discoverygarden

University of Connecticut
Connecticut Digital Archive

# Islandora Migration Assessment Report

*Transitioning CTDA to Islandora 8*

9/01/2021

V. 1.3

# Islandora Migration Assessment Report v. 1.2

*Transitioning CTDA to Islandora 8*

Overview

Islandora 8 Infrastructure
- Introduction
- Technology Stack Components
  - Drupal Modules

Technical Architecture
- File system Layout
- Configuration management

Architecture Diagram - Phase One
- Phase one recommendation
- Implementation Details
  - Firewall Details
  - Load Balancer Access Requirements
  - Drupal Nodes Access Requirements
  - Node Resource requirements
  - Handle server Access Requirements
  - Node Resource requirements
  - Database Cluster Access Requirements
  - Node Resource requirements
  - Crayfish Nodes Access Requirements
  - Node Resource requirements
  - ActiveMQ Access Requirements
  - Server Resource requirements
  - Solr Access Requirements
  - Server Resource requirements
  - Tomcat Access Requirements
  - Server Resource requirements
- Future Architecture Options
- Later Architecture Changes
  - Storage Solution
- Disaster Recovery and Failover Planning
- Meeting Core Trust Seal Requirements
  - 2. Licenses
  - 4. Confidentiality / Ethics
  - 7. Data Integrity and Authenticity
    - Data Integrity

Responses
Authenticity Management
Responses
8. Appraisal
Responses
9. Documented Storage Procedures
Responses
11. Data Quality
12. Workflows
13. Data Discovery and Identification
Guidance
Responses
14. Data Reuse
Guidance
Responses
15. Technology / Technical Infrastructure

Islandora 8 Configurations and Custom Development
Permissions and Access
Member Collection Management
Embargoes
Additional Controls
Simple Workflow
Field Permissions
Collections Search
Integrated Ingest Channels
Connecticut State Regulations (EREGS)
Connectic League of History Organizations (CLHO) Connector
Connecticut Historical Society (CTHS) Ingests
TMS
Koha
General Note
DPLA OAI Export
Manuscript Model
Multipage CSV Ingest Rows
Watermarking
Permanent Watermarks:
Watermarks on Request:
The above resolves the described case such:
Assumptions
Bulk Metadata Updates
Reporting

- User Content Lists
- Set Access Control via CSV Ingest
- Handle Service
- Additional Configuration and Customization
- Unsolicited Feature Guidance

Presentation Layer

Data Migration
- Migration of Collections and Repository Items
  - New Supporting Works from DGI
  - Migration Workflow
    - Creation of Taxonomies
- Remapping Persistent Identifiers
- Non-Repository Content
- Domain Name Management Assistance
- Cutover Strategy Options

Work Breakdown Structure (WBS), Estimated Costs, and Timelines

Assumptions, Risks, and Constraints
- Assumptions
- Risks
- Constraints

A1: Digital Lifecycle Curation Support in Islandora 8

A2: Drupal Modules Deployed with DGI Islandora 8

A3: Known Limitations of Fedora Implementation
- Additional Considerations
  - Multisites
  - Milliner JSON-LD representation of objects
  - TN are being stored in the public file system
  - Older version of Apache Karaf

A4: Technology Stack Components
- Connecticut Digital Archive

# Overview

This report provides a general implementation plan for the Connecticut Digital Archive (CTDA) migration from its current Islandora 7 (I7) platform to Islandora 8 (I8). It outlines the proposed solutions for:

- Islandora 8 Infrastructure
- Data Migration and Cutover Strategy Options
- Islandora 8 Configurations and Custom Development
- The Presentation Layer
- Estimated Costs and Timelines

Supplemental documentation to support each segment of the work will be provided in Appendices as appropriate. Also included is a list of Assumptions, Risks, and Constraints.

The report assumes familiarity on behalf of the reader with the workings of the existing CTDA Islandora 7 repository at https://ctdigitalarchive.org/.

