Massive Newspaper Migration - Moving 22 Million Records from CONTENTdm to Solphal

Alan Witkowski
J. Willard Marriott Library
University of Utah
alan.witkowski@utah.edu

Anna Neatrour
J. Willard Marriott Library
University of Utah
anna.neatrour@utah.edu

Jeremy Myntti
J. Willard Marriott Library
University of Utah
jeremy.myntti@utah.edu

Brian McBride
J. Willard Marriott Library
University of Utah
brian.mcbride@utah.edu

Abstract

Utah Digital Newspapers is a pioneering digital newspapers program at the University of Utah J. Willard Marriott Library. Recently, a small project team completed a successful migration away from CONTENTdm onto a home-grown system called Solphal, built using open-source applications. The migration process is detailed along with examples of scripts used to prepare and enhance metadata. Transitioning away from a limiting vendor-based solution to a home-grown system has enabled the Utah Digital Newspapers program to be more responsive to user requests as well as realizing greater efficiencies in hardware and software. The platform has opened up new possibilities for the future as the collection continues to grow.

History of digital newspapers at the J. Willard Marriott Library

The Utah Digital Newspapers (UDN - https://digitalnewspapers.org) program is one of the largest and longest running statewide programs for making newspapers publically available online. The digital asset management system that made UDN content available was originally CONTENTdm. Due to the large amount of content being made available in UDN, it was determined that CONTENTdm no longer met the needs of the program and a migration to a new system was necessary. This article will briefly discuss the history of the UDN program, along with the process and issues of migrating the content to a

home-grown system built off of open-source tools. The new repository system, called Solphal, is built on the high-performance PHP framework Phalcon (https://phalconphp.com), Solr (http://lucene.apache.org/solr) for indexing and metadata storage, and the web server NGINX (https://nginx.org). While many of the processes that were used in this migration and detailed in the article are unique to how CONTENTdm handled newspaper repository data, and the necessary improvements undertaken to migrate to a new system, the information and processes detailed can be translated to other systems to help other libraries in their own migrations.

The Utah Digital Newspapers (UDN) program was launched in 2001 with the receipt of a Library Services and Technology (LSTA) grant to digitize three newspapers (Arlitsch, Yapp, & Edge, 2003). Subsequent participation in the National Digital Newspaper Program (NDNP) and National Endowment for the Humanities (NEH) grants raised the total number of articles in UDN to over 20 million in 2016. Early on in the history of the newspapers program at the University of Utah, article level metadata was identified as an essential part of the architecture for the newspaper program, allowing users to browse and search for advertisements, births, weddings, and obituaries (Herbert & Arlitsch, 2004). This was accomplished by using a custom indexing program written for the J. Willard Marriott Library by iArchives, where NDNP batches were then split at either the article or page level and prepared for CONTENTdm ingest.

Newspaper digitization continues to be a popular project for LSTA, Utah State Historical Records Advisory Board (USHRAB), and privately funded grants, with recent projects including the Daily Herald, Park Record, American Fork Citizen, and Vernal Express. In addition to grant projects, several cities or counties in Utah have committed funds towards newspaper digitization as well as the processing of current born digital newspaper content for inclusion in UDN. As of June 15, 2017, there were 141 newspaper titles, 191,599 issues, 2,116,565 pages, and 20,475,912 articles available in UDN.

Scalability issues, configuration issues, and customizations required to host newspapers content on CONTENTdm

Large newspaper collections on the UDN CONTENTdm server needed to be split into multiple sub collections for better performance. For example, the Salt Lake Tribune was split into over thirty different sub collections. Custom queries for users were created on the front page of UDN in order to allow users to browse across multiple collections for a single paper by year. In addition, custom search forms were developed for each newspaper title. Another important issue affecting patrons was slow page response times. It was quite common for page load times to reach or exceed ten seconds, and this issue adversely affected user experience. In addition, lack of control over the image viewer in CONTENTdm often caused UDN users to experience frustration when they wanted to download or save snippets of news articles. Maintaining the CONTENTdm servers required approximately .75 FTE staff, covering both systems and application support and maintenance. Additional work beyond this was required whenever there was a new CONTENTdm software update.

