An XML-based migration from Digital Commons to Open Journal Systems

The Oregon Library Association has produced its peer-reviewed journal, the OLA Quarterly (OLAQ), since 1995, and OLAQ was published in Digital Commons beginning in 2014. When the host institution undertook to move away from Bepress, their new repository solution was no longer a good match for OLAQ. Oregon State University and University of Oregon agreed to move the journal into their joint instance of Open Journal Systems (OJS), and a small team from OSU Libraries carried out the migration project. The OSU project team declined to use PKP's existing migration plugin for a number of reasons, instead pursuing a metadata-centered migration pipeline from Digital Commons to OJS. We used custom XSLT to convert tabular data exported from Bepress into PKP's Native XML schema, which we imported using the OJS Native XML Plugin. This approach provided a high degree of control over the journal's metadata and a robust ability to test and make adjustments along the way. The article discusses the development of the transformation stylesheet, the metadata mapping and cleanup work involved, as well as advantages and limitations of using this migration strategy.

By Cara M. Key

Introduction

Content migrations are a fact of life for library technologists. For a variety of reasons, things need to get moved from old systems to new systems. In between the old and new systems is often where the interesting work takes place. In that space between, great pains may be taken to make sure that what one ends up with is essentially the same as what one started with - that everything looks at least as good and functions at least as well, and hopefully better.

It isn't just the digital assets that are relocated; whatever descriptions they possess that allow users to find and understand them must also be brought along. A PDF migrated from one repository to another may not require much special care, but differences in platform-specific metadata schemas can make it quite challenging to preserve the meaning of the accompanying description. Migration solutions that try to serve all use cases may not give sufficient attention to the transformation or even inclusion of metadata, or the preservation of its meaning. This article describes a migration project in which the available tools did not meet our needs, so a custom solution was built from scratch. In this project we focused on moving the metadata to the new system, and brought the content along as well.

Background

The *OLA Quarterly* (OLAQ) is the journal of the Oregon Library Association. First published in 1995, the journal had been hosted and managed by Pacific University within their CommonKnowledge repository since 2014. The full archival run of OLAQ was available digitally. At the time of the migration, there were 93 published issues.

CommonKnowledge is Pacific University's institutional repository, hosted on Bepress' Digital Commons software until 2020. OLAQ was a slightly odd fit within CommonKnowledge since it was not a product of the university, but in Digital Commons the journal had a functionally standalone space separate from the rest of the repository. Pacific decided they would migrate their institutional repository from Bepress to Hyku, a turnkey version of the open source Hyrax repository platform (Meetz and Laird 2020). They recognized in the planning phase that OLAQ would not be able to keep its dedicated space on the new platform.

The OLA Board contacted various universities in search of a new host institution, and Oregon State University's bid was the preferred choice. Oregon State University Libraries (OSUL) maintains a long-established Open Journal Systems (OJS) site, in partnership with the University of Oregon Library under the name "OJS at OregonDigital.org." OJS is a widely used open source software product from Public Knowledge Project (PKP) for publishing open access scholarly journals. It would be able to provide a dedicated site for OLAQ, and costs would be minimal. At the time, OJS at OregonDigital.org was operating on an older version of OJS (2.4.8), with plans to upgrade but resources not yet allocated. The adoption of OLAQ into OJS at OregonDigital.org would be just the nudge needed to complete the upgrade.

Project Planning

The project plan was drafted and approved by the OLA Board in June 2019, and an initial project team was formed at OSU Libraries. Drafted to the team were OJS product owners for OSU and UO, the Scholarly Communication & Publishing Services Librarian at Pacific University, the Communications Coordinator for OLAQ, one OSU Libraries software developer, and two OSU library technicians, with potential student assistance if needed. The target completion date was September 2019, ahead of the deadline set by Pacific University's expected completion date in November. The plan was uncomplicated:

1. Upgrade OJS at OregonDigital to version 3.x;
2. Decide on the design template for OLAQ;
3. Use the existing Bepress to OJS plugin to migrate the content;
4. Test the migration for completeness and accuracy;
5. Update DOIs and indexing;
6. Train editors to use the new software.

