraged shared expertise to build a more stable approach to maintenance.

By Leanne Finnigan and Emily Toner

Acknowledgements

While acknowledgements often appear as a footnote, the work discussed in this article was highly collaborative, and we as co-authors would not have this infrastructure to describe without the invaluable contributions of coworkers, past and present, at Temple University Libraries and PA Digital. We feel it is imperative to begin first by elevating and acknowledging their labor. This includes their involvement in the multiple phases of project planning and management, the design and implementation of technical infrastructure, the iterative development of automated workflows and validations/transformations, and ongoing maintenance.

Between 2018 and 2021, the following people contributed directly to this work: Jennifer Anton, Rachel Appel, Timothy Bieniosek, Leanne Finnigan, Christina Harlow, David Kinzer, Chad Nelson, Steven Ng, Stefanie Ramsay, Holly Tomren, and Emily Toner.

Introduction

Launched in 2016, PA Digital is a network that serves as Pennsylvania's service hub for the Digital Public Library of America (DPLA). Over the last seven years, the development and operations of PA Digital have been funded through Library Services and Technology Act (LSTA) grants from the Institute of Museum and Library Services as administered by the Pennsylvania Department of Education through the Office of Commonwealth Libraries and the Commonwealth of Pennsylvania. With these grants, PA Digital contracted Temple University Libraries staff to support the initiative by providing project management, technical and metadata expertise, and digital infrastructure.

One of PA Digital's primary services is harvesting and aggregating metadata records from the digital collections of contributing institutions throughout Pennsylvania, including libraries, historical societies, museums, and other regional partners. As part of this work, we offer ongoing metadata support, both transforming harvested records for greater consistency according to our metadata guidelines and thoroughly reviewing metadata to provide feedback and guidance to our contributors. Once processed and finalized, aggregated records are then delivered to the DPLA on a quarterly schedule.

When the project first began, the PA Digital team developed a prototype to support OAI-PMH aggregation workflows using a customized Hydra-based solution (DPLAH). From there, PA Digital expanded rapidly. Our total number of contributors increased from 19 at its launch to 84 by 2019 and, over time, required processes for a growing range of digital repositories, including bepress, CONTENTdm, Islandora, Omeka, and Samvera. The team also needed the ability to import CSV files for contributors with technical constraints. Modifications to these workflows and general troubleshooting proved labor-intensive because of inconsistent legacy code. Metadata specialists, working to provide metadata support to our contributors, had little to no flexibility over the processing of incoming metadata and generally relied on the team's developers to make any local adjustments.

Assessing our needs through the creation of user personas and scenarios, the PA Digital team began to plan in 2018-2019 for the implementation of a new aggregation solution. The goals were to build a more flexible metadata aggregator that would afford more autonomy to our metadata specialists and reduce the developer support required to maintain the system. Evaluating our options, we looked at existing DPLA aggregation applications, including Combine from Wayne State University (part of the Michigan DPLA Service Hub). Through extensive trial-and-error, the PA Digital team determined that the best direction for our aggregation workflows was to leverage our existing systems and expertise rather than adopting something new, knowing that doing so would likely lead to many of the same maintenance issues as before. In late 2019, Christina Harlow proposed the idea to develop new processes in Apache Airflow, a robust application that Temple University Libraries already used for other projects. The team decided to pursue this plan, which developed iteratively from 2019 to 2021.

Apache Airflow is an open-source workflow management tool that can be used to manage and monitor pipelines. Timothy Bieniosek and Chad Nelson at Temple University Libraries first introduced Airflow into its technology portfolio for smaller projects, including the aggregation of small usage datasets. In 2018,

our use of Airflow expanded to include processes for harvesting and indexing Alma catalog records into our custom-built discovery platform, Library Search, and other automated workflows for our library's website in 2019.

Below we will outline how the PA Digital team adopted Airflow for metadata aggregation and how this new approach supports related metadata reviews, transformations, and validations.

PA Digital aggregation workflows in Airflow

One of the core concepts in Airflow is a Directed Acyclic Graph, or DAG. In Airflow's architecture, a DAG defines a group of tasks along with their relationships and dependencies, which can make up a data pipeline. DAGs are built with Python, with the files loaded and then run via Airflow. A more extensive overview of Airflow DAGs can be found in Apache's documentation of its architecture and workloads.

These DAGs can be managed and monitored through Airflow's user interface. Each instance of an executed DAG is known as a DAG run.