The J. Willard Marriott Library went through a Digital Asset Management Systems Review beginning in 2013 when it was clear that there were scalability, performance, reliability, and user interface issues with CONTENTdm, both for UDN and other digital collections (Masood & Neatrour, 2014). As part of this process, various digital asset management systems were reviewed, including ChronAm for newspapers. After the systems review, ChronAm proved not to be a viable option for newspapers

content for our digital library program. The primary reason was that implementing ChronAm locally would have required further customization and development work to accommodate the article level display of UDN newspapers. With a limited amount of staff to devote to custom programming for both a newspaper and a digital repository, the library also made the strategic decision to support one single digital library system for both newspapers and cultural heritage content such as images, videos, and other multimedia files in addition to newspapers. In the migration timeline to Solphal, Utah Digital Newspapers was migrated first because the metadata for newspapers and formatting was relatively standardized compared to our digital collections data. The newspapers content was easily made static, as newspaper collections are generally digitized in bulk, without the need for ongoing metadata correction and clean-up.

Preparing UDN for migration

Prior to migration, the existing metadata in UDN needed to be reviewed and standardized before moving to Solphal. Even though digitization for UDN was completed by vendors and NDNP batches are standardized, there were still variations in both the file types, formats, and metadata in UDN, since the newspapers program had been running for such a long time.

Project management for the UDN migration was handled by the lead application developer, who developed scripts to handle data assessment, remediation, and enhancement. When additional manual work was necessary for data clean-up, metadata librarians and staff were consulted. E-mail updates on the status of the project were sent frequently, ensuring that all team members were on the same page. The core team for the migration consisted of the lead developer, the supervisor for the Digital Library Services Department, the supervisor for Digital Infrastructure Development, a metadata librarian, and the assistant head of digital operations for the J. Willard Marriott Library. The relatively small size of the team contributed to the migration process functioning in a lightweight, quasi-agile way, with the majority of the work for the project being completed in a six month timeframe.

Metadata preparation for migration followed the general practices of assessment, remediation, and enhancement which are often required for any digital library systems migration. Custom scripts developed to handle the migration are detailed throughout this article, illustrating the different aspects of the migration process. Bash scripting (https://www.gnu.org/software/bash) was particularly useful, as it allowed us to target particular aspects of metadata that needed to be cleaned up in CONTENTdm first prior to the migration.

Metadata template assessment and standardization

Before work on fixing metadata began, we needed to normalize the schema for all 361 UDN collections. The command in **Fig 1** was used to get field counts across all collections to assist with this step of metadata assessment. Field counts were used to get a sense of the varying metadata templates that were used in defining the fields for each newspaper collection. While the metadata was standardized, there were still some newspaper collections that had empty or non-standardized fields.

```
find -iname "config.txt" -exec sh -c 'echo {}; cat {} | wc -l' \;
```

Fig 1 - A bash command to show field counts for each collection. This finds every file called "config.txt", which contains the CONTENTdm schema for a collection, and outputs the full path to the file with a line count.

Based on the field counts, a subsequent review based on this data set a standardized newspapers schema across all collections, making sure that all the fields had no name variants. Core fields in UDN included title, type, date, year, month, day, rights, publisher, creator, and page. Empty fields in CONTENTdm were noted and not migrated into Solphal.

