Report for the University of Victoria Libraries on Fedora Commons-Based DAMS: Building Collaborative Scholarship Environments, A Test Case

J. Matthew Huculak, Ph.D.

Braydon Justice, BSc

April 2016

Dr. J. Matthew Huculak is Postdoctoral Fellow of Digital Scholarship at the University of Victoria Libraries

Braydon Justice is Digital Scholarship Developer & Analyst at the University of Victoria Libraries

Table of Contents

Abstract

This is a report on two Fedora Commons-Based Digital Asset Management Systems (FCDAMS)—Project Hydra and Islandora—as potential collaborative digital scholarship platforms for scholarly communication at the University of Victoria. The authors of the report tested each system for adaptability and agility by incorporating tools used by the digital humanities community in order to imagine an editing and publication platform hosted by the University Library. The report finds that Project Hydra is most advanced in terms of being compatible with Fedora 4, and it is highly extensible. Project Hydra is widely adopted and supported by libraries in the United States and Canada. The report also finds that Islandora—also widely implemented—is undergoing a massive rewrite in order to be compatible with Fedora 4, and thus it is not currently a prime candidate for implementation.

Introduction

At the 2016 Implementing New Knowledge Environments (INKE) Conference—New Knowledge Models: Sustaining Partnerships to Transform Scholarly Production—Lisa Goddard[1] presented "Developing the Read/Write Library." Her paper outlined how library-hosted Digital Asset Management Systems (DAMS) could "act as research environments that allow faculty and students to help build rich digital collections as part of the research process" (Goddard, 2016). The presentation outlined ways in which the university library could develop and host collaborative technical infrastructures to aid scholarly research and communication in the 21st century without "eroding [the library's] position as trusted providers" of information.

Goddard calls on university libraries to not only provide trusted information to researchers in students, but also to actively collaborate in the development of shared tools that will support digital scholarship (Goddard, 2016). Currently, many libraries are "read only," in that in the scholarly communication lifecycle, libraries are places where researchers find information through library catalogues and other finding aids. But, with the rise of digital scholarship, researchers are routinely creating resources that could potentially feed back into the scholarly information lifecycle managed by libraries. The read/write library asks libraries to think beyond the catalogue and other library-specific tools by providing ways for scholars to "write" back to a common digital repository of information used by both researchers and librarians.

The potential for such collaboration would benefit both librarians and researchers and could provide new ways of engaging with the scholarly communication lifecycle. A library-curated DAMS would give researchers a way to create rich collections for knowledge mobilization and public impact with ready-made, long-term preservation plans, which are required by most large-scale granting agencies in Canada, the United States, and the United Kingdom. Robust FCDAMS like Project Hydra and Islandora allow for shared, core digital repositories from which multiple items/resources can be designed and presented as exhibitions or individual scholarly websites. Though each site would have the appearance of being a "unique" website, it is actually drawing from—and contributing to—a larger, common library system.

The benefit of such a system is that libraries would be able to leverage the expertise of content experts for creating/enhancing description and the discovery of resources hosted at the institution. Moreover, it gives librarians an opportunity to be involved with the entire scholarly

[1] Associate University Librarian, Digital Scholarship and Strategy at the University of Victoria

communication planning process so that a data management plan can be articulated long before a project gets underway.

Problem

Years of bespoke software development have left libraries and institutions with siloed digital scholarship tools that do not advance the field of shared resources and cultures of the library. Moreover, many institutions have relied on vendor-provided software to meet many of the needs of the community. Often, this software is closed and creates walled-gardens of curated content. Since these systems are developed offsite, librarians must manipulate their content/vision to fit the readymade systems rather than being able to adapt those systems to the library's own needs. Relying on closed systems for development does not represent the core values of libraries, which have a long tradition of information sharing and collaboration. The same is true of the Digital Humanities community, which shares common core values with librarians such as its focus on shared tool development, interdisciplinarity, and collaboration. Both communities are looking for ways of sharing the costs of digital scholarship development while maximizing the resources already available in the community.

Years of bespoke software development have left libraries and institutions with siloed digital scholarship tools that do not advance the field of shared resources and cultures of the library.

In recent years, great strides have been made in the development of shared software for collaborative, distributed tool development through platforms like Github. Github has been adopted as the major code-distribution and development platform since it allows a community to share a core codebase with the larger world. This code can be "forked" and developed by a different community. Should the changes made by that different community be beneficial to the whole, the original codebase can incorporate the updated code for the benefit of all. Librarians and digital humanists have adopted Github as a way to foster shared resources for digital scholarship since it allows them to express core values of collaboration in software development.

The University of Victoria

The University of Victoria is in a unique position to contribute to the larger international developments of shared digital scholarship library resources. UVic is home to some of the biggest DH projects in Canada. It hosts the largest Digital Humanities training program in North America: The Digital Humanities Summer Institute (DHSI) headed by Ray Siemens in the Electronic Textual Cultures Lab (ETCL). This institute brings over 800 participants to Victoria each summer for a two weeks of DH training, meetings, and discussions. The Map of Early Modern London (MoEML) is run by Janelle Jenstad in the Humanities Computing and Media Centre (HCMC). Stephen Ross runs Linked Modernisms, and is co-Director with Matt Huculak on OpenModernisms, a free online coursepack builder, and the Modernist Versions Project. These are just a few of the many digital scholarship projects in the humanities across campus.

The University of Victoria Libraries have responded to Victoria's place in the international digital scholarship community. "Objective Three" of the *Operational Plan* states that a goal of the library is to "[c]reate, steward and navigate interdisciplinary, collaborative research environments to bring together content and technology with faculty, student, staff and community expertise to enable knowledge creation and extraordinary integration of research" (Bengtson et al., 2015). Integrated research environments require thinking about both the technological and social changes necessary to bring community members, librarians, researchers, and students together under one umbrella. That is, the problem of creating collaborative research environments is both social *and* technical. The library's recent hire of Lisa Goddard as Associate University Librarian, Digital Scholarship & Strategy, combined with the already existing knowledge base, gives the University of Victoria the necessary technical know-how to be a leader in collaborative tool development. By developing these tools, the University of Victoria library will position itself as a stakeholder in an international network of collaborators building read/write library resources. This will allow libraries to capture and preserve data already being produced by humanities and social sciences researchers who currently operate in siloed environments that are at risk for data corruption or data loss.

The University of Victoria Libraries have responded to Victoria's place in the international digital scholarship

The Needs of Digital Scholarship in the Library

The needs of the read/write library have five primary components:

1. The development of shared, distributed technical frameworks that allow for collaborative environments that serve the needs of both librarians and other researchers on campus
2. The implementation of preservation models developed by librarians for long-term storage of data produced by researchers
3. The ability to import and export library-and-researcher-generated data into a common repository
4. The ability to expose data either in the catalogue or individual projects that draw from a common repository but allow for unique front ends
5. The development of secure, library-grade repositories that are independent of asset management software; the pearls of data should be remain independent from their shells.

Solution: Fedora Commons based Digital Asset Management Systems (DAMS)

The past ten years have seen massive growth in the development of shared, collaborative environments for digital scholarship based on the Fedora Commons repository. Fedora Commons

> is a robust, modular, open source repository system for the management and dissemination of digital content. It is especially suited for digital libraries and archives, both for access and preservation. It is also used to provide specialized access to very large and complex digital collections of historic and cultural

> materials as well as scientific data. Fedora has a worldwide installed user base that includes academic and cultural heritage organizations, universities, research institutions, university libraries, national libraries, and government agencies. ("About Fedora," n.d.)

Originally developed with funding by DARPA, Fedora Commons has become the foundation for many trusted digital repositories around the world. The basic principle of its design is that "It is a modular architecture built on the principle that interoperability and extensibility are best achieved by the integration of data, interfaces, and mechanisms (i.e., executable programs) as clearly defined modules" ("Fedora Commons," 2015). That is, the core information stored in the repository remains intact and secure and is only accessed through modular applications built around it. The most recent version of Fedora, Fedora 4, includes a triplestore for linked data management.[2]

Fedora Commons has become the foundation for many trusted digital repositories around the world

Fedora's Object Model

Fedora arranges information around a "compound digital object" design. This model "aggregates one or more content items into the same digital object. Content items can be of any format and can either be stored locally in the repository, or stored externally and just referenced by the digital object" ("Fedora Digital Object Model - Fedora 3.4 Documentation - DuraSpace Wiki," n.d.). The "basic components" of the object model consist of three core items:

1. A PID (persistent unique identifier);
2. The Object Properties (system-defined properties for object management);
3. Datastreams (content items associated with the PID). (see fig. 1).