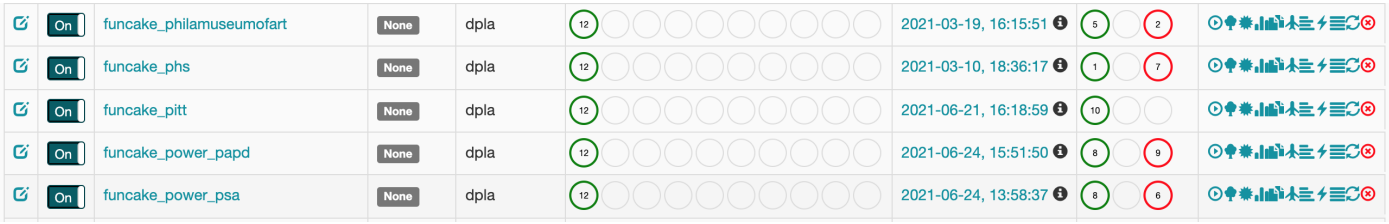

Figure 1. Snapshot of Airflow dashboard, with individual PA Digital DAGs.

In order to manage the variety of external data sources aggregated, the PA Digital team developed a template to generate a DAG for each contributing institution. The tasks within each DAG and the functions they execute are largely standardized based on this template. In order to tailor these workflows individually, the project team utilizes the variables configuration in Airflow, matching each DAG with a set of custom values specific to the contributing institution. These variables are stored in a JSON file and regularly updated within Airflow.

Each PA Digital DAG executes a workflow that harvests, validates, transforms, and publishes metadata into Solr. This process connects to Temple University Libraries' instance of Apache SolrCloud, a distributed approach to Solr indexing.

The set of tasks for each PA Digital DAG include:

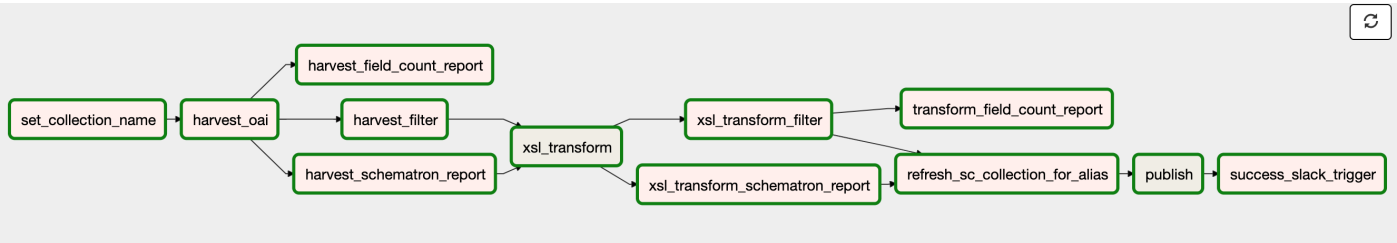

Figure 2. Graph view of a PA Digital DAG in Airflow.

- **Set collection name** (*set_collection_name*): This first task establishes the name for a SolrCloud collection based on the DAG identifier. At the end of this DAG, the output will include this SolrCloud collection. This label is also applied to related files, directories, and logs for each DAG run.
- **Harvest OAI** (*harvest_oai*) *or* **Harvest CSV** (*harvest_aggregator_data*)**:** By default, the PA Digital DAG template uses an OAI-PMH harvester task. For each DAG, we supply OAI-PMH parameters as variables, including the URL for the specific OAI-PMH endpoint, the preferred metadata prefix, and the collection setSpec identifier(s), which are used in making requests. Individual DAGs can also be tagged for the alternative harvester task, which imports CSV-formatted data from a private Github repository instead. In all cases, the records are sent to our cloud storage once harvested.
- Validation tasks for the harvested records, including:
  - **Harvest filter** (*harvest_filter*): This post-harvest task filters out any invalid records based on a Schematron file defined in the variables and generates a CSV file with the list of filtered records.
  - **Harvest schematron report** (*harvest_schematron_report*): This post-harvest task generates an additional CSV report, based on a Schematron file defined in the variables.
  - **Harvest field count report** (*harvest_field_count_report*): This post-harvest task generates a report that includes a count of each metadata field and a percentage of records with that field, in order to assess the completeness of the records. This is adapted from a command-line tool developed by Mark Phillips (Phillips 2013).
- **XSL transform** (*xsl_transform*): Using a Saxon XSLT engine, this task transforms the processed records with an XSLT file defined in the variables. The records are sent to our cloud storage once transformed.
- Validation tasks for transformed records, including:
  - **XSL transform filter** (*xsl_transform_filter*): This post-transform task filters out any invalid records based on a Schematron file defined in the variables and generates a CSV file with the list of filtered records.
  - **XSL transform schematron report** (*xsl_transform_schematron_report*)**:** This post-transform task generates an additional CSV report, based on a Schematron file defined in the variables.