CONTENTdm metadata overview

The metadata for a CONTENTdm collection is stored in a flat file format named desc.all (**Fig 2**) and was approximately a total of 65GB in size pre-migration. The digital library programming staff had developed a variety of processes over the years that took advantage of the ability to script against a CONTENTdm desc.all file for large scale metadata enhancement. For example an enhancement project to add additional geographic values in a CONTENTdm digital collection was accomplished through adding latitude and longitude values via scripting and reindexing the desc.all file (Neatrour, Morrow, Rockwell, & Witkowski, 2011). This type of scripting and reindexing of the desc.all file is possible for institutions who self-host CONTENTdm. In preparing metadata for migration, the metadata in the desc.all file was corrected and enhanced through an iterative process.

```
<dmcreated>2004-02-27</dmcreated>
<dmmodified>2004-02-27</dmmodified>
<dmrecord>0</dmrecord>
<title>American Eagle, 1897-05-08 Page 1</title>
<subjec></subjec>
<descri></descri>
<creato>American Eagle Publishing Co.</creato>
<publis>Digitized by: Univ. of Utah</publis>
<contri></contri>
<dateor>1897-05-08</dateor>
<date></date>
<type>page</type>
<format>text/PDF</format>
<identi></identi>
<source></source>
<langua>eng</langua>
<relati></relati>
<covera></covera>
<rights>Material in the public domain. No restrictions on use.</rights>
<itemye>1897</itemye>
<itemmo>May</itemmo>
<itemda>08</itemda>
<itempa>Page 1</itempa>
<itemtr></itemtr>
<genre>newspaper</genre>
<fullrs></fullrs>
<find>2.pdf</find>
<dmaccess></dmaccess>
<dmimage></dmimage>
<dmad1></dmad1>
<dmad2></dmad2>
<dmoclcno></dmoclcno>
<dmcreated>2004-02-27</dmcreated>
<dmmodified>2004-02-27</dmmodified>
<dmrecord>1</dmrecord>
<title>American Eagle, 1897-05-08 Page 2</title>
<subjec></subjec>
<descri></descri>
```

Fig 2 - An excerpt from the desc.all metadata file. The highlighted area is one record in UDN

Metadata enhancement

One of the priorities for metadata enhancement in Solphal was developing new fields to improve search functionality that had previously been accomplished through workarounds in CONTENTdm. For example in Solphal, it was desirable to be able to browse by paper name. Previously, this type of functionality was accomplished through custom queries developed to compensate for the split newspapers collections required by the architecture needed to support such large collections on a single CONTENTdm server. This previous architecture was not going to be present in Solphal, and needed to be taken into account in ensuring the same type of functionality post-migration. Since there was no dedicated metadata field for the name of the newspaper without using the existing title field which had additional metadata, such as issue, date, page, or article information, the paper name for each record had to be extracted from the title and inserted into a new field. The format for each title depends on the type of record. The new paper name field needed to be added to each level of newspaper record type, at the issue, page, and article level across the entire newspaper repository. Most of the records followed the following format.

Record Type	Title Format
Issue	*Paper_Name, Issue_Date*
Page	*Paper_Name, Issue_Date, Page_Number*
Article	*Paper_Name, Issue_Date, Article_Title*

To evaluate the state of each paper name in UDN, the bash command in **Fig 3** was used to build a list of names. Sometimes there wasn't a comma directly after the paper name, so a more complicated regular expression had to be built to handle all cases, with "Mt. Pleasant Pyramid" as a special case.

```
find -iname "desc.all" -exec grep -P "(?<=<title>)(Mt. Pleasant
Pyramid|.+?)(?=[.,:(])" {} -o \; | sed 's/ *$//' | sort | uniq
```

Fig 3 - A bash command to build a list of paper names by searching every title. The *find* program searches for files called desc.all, then executes *grep* on each one. *grep* searches the title field for all characters leading up to a period, comma, colon, or opening parenthesis, then returns the match. The *sed* program is used to trim trailing whitespace in the match. *sort* and *uniq* are used to sort the output and return only unique lines.