[2] For a full list of features in Fedora 4, visit https://wiki.duraspace.org/display/FF/Fedora+4.0+Feature+Set

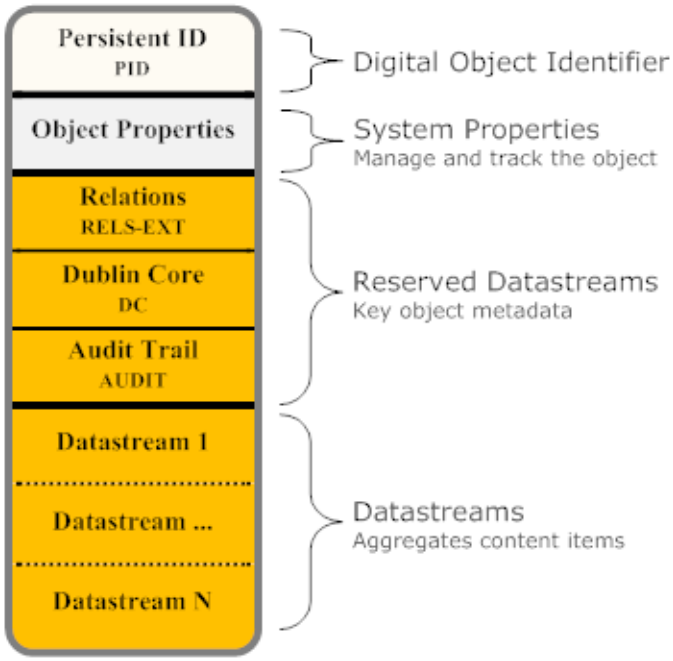


Figure 1: Fedora Commons Basic Digital Object Model

The Key features of Fedora

- Store all types of content and associated metadata
  - Digital content of any type can be managed and maintained
  - Metadata about content in any format (e.g. RDF, XML) can be managed and maintained
- Scale to millions of objects of any size
- Access services via RESTful APIs
- Model content using linked data best-practices
- Store content in local or external filesystems
- Index content to an external search index (e.g. Solr)
- Index content to an external triplestore (e.g. Fuseki, Sesame)
- Trigger workflows to external services using JMS messages
- Manage authorization using a pluggable framework
- Create and manage versions for any repository content
- Conduct routine fixity checks on repository content
- Maintain a complete audit history for all repository content
- Leverage transactions for consistency and performance improvements
- Set up a cluster for high-availability and read performance
- Easily deploy the WAR file into a servlet container
- Disseminate metadata using an OAI-PMH Provider Service
- Choose from multiple, customer driven front-ends, including:
  - Hydra
  - Islandora ("Features," n.d.)

In summary, Fedora Commons provides a robust, librarian-friendly core repository that guards digital information in a secure environment, but it does not provide a user interface to manage objects within the repository. The benefit of having a core repository that operates independently from a user interface is that the "core" data—information that needs to be preserved—exists independently from the software that manages that information. Ostensibly, users could put any shell they wanted on top of the repository and are not bound to one vendor, one system, or one way of managing data. Two major open projects have been developed by librarians to create user-friendly interfaces for Fedora

Fedora Commons provides a robust, librarian-friendly core repository that guards digital information in a secure environment

Commons: Project Hydra and Islandora. These systems will be referred to as "Fedora Commons-Based Digital Asset Management Systems" (FCDAMS) in this paper.

Hydra and Islandora and Collaborative Building

Using Fedora Commons as a base, Hydra and Islandora have created management, discovery, and delivery platforms for the Fedora repository. Both systems are supported by a large network of developers and users across the world. The benefit of Fedora Commons' architecture is that the base repository remains the same (the data intact) even if the management layer—Hydra and/or Islandora—changes. Hydra has built its management layer using Ruby on Rails, while Islandora utilizes Drupal. Moreover, these extensible, modular systems can be adapted to meet the local needs of both librarians and researchers.

In "Toward a Notion of the Archive of the Future: Impressions of Practice by Librarians, Archivists, and Digital Humanities Scholars," Tanya Clement et al. note that these open source technologies encourage software building that that is adaptable to "different institutional settings" (Clement, Hagenmaier, & Knies, 2013). As more and more institutions create software to address their unique demands, the open repository systems become more robust with more and more features. One such area of development is in the scholarly communication lifecycle.

In a FCDAMS, each new layer of data—be it an annotation, a metadata description, a student annotation—is added to the PID as a new datastream. These datastreams can be made visible or invisible to a viewer while still being protected in the repository. That is, rather than being mere static objects, an object in a Fedora Commons repository is a pearl with the potential to grow as users add information to the object: each new information layer allows the pearl to grow larger while protecting the original object at the core of the data model. This model allows the read/write library to be scalable, secure, accessible, and modifiable.

FCDAMS also allow us to think about digital *communication* as well as *preservation* within the same system—certain data layers are used specifically for *preservation*, while other data layers can be used for *presentation*. Moreover, the system can be adapted to offer tool layers for digital scholarship, which in turn will produce more data that can be stored as a datastream around a PID. This is a shift in library tool development in which preservation, presentation, and scholarly activity all occur within the same system—and the data produced for each activity enriches the others. This is an integrated scholarly environment at the core of the read/write library.

FCDAMS also allow us to think about digital communication *as well as* preservation *within the same system*

Testing the Read/Write Library

To test the feasibility of implementing such an environment, Lisa Goddard assigned Dr. Matt Huculak and Braydon Justice to test the two common FCDAMS that could potentially be modified for preservation, presentation, and scholarly activity. There are two purposes of our study: 1. To determine how easy it is to install and maintain the Project Hydra and Islandora platforms on a library server; 2. To determine how easy it is to develop new tools for both platforms. That is, what platform would be the most agreeable to maintain and which platform would allow us the greatest flexibility to meet the local needs of digital scholars at the university.

Tool Development for Each Platform

Ostensibly, the number of tools for digital scholarship that can plugged into a FCDAMS are copious and can be developed based on local needs. Thus, for the purposes of this study, Huculak and Justice decided to focus on a few simple DS tools that might be used in a Read/Write FCDAMS housing material from a digitization lab. Many libraries are currently digitizing books and other book-like material to make those holdings accessible to a larger community. For the purposes of this study, we made two basic assumptions:

1. That the library has a digitization lab
2. That the digitization lab is producing page images or PDFs of books or other bound media that would then be ingested into a FCDAMS

We imagined a situation where scholars would be producing book-like objects based on holdings in the university library. In order to accomplish this, we needed two workflows: 1. A ingest workflow, and 2., an editing workflow.

The ingest workflow would determine how the page objects would be ingested into the FCDAMS; the editing workflow would determine how a researcher/reader might interact with the object within the FCDAMS.

Ingest workflow

For the ingest workflow, we wanted to create a situation where a user would upload a folder containing multiple page images of a book. In our testing, we used a dissertation comprising 236 pages. On ingest, we wanted the FCDAMS to perform two tasks:

1. Paginate the files so they were properly ordered
2. Perform OCR on those page images so that machine-readable text would be automatically produced and associated with the corresponding page object.

Optical Character Recognition

An important tool for digital scholarship is the computational analysis of textual records—or what digital humanist refer to "distant reading" and "macroanalysis" based on computer-readable texts. Moreover, computer readable texts are used for Stylometrics, a method used to determine authorship and the authenticity of texts by the statistical analysis of author word choices. A popular example of this is the stylometric program that helped "reveal" J. K. Rowling as the real author of *The Cuckoo's Calling*, written under the pen name Robert Galbraith (Juola, 2013). In terms of literary scholarship, stylometrics can help editors determine who did what in co-authored works; this is important, for example, in assigning authorship in Shakespeare's collaborations with other playwrights.

For libraries, computer readable texts can be searched for keywords or text strings that match a user's queries in a discovery tool. Rather than relying solely on manually entered metadata, full-text searches are a powerful way of locating information in a vast collection of objects—but only if that text is associated with the proper page image. Thus, we needed to test that each FCDAMS could produce, associate, and organize additional datastreams in a given page object.

Pagination

An essential part of creating reading page objects is to maintain the sequential order of the images. If a 236-page PDF is ingested into a FCDAMS, the system must be able to parse,

store, and retrieve that material in its proper order. This becomes even more complicated should other datastreams be created alongside the page objects. If a text file is created after the OCR process, it, too, must be displayed and exported in the proper order.