The project reality did not, however, follow the project plan. In October, the OJS2 to OJS3 upgrade had not measurably progressed, and early investigations had raised concerns about errors and data loss from the existing migration plugin offered by PKP. Several shortcomings with results of the plugin were clearly acknowledged in the plugin's documentation, including articles being out of order, irregularities with sections, and cover art files not being imported (Felczak 2017; Felczak [updated 2020]). Additional problems were identified, which were amplified by the particulars of our content and situation:

- OSU did not have administrator access to the source content in Digital Commons;

- The CSV to XML conversion script recommended by PKP to prepare data for their plugin introduced more potential errors and data loss (Vilhuber … [updated 2018]);
- We had missing and incomplete metadata in OJS required fields;
- We had metadata, especially at the issue level, that did not align perfectly with OJS but which we wanted to somehow preserve.

This paper's author, OSU's Metadata Librarian, had been very new to the job at the project's inception, but was called in at this point to consult. We initially aimed to identify strategies for handling the fallout from an imperfect but extant solution, but we realized that we had all the tools we needed to try something entirely different. OJS 3.x includes an XML import tool, the Native XML Plugin, and the OJS/PKP Native XML schemas are publicly available online. We also had the exported metadata, and experience writing XSL transformations. We were already facing a time investment for correcting introduced errors and possibly modifying the code, so deciding to change course was not destabilizing. We formulated a new plan: to create custom XSLT, transform the Digital Commons data in-house, and import XML via the Native XML Plugin. The team was reduced to the metadata librarian and software developer, with oversight by our shared department head and consultations available with the journal stakeholders.