- **XSL transform field count report** (*xsl_transform_field_count_report*)**:** This post-transform task generates a report that includes a count of each metadata field present across records and a percentage of records with that field, in order to assess the completeness of the records. This works the same way as the harvest field count report task above.

- **Refresh SolrCloud collection for alias** (*refresh_sc_collection_for_alias*)**:** After the completion of this task, the transformed metadata is indexed in a SolrCloud collection. This step deletes any existing SolrCloud collection for this DAG, creates a new one, and adds it to a SolrCloud alias. An alias provides an alternative way for the querying of one or multiple SolrCloud collections. Depending on the target alias defined in the variables, the SolrCloud collection for each DAG run will be listed under a “dev” or “prod” alias, both of which include any PA Digital collections indexed in other DAGs.
- **Publish** (*publish*): This task indexes the transformed records to the SolrCloud collection, which is included under the “dev” or “prod” alias. The raw XML for each record is stored in a single Solr field.

Here is sample of the JSON variables for an individual DAG:

```
"APS_HARVEST_CONFIG": {

"endpoint": "http://diglib.amphilsoc.org/oai2",

"md_prefix": "oai_dc",

"all_sets": false,

"excluded_sets": [],

"included_sets": [

"graphics_Mss.B.P.31.15d",

"graphics_Mss.B.P165",

"graphics_Mss.Ms.Coll.76.17",

"text_Mss.365.P381p",

"text_Mss.B.F826",

"islandora_graphics_collection" ],

"schematron_filter": "validations/ingest_oai_validation.sch",

"schematron_report": "validations/padigital_missing_thumbnailURL.sch",

"schematron_xsl_filter": "validations/padigital_reqd_fields.sch",

"schematron_xsl_report": "validations/padigital_missing_thumbnailURL.sch",

"xsl_branch": "main",

"xsl_filename": "transforms/islandora_aps.xsl",

"xsl_repository": "tulibraries/aggregator_mdx" },

"APS_TARGET_ALIAS_ENV": "dev",
```

The GitHub repository for the PA Digital Airflow DAGs and templates is https://github.com/tulibraries/funcake_dags. In many cases, the tasks utilized for PA Digital aggregation are not unique to this initiative and are used across other projects in Temple University Libraries. For example, the Harvest OAI task is also used in harvesting and indexing incremental updates of our libraries’ catalog records for discovery. This centralized code can be found at https://github.com/tulibraries/tulflow.

In Airflow, each DAG run is tracked and logged, making it easier to monitor and troubleshoot issues with these workflows and harvested metadata. Individual tasks and dependencies within a DAG run can be cleared and/or rerun as needed. For example, we can redo the XSL transform and subsequent tasks without having to reharvest from the contributing institution again. This proves extremely useful when we add SolrCloud collections to our “prod” alias, which is done by updating the variables and rerunning the “Refresh SolrCloud collection for alias” task rather than reindexing.

Blacklight and SolrCloud integrations

As described above, we use SolrCloud to index the transformed metadata and aggregate SolrCloud collections together through the use of aliases. Once indexed, these collections can be accessed via a dev or prod OAI-PMH endpoint, depending on the target alias supplied for the Airflow DAG.

Both of these OAI-PMH endpoints are set up and configured through the OAI Provider Plugin for Blacklight applications. Blacklight is an open-source Ruby on Rails discovery framework with Solr searching, maintained through cross-institutional collaboration primarily within the libraries/archives/museums profession. Like Airflow, Temple University Libraries uses this open source project in other contexts, most notably for our main discovery platform Library Search. The PA Digital team had prior experience configuring these applications and therefore could easily incorporate this approach into our broader infrastructure and maintenance support.

In addition to the validations and reporting built into our Airflow tasks, we needed to set up a method for the manual review of transformed records, an important service as part of broader metadata support provided to our contributors. With established expertise using the framework, the PA Digital team created Funnel Cake (named in honor of the Pennsylvania Dutch delicacy), a lightweight Blacklight site used to search and preview PA Digital aggregated metadata internally. The GitHub repository for Funnel Cake is https://github.com/tulibraries/funnel_cake. The PA Digital team also developed a simple DAG to index the metadata in the dev alias into a single SolrCloud collection of aggregated metadata. This enables the querying, filtering, and displaying of qualified Dublin Core fields from the aggregated collections in Funnel Cake.