Editing Workflow

Once the page images were ingested, paginated, and OCR'd, we wanted to create a system in which a user could

1. Correct the automated OCR performed on ingest;
2. Display the page images in a book viewer;
3. Annotate the page images should there be illustrations, paratexts, or other features that would not be renderable by OCR.

For example, a medieval manuscript might have a stylized letter that is both an animal and a letter of the alphabet. An image markup tool is needed to describe such a feature using an open annotation system.

Any information created using the FCDAMS shell—whether it be an OCR correction or an image annotation—would have to be stored as a new layer, or datastream, associated with the PID within the Fedora Commons Repository. This is the working model we adopted in testing Hydra and Islandora.

Project Hydra

Project Hydra's "ultimate objective is to produce a community-sourced, sustainable application framework that provides rich and robust repository-powered solutions as an integrated part of an overall digital content management architecture" (University of Virginia Libraries 2013).

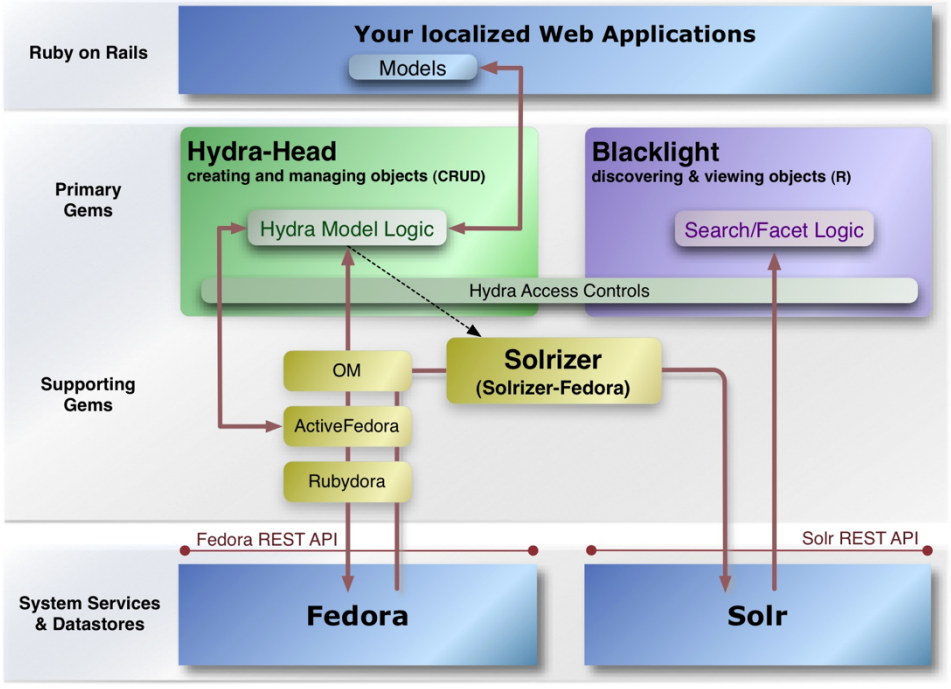


Figure 2: Hydra Architecture (from website)

From the Hydra website:

The primary components of the Hydra technical framework are:

- ***Fedora***, providing a robust, durable repository layer for persisting and managing digital objects. Fedora's disseminator features allow us to place an abstraction layer between it and our Hydra heads,

shielding an institution's applications from any future changes to the repository structure.

- ***Solr*** indexes, providing fast access to information about the institution's resources. Solr can be used as a lingua franca: content from any source that can generate a Solr index (perhaps an OPAC, or repository metadata records with different schema) can potentially be brought into a Hydra discovery environment.
- ***Blacklight***, a Ruby on Rails gem that provides faceted searching, browsing and tailored views of objects
- ***HydraHead***, a Ruby on Rails gem that works with ***ActiveFedora*** to provide create, update and delete actions against objects in the repository, as well as to support various content management actions (e.g., upload file, edit metadata, change permissions)
- A suite of web-based services, supporting granular actions against content to support their management, access and preservation (e.g., checksumming, indexing, transform MARC to MODS, djatoka-based JPEG2000 image streaming)

Finally, these components rely on several background services:

- *authorization*, provided by FESL (Fedora Enhanced Security Layer - a new Fedora framework service part funded by the Hydra partners and others in the community)
- *authentication*, provided by local institutional systems
- *workflow*, which can either be provided as a bundled part of the Hydra framework, or provided by a local institutional systems. ("Technical Framework and its Parts - Hydra - DuraSpace Wiki," n.d.)

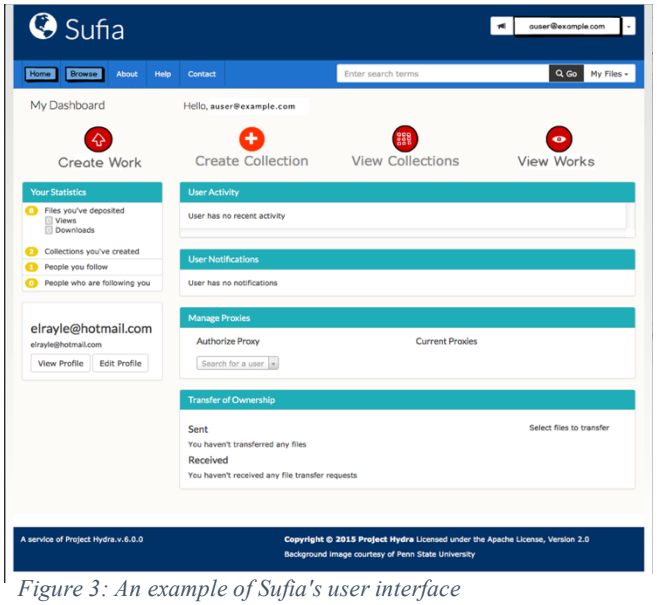


Figure 3: An example of Sufia's user interface

Project Sufia

The core architecture of Hydra provides the foundations on which to build larger systems and interfaces—or what the community calls "heads." The most widely adopted Hydra head is Sufia, which runs on top of Fedora 4. Sufia "is an extensible, out of the box, self-deposit repository" using a user-friendly interface" ("DuraSpace | Open technologies for durable digital content," n.d.). When installing Hydra, users may choose to do a custom installation of the base Hydra software stack, or they may choose to install a Hydra "head" like Sufia. A bare hydra head is basically useless for the purposes of digital scholarship: there is no base mechanisms for

uploading or displaying objects or even setting up user roles—one must build all that one's self.

Huculak and Justice researched various flavours of Hydra, and Justice determined that Sufia was the only Hydra head that could be installed on its own and with little modification. Many of the other Hydra implementations are managed by private companies (such as Avalon) that do installations on pay-per-install basis. Sufia provides users with an open, "out-of-the-box" solution designed by librarians for digital asset management (Coyne, 2015).

Installation Notes

Justice, a recent graduate of Computer Science with a focus on communications and networks, and a working knowledge of Ruby, installed Sufia in four hours. The installation was seamless and we did not encounter any problems.

Software Requirements for Sufia

Sufia requires the following software tools to work:

1. Solr (http://lucene.apache.org/solr/): For indexing and faceted searching
2. Fedora Commons (http://fedorarepository.org/): Digital repository
3. An SQL Relational Database Management System (MySQL, PostgreSQL): [Note that SQLite will be used by default if you're looking to get up and running quickly]
4. Redis (http://redis.io/): A key-value store
5. ImageMagick with JPEG-2000 support (http://www.imagemagick.org/script/index.php): Image editor
6. FITS version 0.6.x (http://projects.iq.harvard.edu/fits/downloads#fits): "FITS is a free and open source tool for identifying and validating file formats, extracting metadata embedded within files, and outputting the metadata in various formats."
7. LibreOffice (version 7) (https://www.libreoffice.org/download/libreoffice-fresh/): Office Suite. (if you encounter a problem, see https://Github.com/projecthydra/sufia#derivatives).

In order to expedite the installation of these tools for any future installations, and to share UVic's work with the wider library community, Justice's chef script can be downloaded here: https://Github.com/UVicLibrary/sufia_chef

Notes on the Installation Process & Documentation

Sufia proved to be notably easier to install than Islandora; one reason for the ease of installation is that one is installing fewer parts than Islandora, which comes with many more "out-of-the-box" tools. Sufia also has fewer documentation resources than Islandora. Most of the documentation for the system is on the Github site; However, Justice found that even though there was less documentation, the installation process was smoother for Sufia. For example, if Sufia said it needed version 2.2.0 of Fedora, the installation would still work with version 2.2.4. This was not the case with Islandora, which needed exact versions and specifications.