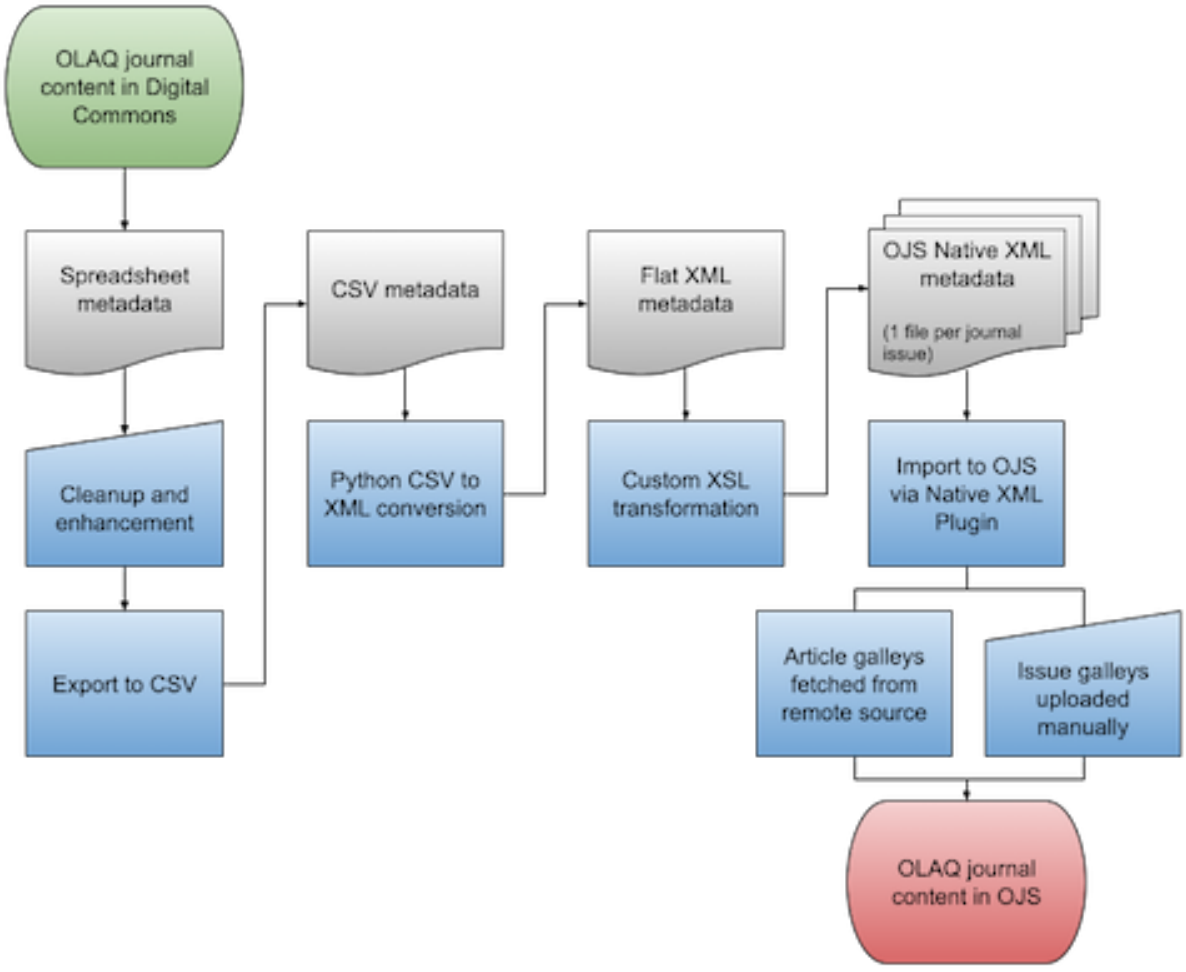

Figure 1: Digital Commons to OJS migration workflow.

Shortly after this pivot, we also concluded that migrating in tandem with upgrading our existing OJS 2.x system was too ambitious. The upgrade was more difficult than anticipated, and was unlikely to be completed before the Digital Commons repository was sunsetted. We resolved to launch OLAQ as a standalone journal, temporarily, on a separate OJS 3.x site while the existing OJS at OregonDigital site remained on OJS 2. This change did not have a significant negative impact on the outcome of the OLAQ migration, which would still be moved to OJS 3.x as intended, and it freed up the developer's time for immediate needs.

Data Preparation

Pacific University provided an up-to-date export of OLAQ metadata from Digital Commons, delivered as an Excel spreadsheet. For ease of reviewing and collaborating, the OSU team used Google Sheets to work with the metadata. Some data cleanup, normalization, and enhancements were required before converting the data to XML format. Key spreadsheet cleanup tasks that impacted the rest of the migration process included:

- Replacing XML reserved characters in the data with character references (`&amp;`, `&lt;`, `&gt;`);
- Verifying the correct order of contents (the ordering of contents in OJS follows the order of the XML import files);
- Correcting publication date formats to `YYYY-MM-DD`, and ensuring all items had publication dates (more on this later).

The cleaned metadata was exported from Google Sheets as a CSV file. We lightly modified a Python script found online to convert the CSV to flat XML (FB26 2010). The flat XML structure has a root element `<csv_data>`; each row of data is a record within a `<row>` element; and each item in the header row becomes an element name, as seen in this excerpt:

```
<row>
    <title>Communicating with Library Donors</title>
    <keywords>library, library marketing, library
       communications[...]</keywords>
    <disciplines>Archival Science; Cataloging and
       Metadata[...]</disciplines>
    <document_type>article</document_type>
    <doi>http://dx.doi.org/10.7710/1093-7374.1829</doi>
    <volnum>21</volnum>
    <issnum>4</issnum>
    <vol_iss>21(4)</vol_iss>
    <fpage>5</fpage>
    <lpage>9</lpage>
```

```
    <distribution_license></distribution_license>
    [...]
</row>
```

Snippet 1: XML output from csv_to_xml.py. A row in the CSV source document is represented by the row container element, and the CSV column headers become individual child elements.

This flat XML was used as the source data for the XSL transformation into the OJS/PKP Native schema.