So through Funnel Cake, two sets of SolrCloud collections can be accessed: a SolrCloud collection via the Blacklight user interface itself and sets of SolrCloud collections through the Blacklight OAI-PMH plugin. Once all the relevant collections are added in the prod alias and made available through the production OAI-PMH feed, DPLA harvests the PA Digital aggregated data.

Metadata transformations and validations, including git integration

Several transformation and validation scripts are used throughout the tasks above. Prior to the implementation of this new infrastructure, metadata processing was hard-coded into our aggregator software, and update permissions were limited only to the development team. During onboarding of new contributing institutions and related metadata review, metadata specialists on the PA Digital team were often faced with a choice when metadata did not meet internal requirements (records lacking a title, rights statement, etc.), either:

1. Ask the developers to update the code to make the metadata passable ("Some of these records are missing rights statements, and the institution, run by volunteers, cannot add them locally. Can you update the code so we can add a provided statement to all of the incoming records?")
2. Ask the contributing institution to make changes locally ("The relation field is filled with html tags and broken links. Can you unmap it from your OAI-PMH feed?")
3. Or, if neither were an option, decide that a collection was too cumbersome to include.

Eventually, all of the piecemeal code modifications made our original aggregator software too difficult to maintain. Despite our best efforts and ongoing testing, an update here would affect a previous update there. The software became a house of cards and felt increasingly fragile.

We saw in Airflow a more sustainable path forward that better integrated metadata specialists into the development of aggregation workflows and provided a way for them to maintain and incorporate transformations themselves without affecting the rest of the infrastructure.

To do this, we opted to maintain a separate GitHub repository, aggregator_mdx, where metadata specialists had full reign over the creation, testing, and maintenance of transformation and validation scripts. Once added to GitHub, the paths to these files are defined as Airflow variables and incorporated into individual DAG workflows as described above.

The repository contains four types of XML files: Schematron scripts for validations, XSLT scripts for metadata transformations, and sample XML metadata and Xspec scripts for unit testing that support the integrity of the repository in a continuous integration workflow. Christina Harlow was instrumental in building this careful infrastructure and providing initial training. Prior to this project, our metadata specialists had varying degrees of comfort with XSLT and little, if any, direct experience with GitHub and CI/CD workflows. Through ongoing applied learning, they now actively contribute to the repository's development. These scripts, and their use for various tasks in Airflow, are described below. The repository's documentation provides more detail on their implementation and use, including required dependencies and folder structure.

Validations

The following Airflow tasks use Schematron files for metadata validation: *harvest_filter*, *harvest_schematron_report*, *xsl_transform_filter*, and *xsl_transform_schematron_report*. Schematron uses XPath to test for different elements and patterns within the metadata. The files we use can be found in the validations folder on GitHub.

The *harvest_filter* and *harvest_schematron_report* tasks run concurrently on harvested metadata before they are transformed. For the *harvest_filter* task, the Schematron files check for the following:

- The presence of a title element in any namespace
- The absence of a stopword ('pdcp_noharvest') anywhere in the metadata – Our old aggregator checked for a stopword, which was added by the data provider to prevent the ingestion of records they wanted us to skip. This was replicated in the harvest filter to accommodate institutions who used that workaround.
- The absence of the value 'Collection' in a type field – We do not accept collection-level metadata or finding aids, and we found that filtering records with a 'Collection' type reduced the likelihood that these would be harvested.

Any invalid records as defined by the script called by this *harvest_filter* task *do not* proceed to the next task. Along with a human-readable error note such as, "There must be a title," invalid records are listed in a CSV that posts to our cloud storage where it can easily be passed on to data providers for review and remediation.

The Schematron scripts called by the *harvest_schematron_report* task behave similarly, except invalid records *do* continue on to the next task. This allows us to create a list of records that match certain XPath selections for assessment purposes. For example, if an institution wants to know which of their records lack a thumbnail URL, we can code that into the Schematron file and provide them with the resulting CSV report without preventing records from moving to the next task.