There was also a slight learning curve for Justice to completely understand the Ruby environment, which is essential when is installing the system. In the end, Sufia was the least difficult system to install because it has fewer "moving parts" to address on installation.

Out-of-the-Box Tools

Once Sufia is installed on the server, users have the following features available to them:

1. User and Role Management
2. The ability to upload and download objects
3. The ability to create collections
4. Automatic derivative creation
5. Manual metadata input[3]

Justice found that the initial Sufia install provided a clear and stable base on which to start our own development of new tools. We began our testing by looking for an OCR gem that would work with our installation and would execute on file upload.

Uploading Objects & Derivative Creation

The ingest procedure in Sufia was superior to that in Islandora—and this was clearly seen when problems were encountered in the upload process. We started by batch uploading at 236-page PDF. In Sufia, if the system encountered a problem, all of the material ingested up-to-the-point of the problem file was saved and readily available within the repository. This was not the case in Islandora, which, when it encountered a problem, failed to assign unique identifiers to the objects that were uploaded before the problem occurred. Thus, if Islandora spent 6 hours ingesting a 400-page document, and the process failed on object 399, the entire upload was lost. When we started adding functions to the ingest process, including automatic OCR, the robustness of Sufia's file handling proved invaluable.

Optical Character Recognition (OCR)

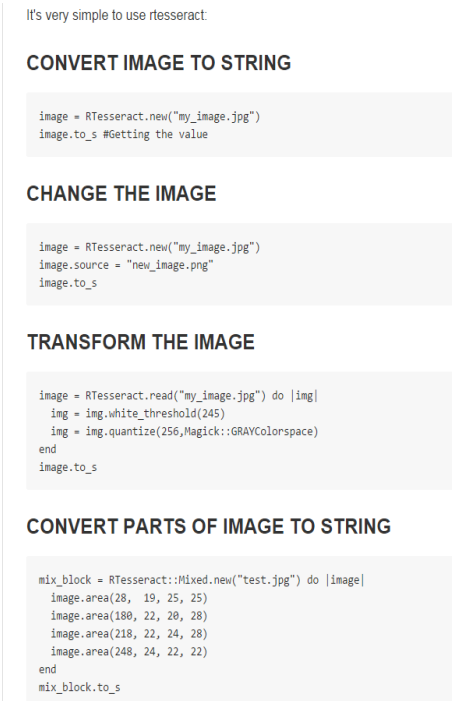
We chose to work with Tesseract (https://Github.com/tesseract-ocr), which is "an Open Source OCR engine, available under the Apache 2.0 license. It can be used directly, or (for programmers) using an API. It supports a wide variety of languages" ("tesseract-ocr/tesseract," n.d.).

We found a few gems that were supposed to integrate Tesseract into Ruby (Sufia is written in Ruby), but the gems did not work. After spending a day working with the Gems, Justice wrote his own code in a matter of hours. Since we were adding a new datastream to the page object (a text file), Justice created a PAGE OBJECT within Sufia to hold image files and text files of a given page scan. Using the Tesseract API, Justice had Ruby perform a system command that initiated Tesseract to read and extract text from the page image associated with the page object in Sufia. Once Tesseract created the new text file, it was saved along with the page image in PAGE OBJECT file in Sufia. Each scan was then assigned a unique page number. It was a straightforward process that was integrated into the base Sufia install with some minor tweaking.[4] Tesseract is the dominant open OCR engine on the market, but it was hard to integrate it with Ruby on Rails. There are RUBY implementations of Tesseract, including

[3] In the University of Victoria fork of Sufia, Justice created a way to batch upload metadata from ContentDM to Sufia using JSON (https://Github.com/UVicLibrary/Sufia-Head) However, this code is specific to the University of Victoria's installation of ContentDM that it might not work for all collections outside our institution.

[4] OCR is one of the areas in which Islandora excelled—more so than Sufia. It is clear developers have taken the time to implement it properly in the Islandora framework.

rTesseract, but we were unable to get it to work in Sufia. Therefore, Justice wrote his own code and was able to easily hook Tesseract into the Sufia head (code below).



It's very simple to use rtesseract:

CONVERT IMAGE TO STRING

```
image = RTesseract.new("my_image.jpg")
image.to_s #Getting the value
```

CHANGE THE IMAGE

```
image = RTesseract.new("my_image.jpg")
image.source = "new_image.png"
image.to_s
```

TRANSFORM THE IMAGE

```
image = RTesseract.read("my_image.jpg") do |img|
  img = img.white_threshold(245)
  img = img.quantize(256,Magick::GRAYColorspace)
end
image.to_s
```

CONVERT PARTS OF IMAGE TO STRING

```
mix_block = RTesseract::Mixed.new("test.jpg") do |image|
  image.area(28,  19, 25, 25)
  image.area(180, 22, 20, 28)
  image.area(218, 22, 24, 28)
  image.area(248, 24, 22, 22)
end
mix_block.to_s
```

```
def makePage(file, i, textPage, logger)
    index = i+1
    jpg = file.id+index.to_s+".jpg"
    txtpg = file.id+index.to_s+".txt"
    filename = file.id+File.extname(file.filename.first)
    data = ActiveFedora.fedora.connection.get(file.content.uri.value).body
    File.open("/tmp/"+filename, "wb") {|f| f.write(data)}
    logger.info "Page: "+index.to_s
    ext = File.extname(file.filename.first)
    logger.info "starting conversion"
    logger.debug "convert -density 300 /tmp/"+file.id+ext+ (ext==".pdf" ?
    system("convert -density 300 /tmp/"+file.id+ext+ (ext==".pdf" ? "["+i.
    logger.info "Conversion Complete"
    if (textPage==nil)
        logger.info "starting Tesseract"
        system("tesseract /tmp/"+jpg+" /tmp/"+file.id+index.to_s)
        logger.info "Tesseract Complete"
    else
        logger.info "writing: /tmp/"+txtpg
        File.open("/tmp/"+txtpg, "w+") { |f| f.write(textPage) }
    end
    page = Page.new(file.id + index.to_s)
    page.number = index
    logger.info "loading text file"
    page.full_text.content = File.open("/tmp/"+txtpg)
```

Scale of programming difficulty from 1 to 5: 2

Pagination

The base install of Sufia does not have automatic pagination capabilities, and we imagine that some users of our system might want to ingest multi-page documents into Sufia. Given that we wanted to create a scholarly environment where users could edit individual pages of text, we had to devise a way to paginate the files ingested into Sufia so that we could properly order the various datastreams associated with page objects. One-page object might have multiple sub-objects within it (for example, a page object might contain a 600dpi page image file, a 72dpi page image file, and the text file of a given page scan), so we needed a way to properly order those pages and datastreams when uploaded or created.

We imagined a scenario where a user might upload a multipage PDF into Sufia (since some departments have only kept PDFs as archival sources); we used Huculak's 250-page dissertation as a test file; we used FITS (part of the base install of Hydra) to pull the metadata properties from the PDF document.

However, we discovered that the information FITS acquired did not allow us to paginate the file properly since the PDF metadata was not consistent. Justice changed his script so that instead of asking the PDF for its information, he instructed the FITS to "Get pages.all. max," which gave us the total number of individual pages in the document.

Justice created a pagination script so that the PDF was disassembled into individual pages that were then linked within the repository. The pages could then be served either individually or as a whole document.

To test the OCR engine, we performed full OCR on the PDF document in order to test the pagination program since it was disassembling each page and associating a new text files once OCR was performed on it.[5] We also tested extracting the text already embedded in the PDF document. In both cases, Justice was able to paginate the document and associate the new datastreams with each file.

Problems/Issues to consider:

The processing resources needed to go through the OCR and pagination were considerable. Thus, if a library were uploading documents and serving documents at the same time, there would be a noticeable server slowdown that would affect user experiences. To address this problem, Justice created a "Command Run On Notice" (CRON) to do the heavy processing overnight when there would be less demand on the server system. The CRON creates page files and then associates page files to the overall main object.