Metadata assessment and manual enhancement

In some cases, record titles didn't contain a paper name or date. These had to be fixed manually. By assessing the desc.all file, the following commands were used to build reports to make the manual editing process easier and more efficient. Reports like the one in **Fig 4** were used for directing manual work, where digital library metadata staff were able to get context for missing data by seeing the paper name, date, and title alongside articles that were missing this information. These targeted manual edits were then added back into the desc.all file so all articles across the repository had consistent paper names and dates.

```
gawk 'BEGIN {FS=OFS="\t"} NR > 1 { print $1, $2 }' desc.all.tsv | egrep -v
$'\t'"(Utah Enquirer|Provo Daily Enquirer|Territorial Enquirer)" -C 2
```

```
------

5637    Provo Daily Enquirer, 1892-11-16, The World Enriched
5638    Provo Daily Enquirer, 1892-11-16, Mission Notes
5639    The Prizes of Literary Work
5640    Provo Daily Enquirer, 1892-11-16, The India Rubber Worm
5641    Provo Daily Enquirer, 1892-11-16, Too Powerful
--
108185  Provo Daily Enquirer, 1896-02-06, Masthead
108187  Provo Daily Enquirer, 1896-02-06, Age of Consent
108189  A Happy New Year
108190  Provo Daily Enquirer, 1896-02-06, Notice
108191  Provo Daily Enquirer, 1896-02-06, Foreign Gatherings
--
111431  Provo Daily Enquirer, 1896-04-24, Combative Congressmen
111432  Provo Daily Enquirer, 1896-04-24, Utah Public Buildings
111433  Again in Public
111434  Provo Daily Enquirer, 1896-04-24, Utah Men Get Positions
111435  Provo Daily Enquirer, 1896-04-24, Gets a Third Term
```

Fig 4 - A bash command to generate bad titles with surrounding record titles for context. The *gawk* program is used to parse the desc.all.tsv file and print the first two columns, which is piped into grep to only display titles that start with Utah Enquirer, Provo Daily Enquirer, or Territorial Enquirer.

Another method of dealing with manual fixes in a shared work environment was exporting reports and placing them in Google Sheets, along with links for manual editing. The report of bad titles listed below illustrates a piece of that process.

```
gawk 'BEGIN {FS=OFS="\t"} NR > 1 { print $1, $2 }' desc.all.tsv | egrep -v $'\t'"(Utah Enquirer|Provo Daily
Enquirer|Territorial Enquirer)" | sed 's|^|http://udn6.lib.utah.edu:81/cgi-
bin/admin/edittxt.exe\?CISOROOT=/de2\&CISOPTR=|g'

------

http://udn6.lib.utah.edu:81/cgi-bin/admin/edittxt.exe?CISOROOT=/de2&CISOPTR=5639    The Prizes of Literary Work
http://udn6.lib.utah.edu:81/cgi-bin/admin/edittxt.exe?CISOROOT=/de2&CISOPTR=108189  A Happy New Year
http://udn6.lib.utah.edu:81/cgi-bin/admin/edittxt.exe?CISOROOT=/de2&CISOPTR=111433  Again in Public
```

Fig 5 - Bash command to generate a list of bad titles with links to the metadata editor in CONTENTdm. This command works similar to the one in **Fig 4**. The main difference is each line is prepended with a url. Below it is the output that can be copied into a spreadsheet program.

Scripting metadata enhancements

Multiple passes through the desc.all file were needed as part of the clean-up process prior to migration. While some tasks were completed through manual editing, many cleanup processes were automated through a series of scripts. For example, some records had titles that had the page number or article title listed in front of paper title. Variations of the following command was used to handle the various cases in which this occurred.

```
cat desc.all | perl -pe 's|<title>(?!Page)(.+) (Provo Daily Enquirer), ([0-
9]{4}-[0-9]{2}-[0-9]{2})</title>|<title>\2, \3, \1</title>|g' >
desc.all.fixed
```

Fig 6 - A bash command to change title from [*Page# Paper, Date*] to [*Paper, Date, Page#]* and generate a fixed desc.all file. Perl is used to run a regular expression on the title line with pattern groups, then each pattern group is rearranged in the substitution.

Fixing title/date inconsistencies

Once all the titles were in the correct format, other metadata fields were looked at more closely. For example, many records had different dates in the Title field versus the Date field. The following command lists date mismatches for each collection. Using this report, the desc.all file could be manually edited and re-uploaded.

```
find -iname "desc.all" -exec sh -c 'echo {}; egrep "(<title>|<dateor>)" {} |
egrep "[0-9]{4}-[0-9]{2}-[0-9]{2}" -o | sed "N;s/\n/\t/g" | gawk
"BEGIN{FS=\"\t\"} { if(\$1 != \$2) { print \$0 } }" ' \;
```

Fig 7 - A command to print out inconsistent dates in title and date fields. The command works by running *grep* on each desc.all file to find the title or date line. Another regular expression finds dates in the format YYYY-MM-DD within those lines. *sed* converts each title-date pair into a tabs-delimited line, then *gawk* is used to print only date values that are different.