# Islandora 8 Infrastructure

## Introduction

This section explains technical details and illustrates potential solutions for a scalable Islandora 8 infrastructure meeting CTDA's short and long term goals. Possible risks, issues and constraints are discussed.

We recommend review of high level Islandora community documentation links to start:

- https://islandora.github.io/documentation/
- Architectural diagram: https://islandora.github.io/documentation/technical-documentation/diagram/.
- Component overview: https://islandora.github.io/documentation/installation/component_overview/

DGI's current recommended approach for production environments is not to use Fedora, and it sees no issues for this approach with respect to compliance with the Core Trust Seal Requirements that CTDA needs to apply for and maintain. Responses to these requirements are given in full under **Meeting Core Trust Seal Requirements**. Details of limitations of Fedora are discussed in **Appendix A3: Known Limitations of Fedora Implementation**.

Given the preceding guidance, some components from the community stack may or may not be used. In particular Gemini (being retired in upcoming releases of Islandora), Recast, Milliner, Blazegraph, Matomo and FITS may not be included.

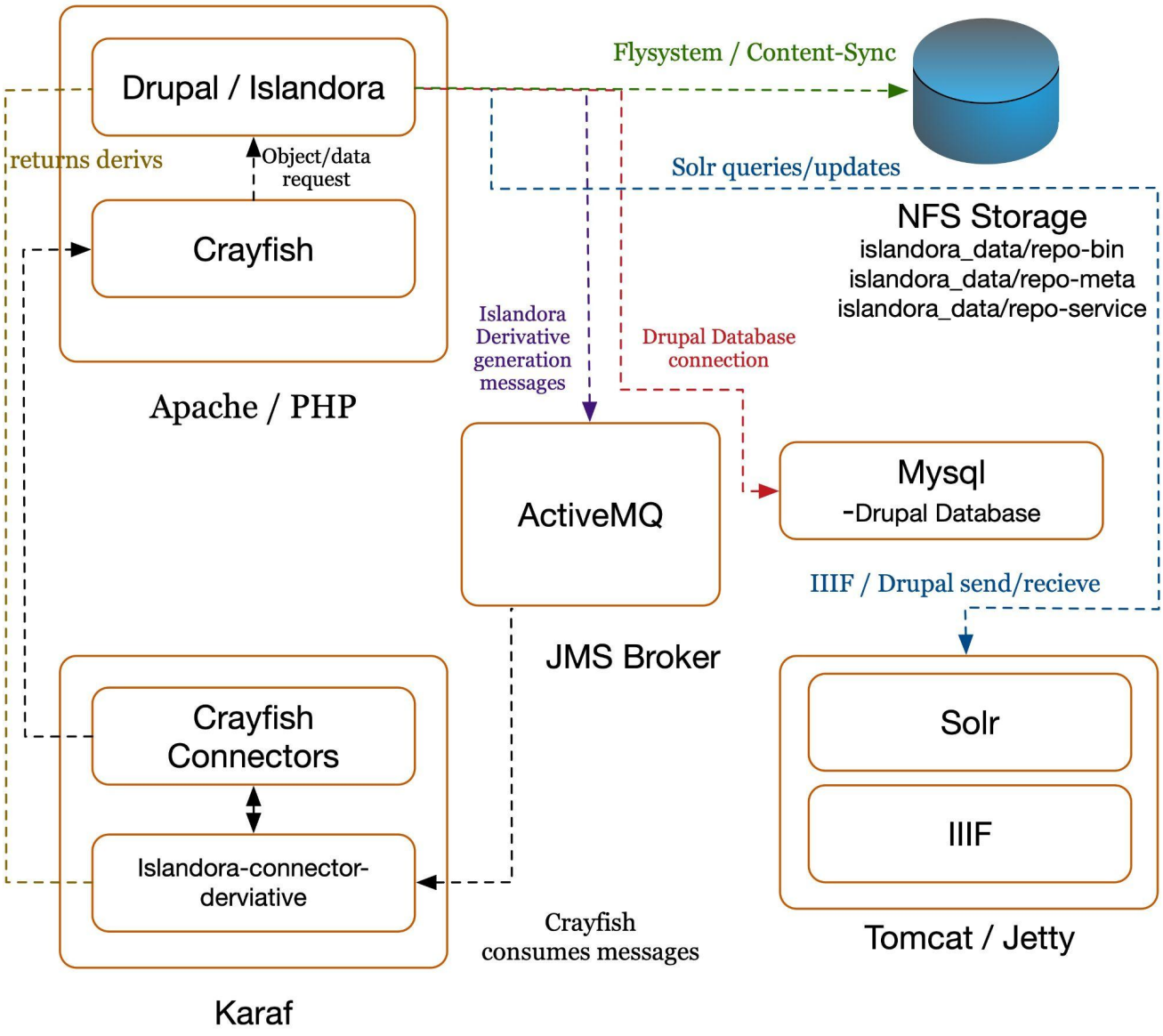
## Technology Stack Components

See **Appendix A4: Technology Stack Components**.