Metadata Mapping and XSLT Development

In the next stage, the Digital Commons metadata fields were mapped to OJS elements according to the PKP/OJS Native XML schema, which could then be used to write XSLT that would produce a valid XML import file. Appendix A shows the fields that were present in the Digital Commons export, and the XPaths used for their element and attribute mappings in OJS. For unmapped fields, either no data was populated for that field in our dataset, there was no straightforward element match in the OJS schema, or the field is not meaningful in the OJS context (such as a Digital Commons system field). Some of the information contained in unmapped fields - specifically Publisher, ISSN, and Journal ID - is provided in the OJS's journal-level configuration. Over the course of the project, additional fields were added to the dataset to fill in gaps and facilitate processing, as more was learned and the transformation became more finely tuned.

The Data Import and Export page of the PKP Administrator's Guide lets intrepid readers know that the Native XML Plugin could be used for "[m]oving a large number of back issues and articles into OJS," but cautions that one will need "[a] basic understanding of XML" (PKP [n.d.]). Links to the schema XSL files and sample XML files are provided, along with several tips. We were unable to find any publicly available supporting documentation for the schema beyond these resources. XSD files are readable enough to experienced XML practitioners, and the overall field mapping was not complicated. However, definitions and guidance detailing how XML data values connect to software functionality would have been tremendously appreciated (especially for brand new OJS 3.x users). As a basic example, the `published` attribute on the `<issue>` element is stated in the schema to have a type `integer`; but the schema does not state that an attribute value of `1` will publish the issue immediately upon import while a `0` value means the issue will be imported in an unpublished state. Furthermore, our testing revealed that the import plugin has stricter requirements for import files than the schema, and therefore XML files that validate may still fail to import.

An XSLT stylesheet for converting the flat XML into OJS Native XML was built based on the field mapping. Oxygen XML Editor software was used to write the XSLT and run transformations. The style of the XSL code is somewhat explicit, naming each OJS destination

element in order and pulling from the source data to populate the element, as seen in this excerpt populating the Keywords field.

```
<xsl:if test="keywords/text()">
    <keywords>
        <xsl:for-each select="tokenize(keywords, ', ')">
            <keyword>
                <xsl:value-of select="."/>
            </keyword>
        </xsl:for-each>
    </keywords>
</xsl:if>
```

Snippet 2: Section of XSL stylesheet handling keywords.

The mapping and XSLT development happened in tandem with the OJS 3 installation, and credentials for the development site were given out just in time for the first fully valid output to be produced. We proceeded with testing the XML output using the Native XML Plugin, experimenting to determine connections between metadata and system functionality, and iterating until we ultimately had results we wanted.

Metadata Challenges

Given the vast differences between the origin and destination systems and our spotty understanding of schema details, troubleshooting and revising were ongoing throughout the project. Of the larger metadata challenges to solve, some were caused by a lack of complete data, and others by an abundance of it. Still other frustrations arose out of the design choices within the Native XML schema and the logic of the import tool. A few notable cases are described below.

One compromise in the move to OJS was the loss of metadata granularity for journal issues. In Digital Commons, especially in more recent years, OLAQ editors had described issue-level objects similarly to article-level objects, and Digital Commons issue records included disciplines, keywords, and guest editor biographies. OJS issue-level metadata fields are limited to Issue Identification (Numeration, Year, and Title), Publication Date, and Description. We were interested in preserving the issue-level metadata, so we accommodated as well as possible by placing all of the above information into Description fields so that the information would be available, even if some of the utility was lost. In working to address the issue description problem, we found that, while valid per the schema, repeated descriptive XML elements are not preserved on import. So while we originally mapped several of the issue fields to multiple `<description>` elements, each successive instance of `<description>` appeared to overwrite the previous one when the file was imported. Re-exporting the issue confirmed that only the

last-occurring instance of the element was saved. We revised the XSLT to combine multiple values in a single `<description>` element, adding HTML markup for headers and newlines within the field text to replicate the appearance of distinct fields for readers.