Fixing miscellaneous fields

Before migration there were around 1500 different values for the Type field (**Fig 8**). This was condensed down to exactly nine. The Type field in UDN isn't an example of the standard usage that someone might expect from the Dublin Core Type Vocabulary, instead in the context of this repository, the Type field was drawn from the article level metadata associated with newspapers content, and allowed users to do more advanced searches for types of newspapers content such as advertisement, article, birth, death, issue, masthead, page, wedding, and articles that were unclassified. Variants in the Type field, with a sample listed below, were condensed for better faceting in the new UDN.

```
article; local performances; technology
article; local performances; theater
article; local performances; theater; music
article; loca news
article; Logan Leader
article; logging; accidents, injuries
article; logging
article; logging; local businesses
article; logging;local businesses
article; logging; mining; colonization and settlement
article;l technology
article;l theater
...
```

Fig 8 - Examples of bad Type values

Migrating to Solphal

Build desc.all.tsv and determine parent record

Compound objects are records which have a parent record that contains the bulk of the metadata, with multiple children records. In CONTENTdm, this relationship is not stored in the desc.all file. Instead there is a **supp** directory stored alongside the images directory which contains a subdirectory for every record, each of which contains a single XML file which stores the parent record id. For UDN's massive size, this lead to a large amount of inodes, which are data structures that describe a filesystem object, being used in the ext4 filesystem. This hinders scalability if the filesystem runs out of inodes, which cannot be adjusted after the filesystem has been created.

To keep the compound record structure from CONTENTdm intact, a special parent field which contains the id of the parent record had to be created. Issue records have a parent value of 0, indicating no parent. The script in **Appendix A** builds a tsv file that contains the parent record value.

File storage architecture

CONTENTdm stores files inside directories named after the collection alias, with all pdfs and thumbnails in a single subdirectory named **image**. Storing files this way can slow down performance, as reading directories with a large number of files is slow. The largest collection in UDN pre-migration, slherald19, contained 529,837 files in a single directory.

In Solphal, files were hashed using the sha1sum program, then renamed to [*sha1sum_hash*].[*file_extension*]. Each file is stored in buckets by taking the first two characters of the hash as the first bucket, and the next two characters as the second bucket.
For example **afde05e975201511a29a511819eed3fc453764ae.pdf** would be stored as **/dlstorage/udn_files/af/de/afde05e975201511a29a511819eed3fc453764ae.pdf** on the server. Thumbnails are stored in a separate directory with the same naming convention. With each bucket having 256 subdirectories, there are a total of 65,536 subdirectories. With around 22 million files, this gives about 330 files per subdirectory and allows for balanced growth without human intervention. Another benefit to this system is that duplicate files aren't stored twice and are easily detectable by faceting the filename field in Solr.

Converting metadata into Solr format

To reduce the amount of indexing and duplication of data, Solr is used as the primary data store for UDN. Metadata is backed up two times a day, and metadata that is in the process of being ingested is also backed up in a workflow management tool. In order to get data into Solr, ingest docs using XML (**Fig 9**) were created from the desc.all metadata files using a python script. Instead of defining a static schema, we use Solr dynamic fields to determine the field type based on the ending suffix, shown in the table below.

Suffix	Field Type
_t	General text fields that are tokenized by spaces and other characters in the index.
_s	String fields that aren't tokenized at all in the index.

_tdt	Date fields that can be compared and used in ranged searches.
_i	Integer fields.