### Drupal Modules

A full listing of the modules DGI includes with its Islandora 8 solutions is given in **Appendix A2: Drupal Modules**. The assessment lists others that will be used for meeting CTDA requirements.

# Technical Architecture




## File system Layout

DGI will utilize the following file system structure in local/NFS storage (defaults to /opt/islandora_data/):

- **repo-bin** - Stores the preservation master binaries attached to an object. Our default configuration broadly organizes files into folders named after the year and month they were first uploaded.

- **repo-service** - Stores files derived directly and indirectly from those in repo-bin. These are the frequent-use files that are actually disseminated to Drupal site users hence the name “service”.
- **repo-meta** - Stores all the Drupal-related metadata for repository objects that could be used to rebuild it from scratch in a recovery scenario.

## Configuration management

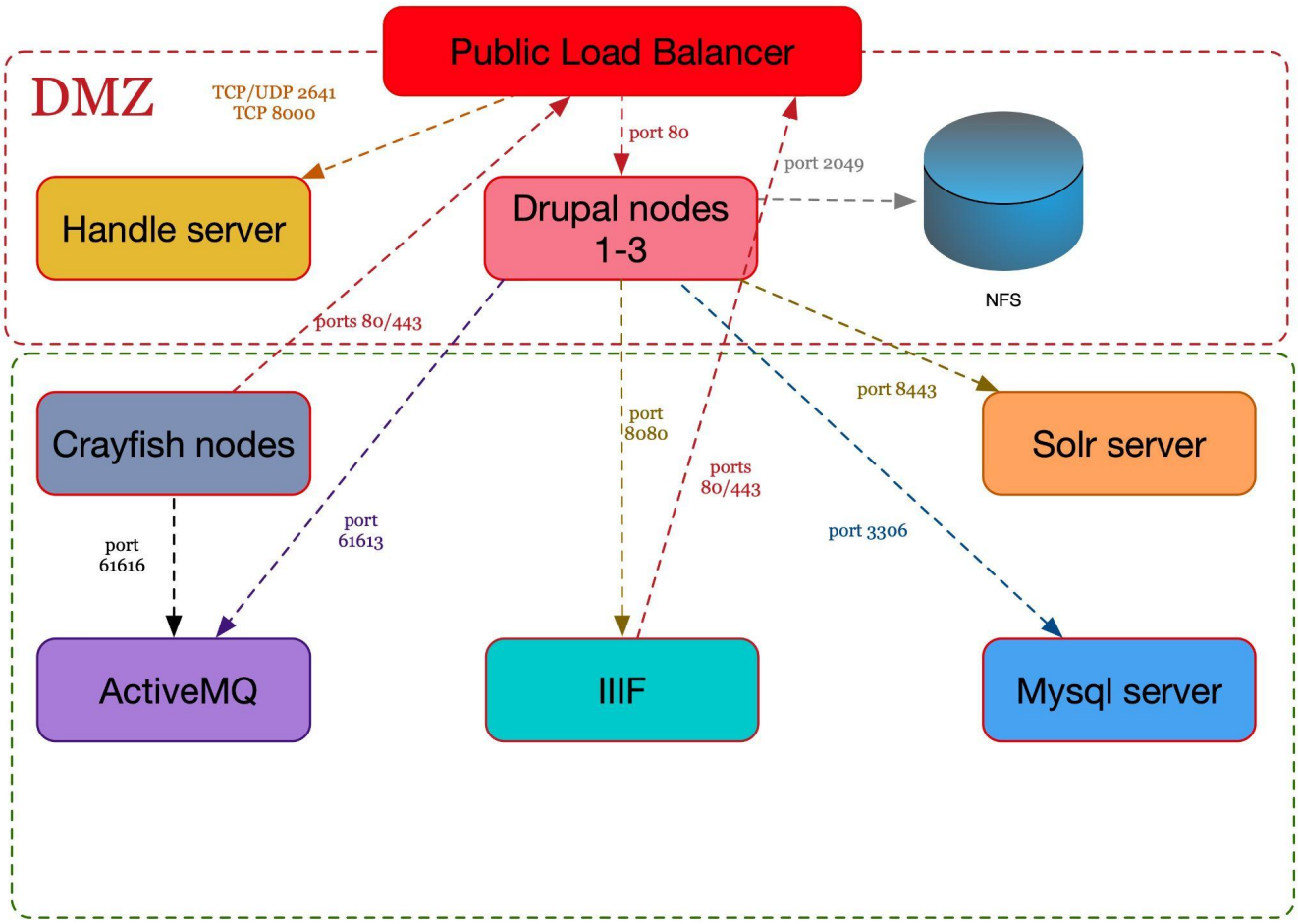
**Puppet** - is a system management tool for centralizing and automating the configuration management process. Discoverygarden makes use of the Puppet Enterprise 2018.1.9+ LTS offering to manage installations and deployments. Depending on your local infrastructure and maintenance contract this may or may not remain present on your system.