All of the pagination code created are freely available at: https://Github.com/UVicLibrary/Sufia-Head

```
page = Page.new(file.id + index.to_s)
page.number = index
logger.info "loading text file"
page.full_text.content = File.open("/tmp/"+txtpg)
logger.info "loading content file"
page.pageContent.content = File.open("/tmp/"+jpg)
page.pageContent.mime_type = "image/jpeg"
page.pageContent.original_name = jpg
logger.info "associating file"
page.associate(file)
File.delete("/tmp/"+filename)
File.delete("/tmp/"+jpg)
File.delete("/tmp/"+txtpg)
```

Scale of programming difficulty from 1 to 5: 2

Text Editing: CKEditor

Huculak and Justice imagined a situation in which users would want to either 1. correct the automated OCR text, or 2. Hand transcribe text associated with a given image. Justice recommended that we use CKEditor (http://ckeditor.com/) to produce an "editing window" that would sit next to the page image in Sufia. A scholar could then modify the text file produced by the automatic OCR, creating a text datastream in the object. Justice was able to tie CKEditor into Sufia with little effort. Each edit we made was versioned within the digital object so changes could be tracked or restored (see fig. 4).

Scale of programming difficulty from 1 to 5: 3

[5] In order to perform new OCR on each page, Justice used ImageMagick (http://www.imagemagick.org/script/index.php) to convert every page of the PDF into an image file.

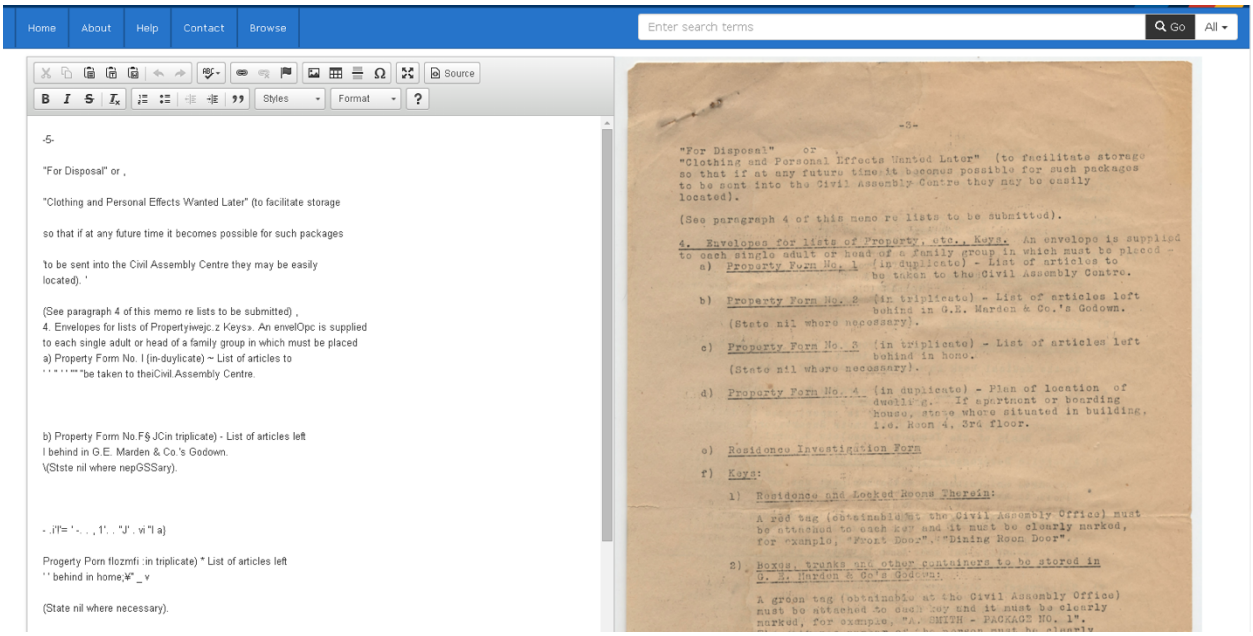


Figure 4: CKEditor running in Sufia

Book Display: Internet Archive Viewer

Since Islandora uses the Internet Archive Viewer to display pages, we wanted to see how difficult it would be to tie it into Sufia as a page viewer. Justice looked at the code for the Internet Archive Viewer (IAV), and found that it is a monolithic file of 5000 lines of Javascript.

The primary weakness we found with this file is that all of the code was located in one file (it is clear this software was developed before Github). Current coding practices dictate that you separate different functions into different files; for example, there would be an animation file that contained all of the functions for page turning; there would be file that takes care of all of the controls, and a file for the actual rendering of the image. But in the IAV, code is added to code, and it is now difficult to decipher what lines of code do what.

The other issue we faced with the IAV was that the commenting on the code contained both programmer comments to one another as well as deprecated code—rather than removing the old code, someone just commented it out. In the era of Github, one would simply delete code that was no longer in use since one can easily return to an earlier version of the code in a versioning system.

```
91
92      // Zoom levels
93      // $$$ provide finer grained zooming
94      /*
95      this.reductionFactors = [ {reduce: 0.5, autofit: null},
96                                {reduce: 1, autofit: null},
97                                {reduce: 2, autofit: null},
98                                {reduce: 4, autofit: null},
99                                {reduce: 8, autofit: null},
100                               {reduce: 16, autofit: null} ];
101     */
102     /* The autofit code ensures that fit to width and fit to height will be available */
103     this.reductionFactors = [ {reduce: 0.5, autofit: null},
104                           {reduce: 1, autofit: null},
105                           {reduce: 2, autofit: null},
106                           {reduce: 3, autofit: null},
107                           {reduce: 4, autofit: null},
108                           {reduce: 6, autofit: null} ];
109
```

Figure 5: The bottom commenting is helpful; the top commenting is unneeded, making the file harder to read

```
1594    /* Flip areas no longer used
1595    this.twoPage.leftFlipArea = document.createElement('div');
1596    this.twoPage.leftFlipArea.className = 'BRfliparea';
1597    $(this.twoPage.leftFlipArea).attr('id', 'BRleftflip').css({
1598        border: '0',
1599        width:  this.twoPageFlipAreaWidth() + 'px',
1600        height: this.twoPageFlipAreaHeight() + 'px',
1601        position: 'absolute',
1602        left:   this.twoPageLeftFlipAreaLeft() + 'px',
1603        top:    this.twoPageFlipAreaTop() + 'px',
1604        cursor: 'w-resize',
1605        zIndex: 100
1606    }).click(function(e) {
1607        self.left();
1608    }).bind('mousedown', function(e) {
1609        e.preventDefault();
1610    }).appendTo('#BRtwopageview');
1611
1612    this.twoPage.rightFlipArea = document.createElement('div');
1613    this.twoPage.rightFlipArea.className = 'BRfliparea';
1614    $(this.twoPage.rightFlipArea).attr('id', 'BRrightflip').css({
1615        border: '0',
1616        width:  this.twoPageFlipAreaWidth() + 'px',
1617        height: this.twoPageFlipAreaHeight() + 'px',
1618        position: 'absolute',
1619        left:   this.twoPageRightFlipAreaLeft() + 'px',
1620        top:    this.twoPageFlipAreaTop() + 'px',
1621        cursor: 'e-resize',
1622        zIndex: 100
1623    }).click(function(e) {
1624        self.right();
1625    }).bind('mousedown', function(e) {
1626        e.preventDefault();
1627    }).appendTo('#BRtwopageview');
1628    */
1629
```

Figure 6: The comment says the code is no longer in use, but the code has been left in, adding confusion. With Github, you would just delete this code because you can go back to an earlier version if need be.

jquery-ui-1.8.1.custom.min.js	Added jquery ui built with only Core and Effects Core	6 years ago
jquery-ui-1.8.5.custom.min.js	Full version of jquery ui	5 years ago

Figure 7: IAV is using an older version of JQuery

Because the IAV file was so large, and because the structure of the file is monolithic, we cannot recommend spending the time incorporating IAV code into Sufia. In short, the code is too unwieldy, and too much time would have to be spent organizing and separating out the code.

IAV Findings:

- IAV uses an older version of jQuery (1.8) from 2010; this version is riddled with vulnerabilities.
- It does not have any type of "mobile friendly" display
- The code is monolithic, meaning it is mostly just one big file. The file is broken up into different functions but there is no helpful order or index, making it very hard to navigate.

- The logging functions from the bookreader are disabled via commenting, meaning that in order to enable debugging and logging you have to search through and un-comment each log line in the code that relates to logging.
- The commenting is confusing due to the fact that old unused code has been left in the file and just commented out, making the file bigger and more confusing.
- There is no .jp2 specific code only a couple commented out log lines referencing the format.

The University would have to invest time and effort in creating a new book viewer, or work with the existing IAV code to produce a more modern code base from which to work.

IIIF: Open Sea Dragon & Mirador

For image viewing and annotation, the emerging standard of image display and markup is the IIIF:

> The International Image Interoperability Framework (IIIF) is a protocol for standardized image retrieval created by a community of the world's leading research libraries, major national libraries and not-for-profit image repositories in an effort to collaboratively produce an interoperable technology and community framework for image delivery" (Pillay, n.d.).