This lets us add any arbitrary field in a Solr doc without having to define it first in a schema file.

```
<field name="id">1</field>
<field name="thumb_s">/28/35/283593ad79aebad753f81b700f46bff7179b5d7d.jpg</field>
<field name="file_s"></field>
<field name="parent_i">0</field>
</doc>
<doc>
<field name="paper_t">American Eagle</field>
<field name="title_t">American Eagle, 1897-05-08 Page 1</field>
<field name="creator_t">American Eagle Publishing Co.</field>
<field name="publisher_t">Digitized by: Univ. of Utah</field>
<field name="year_t">1897</field>
<field name="month_t">May</field>
<field name="day_t">08</field>
<field name="date_tdt">1897-05-08T00:00:00Z</field>
<field name="type_t">page</field>
<field name="rights_t">Material in the public domain. No restrictions on use.</field>
<field name="page_t">Page 1</field>
<field name="oldid_t">americaneagle 1</field>
<field name="id">2</field>
<field name="thumb_s">/28/35/283593ad79aebad753f81b700f46bff7179b5d7d.jpg</field>
<field name="file_s">/6a/59/6a59096aa1b9d06b821625e030bec919a3be43c0.pdf</field>
<field name="parent_i">1</field>
</doc>
<doc>
<field name="paper_t">American Eagle</field>
<field name="title_t">American Eagle, 1897-05-08 Page 2</field>
<field name="creator_t">American Eagle Publishing Co.</field>
```

Fig 9 - Solr XML ingest file. Highlighted is one record.

The *paper_t* field is created in the script using the regular expression from **Fig 3**. The field *oldid_t* was created to handle redirects from old CONTENTdm urls, to ensure that cached links from the old repository would still resolve successfully. It contains the original collection alias and CONTENTdm record number, which gets extracted from old CONTENTdm reference urls by NGINX and translated to Solphal's details page handler.

Once all of the Solr XML docs are created, Solr's DataImportHandler module is used to ingest the data. Indexing UDN's 55GB of metadata takes around 2.75 hours.

Page caching with NGINX

Since newspaper content doesn't change often, UDN utilizes NGINX's fastcgi cache module to speed up page requests (**Fig 10**). When a URL is visited for the first time, NGINX generates a static html version of the page. Subsequent requests to the same url pull from the same static file, reducing the processing time associated with generating html code from PHP, performing Solr queries, and database calls. This reduces pages requests from up to 700 milliseconds down to 5-10 milliseconds.

```
fastcgi_cache_path /var/cache/NGINX levels=1:2 keys_zone=DEFAULT:1000m inactive=5000m;
fastcgi_cache_key "$scheme$request_method$host$request_uri";

server {

	...

	set $fastcgi_skipcache 0;
	if ($uri ~ "^/login") {
		set $fastcgi_skipcache 1;
	}

	location ~ \.php$ {

		...

		fastcgi_cache DEFAULT;
		fastcgi_cache_valid 200 24h;
		fastcgi_cache_bypass $cookie_PHPSESSID $arg_nocache $fastcgi_skipcache;
		fastcgi_no_cache $cookie_PHPSESSID $fastcgi_skipcache;
	}

	add_header X-Cache $upstream_cache_status;
}
```

Fig 10 - Relevant parts of the NGINX config to enable page caching.

Performance

Slow performance, scalability, and sustainability concerns were the main factors in choosing to migrate from CONTENTdm to Solphal. In CONTENTdm, a full reindex of the repository would take approximately 1,440 hours with the site online and 240 hours with the site offline. In contrast the equivalent index in Solphal would take 2.75 hours to index all 55 GB of Newspapers metadata, with incremental indexing taking milliseconds. Page load times significantly affected user experience on CONTENTdm, with wait times sometimes reaching up to 10 seconds. In Solphal average page load times are 150 milliseconds for cached content and 1 second for uncached. Another user experience issue that was problematic in the previous version of UDN was the inconsistency in the CONTENTdm PDF viewer for newspapers content across different browsers and operating systems. Solphal uses PDF.js (https://mozilla.github.io/pdf.js/), an open-source library that uses javascript to render PDFs for a consistent look and feature set across all browsers. User requests for support in accessing image files were frequent when UDN was on CONTENTdm, but have diminished significantly since moving to the new system. User feedback on Solphal for UDN has been positive, with people commenting on the ease and speed of searching the site for genealogical information. By far, the most common request from users is that more content be added to UDN. Another benefit of moving to home-grown system is the ability to act directly on user support and feature requests. User feedback is discussed and incorporated into monthly enhancements and updates in Solphal.