Overall, OLAQ's metadata was high quality, but there were a number of holes in the data including some that required consulting stakeholders to approve any solution. Two fields in particular were required for every OJS submission but incomplete in the Digital Commons dataset. ("Submission" is OJS terminology for an individual component of a journal issue, predominantly but not exclusively articles.)

- The first was author(s), including subfields for first name, last name, and affiliation. Authorship was diligently noted for all articles except editorials, but OLAQ also had separate submission objects for tables of contents and back matter components, and these pieces did not have stated authorship. We agreed to assign the journal name as the default author if none was given. (Because OJS enforces a Given Name + Family Name rule that does not accommodate corporate authors well, we ended up with Given Name = "Oregon Library," Family Name = "Association," and Affiliation = "OLA"). This was handled within the XSLT script.

- The second was a full `YYYY-MM-DD` publication date. In the source data prior to 2014, when OLAQ was moved into Digital Commons, dates were year-only. Issue titles indicated the season, as in Fall 2010. To fill in the gaps, we agreed to assign retroactive dates based on the season, where Fall issues were now recorded as being published on 09-01, Winter on 12-01, Spring on 03-01, and Summer on 06-01 of their given year. We altered the dates in the spreadsheet using a Google Sheets function.

Processing and Import

The Native XML Plugin provides means to import XML journal content into OJS. It is included "out of the box" and does not require separate installation. There is a command-line import option, but in this project we exclusively imported through the admin dashboard user interface. XML import files may contain one or multiple issues and/or articles. File size limitations do apply to imported XML files, though these can be adjusted in the system configuration. For our use case, the ability to easily delete, edit, and re-import content was essential, so our XSLT was designed to process the complete dataset and output one XML file per issue. Content imported using the plugin results in fully structured journal issues, which may be either immediately published or set to pre-publication, depending on attribute values. Ideally, little to no additional work is needed post-import.

We tested the transform-and-import workflow using three OLAQ issues representing distinct time periods and thus differing descriptive practices. This test set was sufficient to identify and resolve the metadata problems previously described. When the initial batch was successful, we generated three more issues and continued to review; then finally executed the XSLT across the full corpus to import all 93 issues.

The OJS infrastructure allowed us to fetch the PDFs remotely from Digital Commons and insert them as article galley files with the simple usage of a `<href src="{URL}">` reference in the XML. These PDF URLs were not present in the Digital Commons export data - the field exists, but was blank in our dataset. We were not successful in retrieving them with the XSLT due to the Bepress HTML being reportedly ill-formed. However, Google Sheets was more permissive, so we used the IMPORTXML function to add the PDF URLs to the source data. This process worked very reliably for the 873 submission objects. However, for unclear reasons, while submission galley files can be fetched remotely, issue galley files may only be added via the XML import file if embedded in base-64 encoding. The same is true for cover art files. OLAQ had both full-issue PDFs and cover art for each of the 93 issues. We decided to create local copies of the full-issue PDFs and manually add them to the issue objects post-import. We took the opportunity to regenerate the cover art files from the PDFs to replace the lower-resolution GIFs used in Digital Commons; these were added manually post-import as well. If we had been working with a larger body of content, we would likely have investigated encoding the PDFs and adding them to the import files. In our case, the admittedly tedious UI-button-clicking time was almost certainly shorter than the development time would have been. This is an area for potential future work, if the XSLT tools are to be carried forward.

Finishing Details

We created custom CSS to apply on top of the default OJS theme, which mimics the look of the former OLAQ site at Pacific University, which in turn had been designed to resemble the print journal's cover art style. Populating the static site content, such as about pages and policy information, was a manual copy-paste operation, in part because we only had normal public access to that static content in Digital Commons. Since our project had only a single journal to configure, this was not excessively laborious. With the content in place, we did a soft launch of the production site and invited the OLA stakeholders to review the work. While there were a few tweaks needed, happily no problems were found that required reprocessing any of the content.

As they shut down their Digital Commons repository, Pacific University implemented a base redirect to our OJS site. Setting identifiers in the metadata that resembled the original paths allowed for a simple redirect rule on our end for all submission-level items, though issue objects and static pages needed explicit redirects.