The *xsl_transform_filter* and *xsl_transform_schematron_report* tasks run concurrently on metadata records after they are transformed and allow us to assess whether the transforms behave as expected. The Schematron script associated with the *xsl_transform_filter* task, similar to *harvest_filter*, prevents invalid records from being passed to the next task. It checks for the following, which reflect internal and/or DPLA requirements:

- The presence of <dcterms:title> with a non-null value
- The presence of <dcterms:rights> for textual rights statements or <edm:rights> for rights URIs with non-null values
- The presence of <dcterms:isShownAt> with a valid URL to the object in the provider's repository
- The presence of <edm:dataProvider> with a non-null value
- The presence of an identifier in <dcterms:identifier>

The Schematron script associated with the *xsl_transform_schematron_report* task, similar to *harvest_schematron_report*, allows us to assess transformed metadata and create a report of validation errors while allowing such records to proceed to the next task.

Transformations

The XSLT scripts used in the *xsl_transform* task are written to process XML metadata harvested via OAI-PMH as well as static XML or CSV files harvested from our private GitHub repository. To limit the number of scripts we need to maintain and update, many are linked, making generous use of <xsl:include> to avoid duplicate transform tasks as much as possible while allowing for specialty mappings to support specific repositories, institutions, or collections.

The three main XSLT script types are:

Base crosswalk XSLT (e.g., transforms/oai_base_crosswalk.xsl): The most general, this crosswalk type is included in all transforms in our repository that process metadata, and define only those mappings we want applied to every record processed unless defined elsewhere. Our OAI-PMH base crosswalk contains, for example, the mapping of incoming <dc:title> to PA Digital's <dcterms:title>, <dc:creator> to <dcterms:creator>, etc. It also removes extraneous whitespace and delimits on semicolons for all but the following fields where semicolons are often used as punctuation: description, title, publisher, extent, relation, and rights. Separate base crosswalks process CSV and non-OAI XML stored in PA Digital's private data repository (csv_generic_semic.xsl, which delimits on semicolons and csv_generic_pipe.xsl, which delimits on vertical bars). Base crosswalks are not modified unless a change is desired for all processed records. If an override of the base crosswalk is needed, it is done in an institution-, repository-, or collection-specific XSLT as described below. All base crosswalks import remediation XSLTs, also described below.

Institution- repository-, or collection-specific XSLT (e.g., transforms/cdm_temple.xsl): These are the files that run against harvested metadata; their paths are given as variables in Airflow. They include mappings to crosswalk metadata whose location or pattern varies depending on repository and institution. By defining substring variables of a particular metadata element, such as the unique URL to a digital object found in <dc:identifier>, we can generate other needed metadata not otherwise found in the record. These may include thumbnail preview URLs; IIIF URLs; data provider, intermediate provider, and collection names; and locally generated identifiers. As stated above, if an override of the included base crosswalk is needed, it is done here so as not to affect other records that use the same base crosswalk (see dcterms:isPartOf mapping in cdm_temple.xsl for an example of a crosswalk override; the priority attribute should be used to avoid ambiguous rule matching).

If necessary, numerous XSLT files can be included in the one run against ingested metadata to avoid duplicating transform templates. The Pennsylvania State University (Penn State), for example, contributes metadata from CONTENTdm and most of their mappings are consistent with other institutions that use that repository. However, Penn State maps an object's original creation date to <dcterms:created>. Others use this field for the date of digitization and <dc:date> for the original creation date. To accommodate Penn State's mapping without negatively affecting other records that use the shared cdm_generic.xsl transform, we added another XSLT, cdm_pennstate.xsl, that includes the repository-specific transform. Since Penn State's DAG is the only one that calls cdm_pennstate.xsl, only their records get the additional mapping defined in that transform.

Remediation-specific XSLT (transforms/remediations/*.xsl): These define lookup parameters used by templates in the above XSLT types to normalize string values against certain vocabularies and generate metadata not found natively in the incoming XML. This includes the translation of OAI-PMH setSpec values into human-readable collection names, and digital object URLs into a data provider name.