The UDN migration had a variety of benefits for the digital library program at the J. Willard Marriott Library that went beyond providing a better experience for UDN users. The licensing, hardware, and personnel costs decreased. The server footprint for the system previously needed to run CONTENTdm decreased and resulted in a significant savings after migrating to Solphal. UDN alone previously

required a server with dual processors containing 8 cores and 96 gigabytes of memory, the new system is run on two virtualized servers. The indexing server, which is also used for other digital library collections, utilizes 10Gb of memory and eight cores. The front-end web server uses 4Gb of memory and four cores. In addition, the UDN migration was used as a test case in preparing to migrate the library's digital collections to Solphal as well, which are now available at https://collections.lib.utah.edu.

Future directions

With the change in infrastructure for UDN, we are now able to pursue more large newspaper projects to continue making historical newspaper content from Utah freely available online.
For the majority of content that has been ingested into UDN, the newspapers have been segmented at the article level. After conducting several tests and reviewing the user benefits of article level vs. page level newspaper content, it was decided that while article level newspapers were preferred, this wasn't always possible due to the high costs related to article segmentation. The page level content still allows the user to search the full text of the newspaper; they are taken to the page where the search terms are located rather than the individual article. By choosing to ingest page level content, it is now possible for many of our smaller partners to add more of their newspaper content to UDN if they don't have enough money to fund the article segmentation.

One new area that we have started to explore is ingesting born digital newspapers. In a project with the Park City Historical Society, we were able to take twelve years of born digital PDF pages from the Park Record (http://bit.ly/2saJhwL) and successfully ingest them into UDN. With the success of this project, we are investigating how we can obtain more born digital newspaper content to ingest and make available. At the same time, we want to explore ways of harmonizing the blended content of both article level and page level newspapers.

Conclusion

In retrospect, there are a few things that could have been done differently during migration. In UDN, the title field contained information already found in the metadata, such as paper name and date. The article title could have been pulled from the title field into a separate field, then the title field could be deleted entirely. This saves space in the index, and makes metadata remediation easier because metadata isn't being duplicated.

Prior to the migration it would have been beneficial if extraneous metadata had been removed earlier, such as duplicate rights and publisher metadata at the article level. After the experience of needing to perform metadata enhancements to compensate for workarounds that relied on software functionality as opposed to well-structured metadata, prioritizing well-structured metadata first is going to be a core value for the digital library team moving forward. In newspaper projects, exploring better ways to model the data for related titles is also a good project to pursue, for example ensuring a consistent user experience for people reading papers that have morning, midday, and evening editions.

By transitioning to an open-source framework, UDN is well positioned to continue to build on previous success in making Utah Digital Newspapers freely available to the public. With Solphal, the newspapers program is no longer encumbered by slow page load times and inefficient system architecture. The transition to using virtual machines and cloud-based architecture as opposed to a dedicated server with a large amount of memory has also freed up resources that can be better spent

elsewhere in service of the digital library program at the J. Willard Marriott Library. Using bash scripting and a lightweight, iterative approach to newspapers metadata remediation and preparation, this large scale migration was completed in a timely fashion. The migration of UDN also informed subsequent work with digital library migration as both CONTENTdm repositories were migrated to Solphal. After seeing the significant gains in systems architecture, staffing needs, and the ability of UDN to be more responsive to patron feature development, the only regret about migrating away from a vendor-based solution is that we didn't pursue it sooner.

Bibliography

Arlitsch, K., Yapp, L., Edge, K. (2003). The Utah Digital Newspapers Project. D-Lib Magazine, 9 (3). http://www.dlib.org/dlib/march03/arlitsch/03arlitsch.html

Herbert, J., and Arlitsch, K. digitalnewspapers.org. (2004) The Serials Librarian, 47 (1-2). 99-115. http://www.tandfonline.com/doi/pdf/10.1300/J123v47n01_07

Masood, K. and Neatrour, A. (2014). Digital Asset Management Systems Options: Report of the University of Utah Libraries Dam Review Task Force. Mountain West Digital Library Webinar. http://mwdl.org/events/DAMS_options.php