The bulk of the project work took place between November 2019 and February 2020; the early majority went toward installation of the software and development of the transform, while the execution of the final transformation and import of the full journal's metadata and content - the "actual migration" - required less than a week. Pacific University's timeline for sunsetting Digital Commons was extended at least twice, which was to our project team's benefit as we were consistently operating behind schedule. The official launch suffered from unfortunate timing, taking place in March 2020 just as COVID-19 sent us all out of our offices, but the journal's site has enjoyed strong readership and three new OLAQ issues have been published post-migration.

Reflections

In addition to the previously described metadata solutions and the manual process for uploading issue files, the overall project encountered plenty of other challenges and made plenty of other compromises. The biggest overall challenge was that OSU's upgrade from OJS 2 to OJS 3 could not be completed before the deadline to move OLAQ, so we are running two separate OJS instances, which is not an ideal situation. Work on the upgrade has progressed, but carving out time and resources from an established workcycle rhythm with many competing projects is more difficult without the looming deadline of loss of access to the content. There were plenty of minor irritations along the way, too; for instance, discovering that DOIs imported with the XML metadata could only be saved in the system if the DOI plugin is turned on prior to import.

While there is much to like about publishing in OJS, the move from Digital Commons meant some compromises for the journal as well. OLAQ's usage statistics from Digital Commons were lost, and all items' views were reset to zero in OJS. A few Digital Commons features are not available in OJS, such as the readership map and hidden keywords, and browsing functionality and analytics are less robust.

The OJS 3.x software was new to the project team, and there was an associated learning curve. The lack of official documentation for trying to migrate journals from outside systems into OJS or for understanding the XML schema indicated this is not a process that the developer feels strongly about supporting - we often had the sense that we were not using it the way it was meant to be used. There was a great deal of trial and error and fixing and retrying; but that's exactly the spirit of work that our approach fostered. Designing in-house transformation tools that were customized for our particular needs lent itself perfectly to a testing and iterating workflow. The metadata-forward approach meant we had precise control over the results, at least within the confines of the system. Output could be easily destroyed and remade with improvements. At the conclusion of the project, while the results might not have been perfect, we were able to feel confident that they were as close to perfect as we could reasonably have gotten.

Our XSLT pipeline was not designed with the intent to cover every possible Digital Commons to OJS migration, but worked well for our own use case. The main advantage of the XML-based, metadata-forward approach is the degree of control it provides over the data. Other institutions working to migrate content from Digital Commons to OJS may be able to repurpose these tools; adjustments may be needed to the XSLT code, but the majority of the work is likely to be in data preparation.

Since the OLAQ migration has concluded, OSU Libraries does not currently have a use case for further development of the Digital Commons to OJS migration tools. In addition to supporting OLAQ editors with the transition, we are continuing to work toward moving our existing OJS at OregonDigital.org journals from our OJS 2.x instance onto the OJS 3.x, alongside OLAQ. That said, the Digital Commons to OJS XSLT pipeline would benefit from additional stylesheet

versions corresponding to XML schema updates in recent OJS releases, since adopters going forward are less likely to be installing OJS 3.1 versus a later version. Another possible enhancement, as previously mentioned, would be automating the incorporation of base-64 encoded content files within the XML output, to avoid having to upload issue galleys post-import; this would also provide flexibility for including article galleys without relying on a live remote source location. Potential adopters of this toolset may also have special circumstances that suggest other directions for development not anticipated here.

The project to migrate OLAQ from Digital Commons to OJS involved building a custom solution despite the existence of established and tested tools. This was a bold decision, and we ultimately can't measure how different the results would have been if we had used the existing tools instead. We can say with certainty that the migration was successful, and that we gained familiarity with OJS 3.x that will be useful as we continue to support OLAQ and other journals on the platform.

Acknowledgements

This project leaned heavily on the hard work of OSU Libraries Analyst Programmer Ryan Wick, and the coaching of our department head, Margaret Mellinger. We are also grateful for the patience of our associates at OLA, especially Charles Wood and Elaine Hirsch. Finally we would like to extend our congratulations to Pacific University on their beautiful new CommonKnowledge repository.