Project Mirador (http://projectmirador.org/) allows users to annotate images on top of the OpenSeaDragon (https://openseadragon.Github.io/) web-based viewer. Although this is talk about integrating Mirador into Hydra ("Yale Hydra Project," 2015), we were unable to find a working gem to install. That said, this type of functionality is under active development within the Hydra community.

Tools Created

All of the tools created by Justice during our testing have been made freely available on the UVic Library Github account (https://Github.com/UVicLibrary). The testing of these tools has already resulted in giving back to the community of digital scholarship.

Future of Hydra

The future of Hydra is bright. It is being supported by a host of institutions including Stanford, Yale, and more importantly, the Digital Public Library of America. The National Endowment for the Humanities is funding the creation of "Hydra-in-a-box," a software stack that will provide a ready-made system like Sufia. For a full list of Hydra partners, see this page: http://projecthydra.org/community-2-2/partners-and-more/
The versatility, modularity, and international support behind Project Hydra suggests that it will be a dominant FCDAMS for years to come.

Islandora

Islandora is a Drupal-based FCDAMS developed by the University of Prince Edward Island. It, too, operates on a modular model. Modules in the Islandora environment are called "solution packs." The Drupal interface of Islandora provides the user-and-object-management

functions of the DAMS, and the system can be modified by adding modular solution packs to the workflow.

Out of the Box Tools

Islandora immediately provides the same functionalities provided by the Drupal CMS. On top of that functionality, it also has the following solution packs freely available[6] (available on Github):

- Islandora Scholar - Enables Institutional repository features such as citations, author pages, etc.
- Google Analytics - Enables support for Google Analytics.
- Collection Solution Pack - Allows Islandora to view and manipulate objects as collections.
- Islandora Importer - Enables batch ingesting functions.
- Islandora Bookmark - Enables users to create lists of objects and save them.
- Islandora OAI - Enables OAI harvesting of Islandora objects.
- Audio Solution Pack - Enables support for WAV and MP3 files.
- Book Solution Pack - Enables creation of book collections and the ingesting of pages.
- Image Solution Pack - Enables support for JPEG, PNG & GIF images.
- Large Image Solution Pack - Enables support for TIFF images.
- PDF Solution Pack - Enables support for PDF documents.
- Video Solution Pack - Enables support for the most popular video formats.
- Islandora Paged Content - Enables support for paged content.
- Islandora Internet Archive Book Viewer - Enables the Internet Archive (archive.org) Book Reader.
- Islandora OCR - Enables Optical Character Recognition support for uploaded images.
- Islandora Open Seadragon Viewer - Enables Open SeaDragon image viewer.
- Islandora JWPlayer Video Viewer - Enables the JWPlayer video player.
- Islandora FITS - Enable support for File Information Tool Set technical metadata standard.
- Islandora Simple Workflow - Enables a simple editorial workflow for Islandora which defines a default "inactive" state for all newly ingested objects.
- Islandora Book Batch - Enables simple batch ingesting support for books.
- Islandora IP Embargo - Enables content embargoing based on IP ranges.
- Compound Solution Pack - Enables grouping of objects into a generic parent-child relationship.
- Newspaper Solution Pack - Adds support for newspaper collections.
- XACML Security Policy Editor - Adds support for simple and complex object and collection level security policies.
- Islandora MARCXML - Enables support for transforming metadata between MODS and MARCXML.
- Islandora Accordion Rotator (Private Repo/OnDemand Only) - Enables usage of an image rotator for the Islandora OnDemand theme.

[6] http://www.discoverygarden.ca/modules-and-solution-packs/

- Islandora SOLR Views - Utilize SOLR search results in Drupal Views.
- Islandora SOLR Metadata - Provides an interface to construct configurations used for displaying metadata on Islandora objects.
- Document Solution Pack - Provides a collection and a content model for user documents which are converted to PDF for display.
- Islandora JOD Converter - Utilizes the OpenOffice/JOD Converter as a service to convert documents between various formats.
- Entities Solution Pack - Provides support for entities such as people, places, events and organizations in Islandora.
- Islandora PLUpload - Supports uploading of large files.
- Islandora Job - Supports asynchronous/parallel job processing via gearman

Digital Scholarship

Huculak worked with Islandora in 2011 to develop a Digital Humanities Solution Pack to work with the system. The Internet Archive Book Viewer, Tesseract OCR, and CWRC Writer were incorporated into the platform for digital scholarship. These functionalities were based on the "Islandlives Workflow" developed for UPEI, in which page images were ingested into the DAMS and automatically edited and OCR'd (see fig. 8).

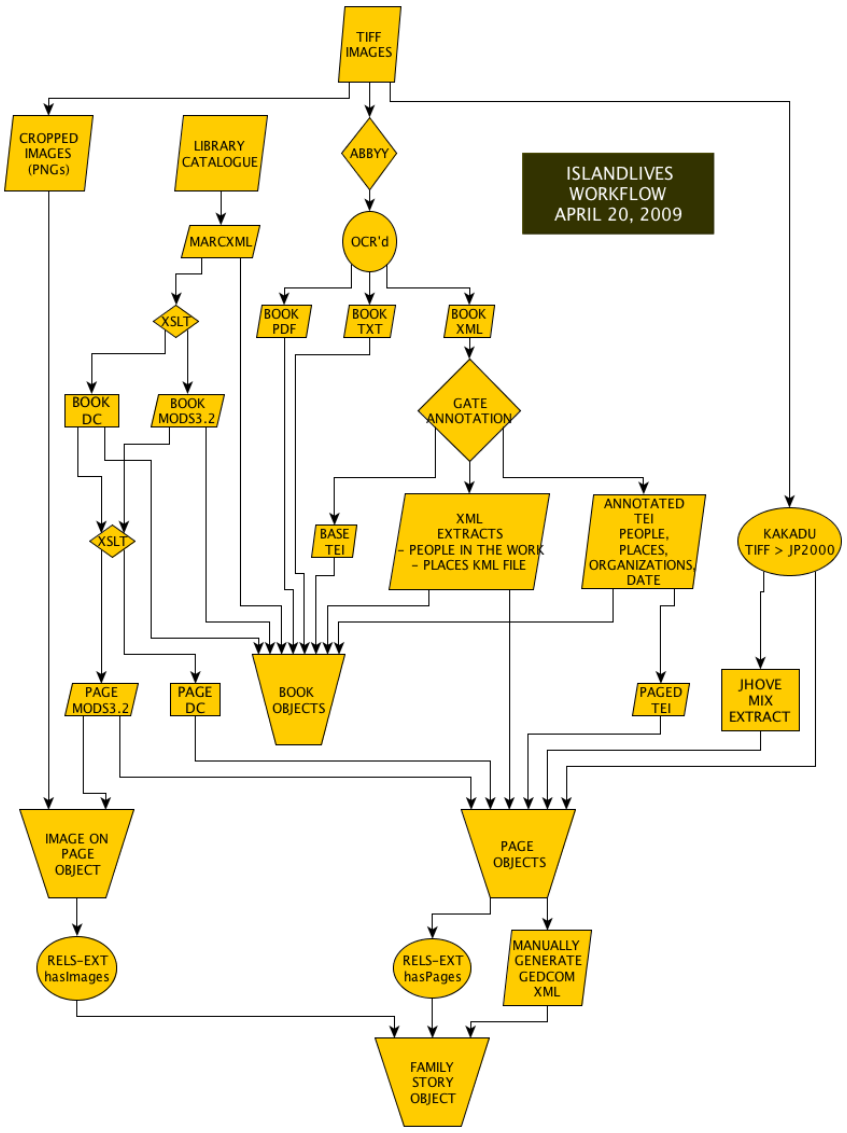


Figure 8: Islandlives Workflow

Installation

In order to install Islandora, one must first install Drupal and Fedora Commons 3, after which the Islandora module and servlet filter will be installed. We installed the system on a server with 8GB RAM and 50GB drive space.

The installation process was riddled with problems. Although Islandora has much more installation documentation on Duraspace than Hydra, we found the instructions were lacking in that they made many assumptions about installation procedures. Moreover, the instructions for installation are not laid out for easy comprehension. Justice found that it was easy to miss important steps in the installation process because of the page layout.

Adding to these problems was that certain software versions that were supposed to have been supported by the system were in fact, not supported. If something did not work, the system did not produce an error log to show where the failure was occurring. Eventually, Justice became so frustrated with the process that he stopped using the instructions on Duraspace and looked for a Chef script to do the work for him.