Neatrour, A., Morrow, A., Rockwell, K., Witkowski, A. (2011). Automating the Production of Map Interfaces for Digital Collections Using Google APIs. D-Lib Magazine, 17 (9/10). http://www.dlib.org/dlib/september11/neatrour/09neatrour.html

Appendix

```
#!/usr/bin/env python3
import re
import glob
import csv
import sys
import os

# list of fields to use
fields = ['dmrecord', 'title', 'find', 'identi', 'fullrs', 'ark']

# parse index.xml or newsindex.xml in supp directory and grab node value
def get_field(path, field):
    for line in open(path, 'r'):
        match = re.search("<" + field + ">([^>]*)</" + field + ">", line)
        if match:
            return match.group(1)

    return "-1"

# check for supp record and return parent dmrecord
def get_parent(dmrecord):
    global cache_path
    for supp_bucket in supp_buckets:
        index_path = cache_path + "supp/" + supp_bucket + dmrecord + "/index.xml"
        if os.path.exists(index_path):
            return get_field(index_path, "parent")

        # check for newsindex.xml
        newsindex_path = cache_path + "supp/" + supp_bucket + dmrecord + "/newsindex.xml"
        if os.path.exists(newsindex_path):

            # Check type
            if get_field(newsindex_path, "itemtype") == "Page":
                return get_field(newsindex_path, "issue")
            elif get_field(newsindex_path, "itemtype") == "Article":
                page = get_field(newsindex_path, "page")
                pageindex_path = cache_path + "supp/" + supp_bucket + page + "/newsindex.xml"
                if os.path.exists(pageindex_path):
                    return get_field(pageindex_path, "issue")

    return "-1"

def write_buffer():
    global record_buffer, file_out, line_count
    column = 0

    # get field data
    field_data = {}
    for field in fields:
        match = re.search("<" + field +">([^>]*)</" + field + ">", record_buffer)

        if match:
            data = re.sub('[\r\n\t]', '', match.group(1))
            field_data[field] = data
        else:
            field_data[field] = ""

        column += 1

    # check for a dmrecord
    if field_data['dmrecord'] == "":
```

```
        record_buffer = ""
        return

    # write out line
    column = 0
    for field in fields:
        if column > 0:
            file_out.write("\t")
        file_out.write(field_data[field])

        column += 1

    # try to find the parent object
    find_data = field_data['find']
    find_match = re.search("\.cpd$", find_data)

    # make sure it's not a cpd record
    has_parent = "-1"
    if not find_match:
        has_parent = get_parent(field_data['dmrecord'])
    file_out.write("\t" + has_parent)

    # end line
    file_out.write("\n")

    record_buffer = ""

# check arguments
if len(sys.argv) < 2:
    print("Usage: ./desc2table.py col_path [extra_fields]")
    exit(1)

# get filenames
col_path = sys.argv[1]
cache_path = "col" + col_path + "/"
file_xml = cache_path + "desc.all"
file_xml_new = cache_path + "desc.all.new"
if os.path.isfile(file_xml_new):
    file_xml = file_xml_new

# add extra fields
if len(sys.argv) > 2:
    for field in sys.argv[2 :]:
        fields.append(field)

# open files for read/write
file_in = open(file_xml, 'r', encoding='utf-8')
file_out = open(file_xml + '.tsv', 'w', encoding='utf-8')

# write header
column = 0
for field in fields:
    if column > 0:
        file_out.write("\t")
    file_out.write(field)
    column += 1

file_out.write("\tparent")
file_out.write("\n")

# build supp bucket list
supps = glob.glob(cache_path + "supp/D*")
supp_buckets = ['']
for supp in supps:
```

```
    supp_buckets.append(os.path.basename(supp) + "/")

# parse desc.all
line_count = 0
record_buffer = ""
for line in file_in:
    line_count += 1
    record_buffer += line

    # check end of record
    tag_match = re.search('<dmrecord>', line)
    if tag_match:
        write_buffer()

# write out last record buffer
write_buffer()

file_in.close()
file_out.close()
```

Appendix A - The desc2table.py python script. This converts a few key fields from desc.all into a tabs delimited file and determines the parent id number for each record.