Read OLA Quarterly *on OJS at https://journals3.oregondigital.org/olaq.*
Find the code on GitHub at https://github.com/osulp/bepress-ojs-xslt.

Appendix A. Bepress Digital Commons to OJS 3.1 Field Mapping

This table shows the fields included in the OLAQ metadata export from Digital Commons and the destination OJS fields to which they are mapped. For OLAQ, objects with document type "full_issue" became OJS Issues, and all other document types were migrated as OJS Submissions.

Data transformations were applied to many of the fields during the course of both spreadsheet cleanup and XSL processing, which are not specified in the table. Readers are invited to visit this project's Github repository for those details. It should also be noted that the XPaths given in this table are specific to OJS 3.1, and that more recent versions of the schema are known to deviate for some of these.

Bepress Field	OJS XPath - Submissions	OJS XPath - Issues
title	//article/title	//issue/issue_identification/title
keywords	//article/keywords	//issue/description
disciplines	//article/disciplines	//issue/description
document_type	//article/@section_ref	
doi	//article/id[@type="doi"]	//issue/id[@type="doi"]
volnum		//issue/issue_identification/volume
issnum		//issue/issue_identification/number
fpage	//article/pages	
lpage	//article/pages	
distribution_license	//article/licenseUrl	
abstract	//article/abstract	//issue/description
comments		
fulltext_url		
peer_reviewed		
publication_date	//article/@date_published	//issue/date_published //issue/issue_identification/year
rights	//article/copyrightYear //article/copyrightHolder	
license_statement		
filename		
erratum		
publisher		
issn		
journal_id		
author[#1-5]_fname	//article/authors/author/givenname	
author[#1-5]_mname	//article/authors/author/givenname	
author[#1-5]_lname	//article/authors/author/familyname	
author[#1-5]_suffix		
author[#1-5]_email	//article/authors/author/email	

author[#1-5]_institution	//article/authors/author/affiliation	
calc_url	//article/id[@type="public"]	
context_key		
issue		//issue/id[@type=”public”]
ctmtime		

For unmapped fields, either no data was populated for that field in our dataset, there was no straightforward element match in the OJS schema, or the field is not meaningful in the OJS context (such as a Digital Commons system field). Some of the information contained in unmapped fields - specifically Publisher, ISSN, and Journal ID - is provided in the OJS's journal-level configuration.

Bibliography

FB36. 2010. Convert CSV to XML (Python code). ActiveState Code: Recipes; [cited 2021 July 15]. Available from: https://code.activestate.com/recipes/577423-convert-csv-to-xml/

Felczak M. [Internet]. [updated 2020 July 23]. OJS bepress Digital Commons import plugin (Readme); [cited 2021 July 15]. Available from: https://github.com/mfelczak/bepress

Felczak M. 2017. Migrating bepress Digital Commons Journals to OJS (blog post). Public Knowledge Project; [cited 2021 July 15]. Available from: https://pkp.sfu.ca/2017/12/07/migrating-bepress-digital-commons-journals-to-ojs/

Meetz J, Baird L. 2020. Migrating an IR at a small, liberal arts university [presentation]. Northeast Institutional Repository Day (NIRD); 2020 December 3; Virtual. https://doi.org/10.13028/0q6z-2g81.

PKP. [Internet]. [no date]. PKP Administrator's Guide: Data Import and Export; [cited 2021 July 15]. Available from: https://docs.pkp.sfu.ca/admin-guide/en/data-import-and-export

Vilhuber L. [Internet]. [updated 2018 February 26]. Migration scripts Bepress (CSV) -> OJS (Readme); [cited 2021 July 15]. Available from: https://github.com/journalprivacyconfidentiality/jpc-migration

About the Author

Cara M. Key (she/her) is the Metadata Librarian at Oregon State University Libraries & Press, where she supports open access publishing and digital repository services. Contact: cara.key@oregonstate.edu