By using Chef scripts that were modified for installation, Justice was able to see precisely where errors were occurring during installation, which made them easier to correct for the proper installation.

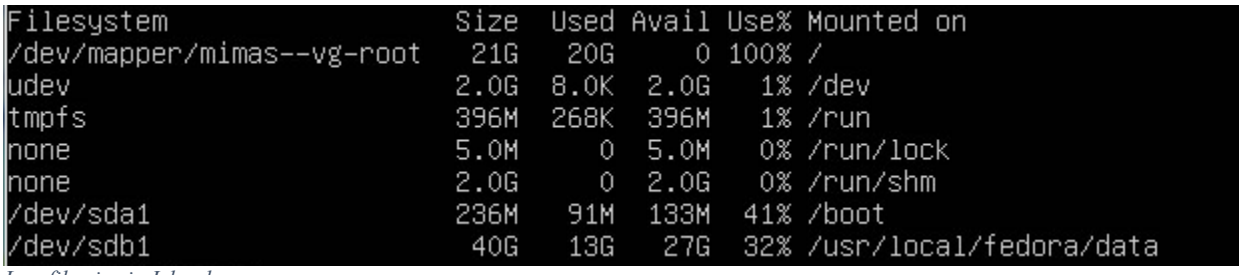
Object Model

The object model for Islandora is similar to Hydra since they use the same underlying Fedora Commons framework:

- Everything in Islandora's repository is an Object
- Objects are made up of Datastreams
- Objects have Relationships to one another
- Objects have Persistent Identifiers (PIDs) that are unique in your repository ("Getting Started with Islandora - Islandora Documentation - DuraSpace Wiki," n.d.)

Uploading Material into the Repository

We also encountered problems when attempting batch uploads into the Islandora repository. In Sufia, if an upload failed, the files that had already been uploaded still appeared in the GUI. In Islandora, however, if an upload failed after 249 pages of a 250-page document, the files that were already ingested were lost in the repository. Justice had to go in to Fedora and manually remove the files. The files were still there, safe in the repository, but the system would not provide links to those files so they could not be accessed through the DAMS. We recommend that Islandora fix this issue in the next release.



Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/mapper/mimas--vg-root	21G	20G	0	100%	/
udev	2.0G	8.0K	2.0G	1%	/dev
tmpfs	396M	268K	396M	1%	/run
none	5.0M	0	5.0M	0%	/run/lock
none	2.0G	0	2.0G	0%	/run/shm
/dev/sda1	236M	91M	133M	41%	/boot
/dev/sdb1	40G	13G	27G	32%	/usr/local/fedora/data

Figure 9: Log file size in Islandora

The other issue we encountered in uploading material to the repository is that the drive would become full because the system created massive log files. The issue is that Islandora creates 300dpi TIFF image for all images that are ingested (even jp2 images that are already compressed); thus, hard drive space is quickly filled because of this. Eventually, these log files became so large that our drive was overwhelmed—the 6MB dissertation we uploaded caused Islandora to create a 21GB log file. We quickly ran out of space and the ingest failed. We attempted to create a max file size command, but the system ignored it.

Why We Stopped Testing: New Software Forthcoming

After encountering a number of issues working in Islandora, we decided to cease testing since we are aware that a new system, based on Fedora 4, is now in production. This new version of Islandora completely rewrites the system, and thus, we believe it would not be useful to compare an older version of Islandora with a newer version of Hydra.

Conclusion:

Since Islandora is undergoing a radical redesign that will make it compatible with Fedora 4, as well as allow greater integration with Drupal plugins, we are unable to make a fair assessment of this software stack. At the time of producing this paper, Hydra Sufia, which utilizes the functions of Fedora 4, provides an extensible, agile, environment on which to build a FCDAMS. However, in order for Hydra to be effective as a FCDAMS, the library must have a Ruby and Rails programmer committed to its development. Once Hydra-in-a-box is released, the system might be more user friendly on install for smaller institutions without programmer support.

Once the new version of Islandora is released, it will provide a useful FCDAMS for those institutions that do not have resources for a fulltime developer.

Appendix

Server Specs

We installed Hydra on Linux server with multiple virtual hosts. We originally allotted 4GB RAM and 20GB of hard drive space with two cores. However, for greater efficiency, we found that we needed a minimum of 10GB RAM, a 20GB hard drive, and four cores for faster multitasking.

Justice notes that if Hydra is creating derivatives of objects ingested into the system on top of serving webpages, it's important to have multiple cores for multitasking. Thus, if Hydra is to act as the primary DAMS for the library, in which employees will be uploading content at the same time as users are accessing content on the web, Technical Services will want to dedicate more cores to the system—otherwise the system will be overwhelmed. If Hydra is only being used occasionally, and there is not sufficient demand on the system in terms of serving content to the web, two cores are satisfactory.

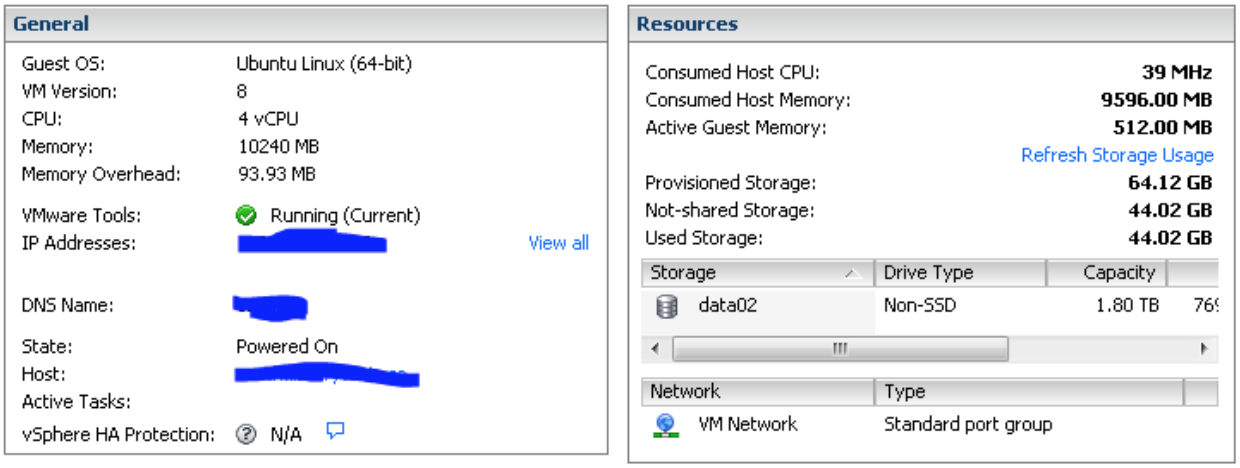


Figure 10: Screenshot of hardware

Hydra Partners[7]

1. Stanford University

[7] http://projecthydra.org/community-2-2/partners-and-more/

2. University of Virginia
3. University of Hull
4. Fedora Commons (now part of DuraSpace)
5. MediaShelf LLC
6. University of Notre Dame
7. Northwestern University
8. Columbia University
9. Penn State University
10. Indiana University
11. London School of Economics and Political Science
12. Rock and Roll Hall of Fame
13. The Royal Library of Denmark
14. WGBH
15. Boston Public Library
16. Duke University
17. Yale University
18. Virginia Tech
19. University of Cincinnati
20. Princeton University Library
21. Cornell University
22. University of Oregon (as Oregon Digital)
23. Oregon State University (as Oregon Digital)
24. Case Western Reserve University
25. Tufts University
26. Duoc UC
27. University of Alberta
28. Digital Public Library of America (DPLA)
29. University of Michigan
30. University of California, San Diego
31. University of York
32. Amherst College
33. Australian Department of the Environment
34. Brown University
35. Chemical Heritage Foundation (CHF), Philadelphia
36. Chinese Historical Society of Southern California
37. Digital Commonwealth (Massachusetts Collections Online) (https://www.digitalcommonwealth.org/)
38. The Digital Repository of Ireland (Trinity College Dublin)
39. Durham University, UK
40. The George Washington University
41. Johns Hopkins University (levysheetmusic.mse.jhu.edu/)
42. Lafayette College
43. Los Alamos National Laboratory (LANL)
44. McGill University
45. Museum of the Performing Arts (MAE) of the Theatre Institute of Barcelona (colleccions.cdmae.cat)