**Composer** - A Dependency Manager for PHP. We utilize composer to install/update modules within the Drupal ecosystem.

**Github** - hosting for software development and version control using Git. Github is utilized as our central code repository. We make use of both private and public repositories in an Islandora8 installation.

**Packagist** - Packagist is the main Composer repository. We make use of our own private packages as well as public ones.

## Architecture Diagram - Phase One




## Phase one recommendation

This is a basic starting architecture which will allow for quicker implementation times while still allowing DGI to scale the stack as it learns more about the performance requirements and possible bottlenecks of varying components. The migration of all of CTDA's data will take time.

Islandora 8 is drastically different from Islandora 7. Many components will use different levels of resources. For example Blazegraph isn't actually required by any component it just provides a Triplestore endpoint for specialized users to carry out SPARQL queries as required so it shouldn't need the same level of resources as CTDA's existing Blazegraph which is constantly answering queries. It still however will end up needing the same amount of disk space. The vast majority of traffic and requests will be against Drupal and its corresponding database/file system.

Since Drupal itself is the main point of truth for the repository there's not really the same concept of a Public and Private Drupal frontend as in Islandora 7, which used a backend Fedora as the repository serving up

content. It is likely possible to have two separate sets of Drupal servers for “Public” and “Private” for security isolation purposes, but this needs to be examined further since they would be sharing the same Drupal database. If a “Public” server is unable to write to the database for caching and other purposes it likely won’t work properly without additional work.

Since this is still under active development the CPU / memory recommendations are subject to change.

## Implementation Details

### Firewall Details

Firewall rules will be handled by UITS.

**Public Load Balancer** - It is assumed that this is an Application Load Balancer able to intelligently direct HTTP/HTTPS traffic on various ports to healthy nodes. Functionality should be similar to an AWS ELB Application Load Balancer. Different ports should be restricted to intended services/users.