For instance, we can define the base URL of an object in the field below by selecting a substring, the pattern of which is found consistently in CONTENTdm records (substring-before(.,"cdm/ref/")):

```
<dc:identifier><a href="http://digital.libraries.psu.edu/cdm/ref/collection/wpamaps/id/3314">http://digital.libraries.psu
```

By creating a parameter that identifies the expected base URL, we are able to generate normalized data provider names and consistent identifiers while accommodating other institutions that use CONTENTdm. For example, the following element in remediations/lookup.xsl defines Penn State's expected base URL as well as attributes for their institution name and code:

```
<padig:url string="Pennsylvania State University" code="PENNSTATE">http://digital.libraries.psu.edu/</padig:url>
```

Variables defined in the repository XSLT are compared to these lookup parameters and, when there is a match, result in normalized values such as those found in the following generated elements for Penn State:

```
<edm:dataProvider>Pennsylvania State University</edm:dataProvider>
<dcterms:identifier>padig:PENNSTATE-wpamaps-3314</dcterms:identifier>
```

Testing

Several features are embedded within aggregator_mdx to support the integrity of our overall infrastructure and metadata quality. The Github repository requires that commits receive at least one review and approval, usually by other metadata specialists, before they are merged to the main branch. CircleCI enables automated unit testing within the repository itself as well as into the merge process.

Unit tests use Xspec to define expected behavior when a specific transform or validation is run against sample XML metadata. We use actual source metadata from our data providers to ensure the utility of the tests. As is seen in tests/xslt/samvera_shi.xspec, the transform is defined as an attribute within the <x:description> element. Scenarios include a human readable label describing the test, a path to the sample XML metadata to be tested (<x:context>), and expected results when the transform is applied (<x:expect>).

When this test suite is run locally via the command line, all tests in the directory run and the results are available as formatted HTML reports. When viewed in a browser, scenarios or tests that pass are highlighted in green, while those that did not are in red along with a visual demonstration of the expected vs. actual result. The scenarios, or the transform or validation scripts they test, can then be modified to produce the expected results.

There is an Xspec add-on for Oxygen that is useful to quickly run individual tests, but since a single transform often runs against multiple sample XML files, it is best to run the suite against the whole directory as described in this repository's documentation.

Migration, Maintenance, and Sustainability

When we first began using Airflow for PA Digital in 2019, there were 394,774 metadata records from 926 collections in our old aggregator. Rather than attempting an immediate migration that involved reprocessing all the metadata in Airflow at once, the team opted instead to do so iteratively over a longer period of time. This work began in December 2019 and concluded in March 2021. We grouped institutions by repository since transformations and validations could more likely be shared, and incrementally reprocessed collections from these groups as Airflow DAGs. This had the added benefit of allowing time for additional quality assessment of metadata as we went.

During this interim period, the PA Digital team also harvested and indexed static data from our old aggregator endpoint using Airflow and created Schematron files to filter records from institutions that had been reprocessed. Every time an institution was added to the Schematron filter, we watched their records disappear from the old endpoint until reprocessing was complete. As of July 2021, the PA Digital team has phased out the old aggregator software, an important and cathartic milestone in our project.

The old aggregator caused endless headaches because of its inflexibility. The new aggregation infrastructure and workflows adopted by the PA Digital team are still quite complex and require significant technical expertise. However, the knowledge and support of these systems is now much more distributed. Developers, metadata specialists, and project managers on the PA Digital team all actively participate in the development, maintenance, and monitoring of the aggregation workflows at different levels. With Airflow's flexibility, the team can also modify and adapt our processes more easily. Likewise, with the use of transformation and validation scripts in GitHub, the metadata specialists on the project can now independently build and maintain customized workflows for individual contributing institutions. This allows us to provide better metadata support to our contributors and partners as well.

The set of applications selected (Airflow, SolrCloud, and Blacklight) also align with what is already maintained by the development team at Temple University Libraries. This choice will help ensure continued technical support of the PA Digital workflows as long as that infrastructure is maintained at Temple.

The completion of the Airflow implementation and migration project came at a critical point for PA Digital. In 2020-21, PA Digital as a network began a planning process to shape the future vision and direction of the initiative beyond our initial grant funding. These planning efforts focus on the long-term sustainability of the initiative, both organizationally and financially. The operational changes made to PA Digital as part of the migration to Airflow align with those efforts, allowing our team at Temple to better maintain the aggregation workflows for the project, and thus improving the technical sustainability of the project for the years ahead.

References

Phillips, Mark. 2013. Metadata Analysis at the Command-Line. *Code4Lib Journal* [Internet]. [Cited 2021 Aug 12]; 19. Available from: https://journal.code4lib.org/articles/7818.

About the Authors

Leanne Finnigan is Database Management Librarian at Temple University Libraries. She was the co-project manager when PA Digital launched and now serves as its metadata coordinator.

Emily Toner is Technology Projects Librarian at Temple University Libraries and currently a co-project manager for PA Digital.