46. National Library of Ireland
47. Northeastern University (http://repository.neu.edu/)
48. Oregon Shakespeare Festival
49. Temple University
50. University of California, Berkeley
51. University of California, Santa Barbara
52. University College Dublin
53. University of Illinois at Urbana-Champaign
54. University of Kentucky
55. University of Oxford
56. University of Washington
57. University of West Virginia

Projects Using Islandora[8]

1. Adventist Digital Library (link is external)
2. American Philosophical Society (link is external)
3. Andrews University
4. Atlantic Climate Adaptation Solutions Association (link is external)
5. Barnard College (link is external)
6. Berklee College of Music Archives (link is external)
7. BESS Digital Archive (link is external)
8. Biblioteca do Ministério da Fazenda no Rio de Janeiro (BMF/RJ)
9. Boston College
10. Botanical Research Institute of Texas
11. British Columbia Electronic Library Network (link is external)
12. Broughton Archipelago Monitoring Plan (BAMP)
13. California Historical Society
14. California Institute of Technology (Caltech)
15. California Polytechnic State University (link is external)
16. Canadian Writing Research Collaboratory - University of Alberta (link is external)
17. Centre de recherches acadiennes de l'Île-du-Prince-Édouard (link is external)
18. Chinese University of Hong Kong (link is external)
19. City of Hope
20. CNR IPSP and CNR IRCrES - V2P2 Project (link is external)
21. Colorado College (link is external)
22. Colorado State Publications Library (link is external)
23. Commission for Environmental Cooperation - Green Building Library (link is external)
24. Commission for Environmental Cooperation - Virtual Library (link is external)
25. Danmarks Tekniske Informationscenter - Technical Information Center of Denmark
26. Davidson College (link is external)
27. Delft University of Technology (link is external)
28. Detroit Public Library (link is external)
29. discoverygarden, Inc. (link is external)
30. Drexel University Legacy Center College of Medicine (link is external)

[8] http://islandora.ca/islandora-installations

31. DuraSpace (link is external)
32. E-pistemetec Project
33. Editing Modernism in Canada Project (link is external)
34. Florida Gulf Coast University (link is external)
35. Florida State University (link is external)
36. Florida Virtual Campus (link is external)
37. Freshwater Biological Association (link is external)
38. Fundación Juan March (link is external)
39. Ghent University
40. Grinnell College (link is external)
41. Hagley College
42. Hamilton College (link is external)
43. Hampshire College
44. Hasat Kilise Kaynaklarına (link is external)
45. Innisfil Public Library (link is external)
46. Lafayette College (link is external)
47. Lawrence Berkeley National Laboratory (Berkeley Lab)
48. Library of the Oneida Community (link is external)
49. LYRASIS (link is external)
50. MacEwan University (Grant MacEwan)
51. Marmot Digital Repository
52. McMaster University (link is external)
53. Metropolitan New York Library Council (METRO) (link is external)
54. Minnesota State University - Mankato (link is external)
55. Moss Landing Marine Lab (link is external)
56. Mount Holyoke College (link is external)
57. National Agriculture Library, USDA
58. National Library of Medicine
59. National University of Ireland, Galway
60. New Orleans Jazz & Heritage Foundation
61. New York University Medical (link is external)
62. Northern Illinois University (link is external)
63. Oakridge National Laboratory
64. PALS (link is external)
65. Presbyterian Historical Society (link is external)
66. Prince Rupert Library (link is external)
67. Progressive Librarians Guild - Toronto (link is external)
68. Red Biodiversidad Chile (REUNA) (link is external)
69. Regis University (link is external)
70. Ryerson University (link is external)
71. Shang Shung Institute - Italy
72. Simon Fraser University
73. Smith College
74. Smithsonian
75. Southwest Minnesota State University (link is external)
76. St. Cloud Technical and Community College (link is external)

77. The Cherry Hill Company (link is external)
78. Tulane University Digital Library (link is external)
79. UCLA (link is external)
80. UNED, Universidad Nacional de Educación a Distancia
81. Universidad Catolica del Norte - Biodiversity Repository (link is external)
82. Universidad de la Frontera - Biodiversity Repository (link is external)
83. Universidad Metropolitana de Ciencias de la Educación - Biodiversity Repository (link is external)
84. University of Arkansas (link is external)
85. University of Connecticut (link is external)
86. University of Denver (link is external)
87. University of Hamburg - Hamburger Zentrum für Sprachkorpora (link is external)
88. University of Limerick (link is external)
89. University of Manitoba (link is external)
90. University of Missouri (link is external)
91. University of Missouri - Kansas City (link is external)
92. University of Missouri – Columbia (link is external)
93. University of Missouri – St. Louis (link is external)
94. University of New Brunswick (link is external)
95. University of New Brunswick - Saint John (link is external)
96. University of New Hampshire (link is external)
97. University of North Carolina at Charlotte (link is external)
98. University of North Texas - Comparative Assessment of Peer Review (CAPR) Project Repository (link is external)
99. University of Northern Colorado (link is external)
100. University of Otago Library (link is external)
101. University of Pittsburgh
102. University of Prince Edward Island (link is external)
103. University of Saskatchewan (link is external)
104. University of South Carolina (link is external)
105. University of St Andrews Library (link is external)
106. University of Tennessee (link is external)
107. University of Toronto (link is external)
108. University of Toronto - Scarborough
109. University of Vermont
110. University of Western Sydney
111. University of Wyoming (link is external)
112. US Geological Survey
113. Vassar College (link is external)
114. Vilnius University Library (link is external)
115. Washington Research Library Consortium (WRLC) (link is external)
116. Williams College (link is external)
117. Worthington Libraries (link is external)
118. York University (link is external)
119. Zuse Institute Berlin

References

About Fedora. (n.d.). Retrieved January 25, 2016, from http://fedorarepository.org/about

Bengtson, J., Bedi, S., Cooley, K., Goddard, L., McHenry, W., Nayyer, K., … Wilson, L. (2015). *UVic Libraries Operational Plan2015* (p. 11). Victoria, B.C.: The University of Victoria Libraries.

Clement, T., Hagenmaier, W., & Knies, J. L. (2013). Toward a Notion of the Archive of the Future: Impressions of Practice by Librarians, Archivists, and Digital Humanities Scholars. *The Library Quarterly: Information, Community, Policy*, *83*(2), 112–130. http://doi.org/10.1086/669550

Coyne, J. (2015, January 9). Try Sufia: A Fedora 4 plus Hydra Combination. Retrieved January 27, 2016, from http://duraspace.org/articles/2415

DuraSpace | Open technologies for durable digital content. (n.d.). Retrieved March 23, 2016, from http://duraspace.org/articles/2415

Features. (n.d.). Retrieved January 25, 2016, from http://fedorarepository.org/features

Fedora Commons. (2015, March 13). In *Wikipedia, the free encyclopedia*. Retrieved from https://en.wikipedia.org/w/index.php?title=Fedora_Commons&oldid=651163310

Fedora Digital Object Model - Fedora 3.4 Documentation - DuraSpace Wiki. (n.d.). Retrieved March 22, 2016, from https://wiki.duraspace.org/display/FEDORA34/Fedora+Digital+Object+Model#FedoraDigitalObjectModel-DigitalObjectModel-AccessPerspectiveaccess

Getting Started with Islandora - Islandora Documentation - DuraSpace Wiki. (n.d.). Retrieved March 24, 2016, from https://wiki.duraspace.org/display/ISLANDORA/Getting+Started+with+Islandora

Goddard, L. (2016). Developing the Read/Write Library. In *New Knowledge Models: Sustaining Partnerships to Transform Scholarly Production*. Whistler, B. C.

Juola, P. (2013, August 20). How a Computer Program Helped Reveal J. K. Rowling as Author of A Cuckoo's Calling. Retrieved January 27, 2016, from http://www.scientificamerican.com/article/how-a-computer-program-helped-show-jk-rowling-write-a-cuckoos-calling/

Pillay, R. (n.d.). IIPImage » Blog » IIIF – The International Image Interoperability Framework. Retrieved March 23, 2016, from http://iipimage.sourceforge.net/2014/12/iiif/

Technical Framework and its Parts - Hydra - DuraSpace Wiki. (n.d.). Retrieved March 23, 2016, from https://wiki.duraspace.org/display/hydra/Technical+Framework+and+its+Parts#TechnicalFrameworkanditsParts-ArchitectureandComponents

tesseract-ocr/tesseract. (n.d.). Retrieved January 28, 2016, from https://github.com/tesseract-ocr/tesseract

Yale Hydra Project. (n.d.). Retrieved February 25, 2016, from http://campuspress.yale.edu/yalehydranews/