### Load Balancer Access Requirements

- ports 80/443 opened to intended audience
- ports 80/443 forwarded/balanced between healthy Drupal nodes.
- ports 80/443 opened to IIIF server as well as Crayfish nodes.

**Drupal nodes / Islandora (fronted by Load Balancer)-** Generally there should be a minimal number of servers across several hypervisors for fault tolerance. DGI recommends starting with 2-3 nodes. Drupal and Islandora are installed on each node’s block storage. The Drupal database is a database server/cluster. Data such as sites directories, files, private files are stored within NFS which is attached to each instance. The Drupal file system on the NFS mount will make up the bulk of your repository size. The sizing of this will likely be similar to what your current NFS mount has (which is currently using 52TB). When traffic comes into the Load balancer it will route traffic to the appropriate healthy instance. Messages generated by Islandora are sent to ActiveMQ which in turn get picked up by the Crayfish stack. The Load balancer ports for Drupal will need to be accessible by the intended audience along with Crayfish and the IIIF server.

These nodes will be serving up web traffic and kicking off ingest jobs. Actual derivative generation will be carried out by Crayfish on separate nodes.

Starting spec: 4CPU and 8GB of RAM.

### Drupal Nodes Access Requirements

- ports 80/443 opened to Load Balancer
- Port 3306 -> Database cluster
- Port 61613 -> ActiveMQ server
- Port 8983 -> Solr server
- Port 8080 -> Tomcat server
- Port 2049 -> NFS
- SSH access for DGI

### Node Resource requirements

- 4 CPU
- 8GB Memory
- 200GB OS System Disk
- mounted filestore which contains the staged data to ingest
- mounted filestore which will contain Drupal file system and bulk of repository data. It will likely need to be similar in size to the existing NFS mount being used for the Islandora7 Fedora repository. Which is using 52TB currently.
- RHEL 8.x

**Handle Server** - Handle.net server. It does seem that the Handle.net server has mirror/replication abilities http://www.handle.net/tech_manual/HN_Tech_Manual_9.pdf. We might consider having two Handle servers for fault tolerance. Additional testing would need to take place to ensure that it is operating as we expect. UITS Load balancer is likely an application load balancer which might not be able to deal with UDP. The Handle server can use just TCP but could suffer a performance hit. We might have to have it come in from a separate entry point like a firewall if it does not work well through the Load balancer.

### Handle server Access Requirements

- Port TCP 8000 opened to load balancer (or Firewall)
- Port TCP/UDP 2641 opened to load balancer (or Firewall)

### Node Resource requirements

- 1 CPU
- 4GB Memory
- 200GB OS System Disk
- RHEL 8.x

**Database Cluster / Server** - This can be a stand alone server/cluster or it could utilize a managed database service if the University offers one compatible with MySQL or PostgreSQL. Memory allocations are difficult to tell at this point since its usage will be different from Islandora 7. Additional tuning / memory changes are likely.

### Database Cluster Access Requirements

- Port 3306 opened for Drupal Nodes.

### Node Resource requirements

- 4 CPU
- 25GB Memory
- 200GB OS System Disk
- RHEL 8.x

**Crayfish Nodes -** DGI recommends starting with 2-3 nodes, a server group containing nodes with Karaf, Apache, PHP and Crayfish installed. These nodes consume Islandora messages from ActiveMQ which in turn generate derivatives and send the data to Islandora. In this case "Islandora" would actually be the Load Balancer which would send the traffic to nodes in the Drupal stack. The Crayfish stack needs to connect to Islandora in order to retrieve data to generate derivatives. After it generates the derivatives, it sends them to Islandora. This group can be any number of nodes depending on how quickly you want your queue worked.

These nodes will be feeding off ActiveMQ and generating derivatives. They will be sending information to the Drupal Load Balancer to update Drupal..

### Crayfish Nodes Access Requirements

- ports 80/443 -> Load Balancer
- Port 61616 -> ActiveMQ server
- SSH access for DGI

### Node Resource requirements

- 4 CPU
- 8GB Memory
- 200GB OS System Disk
- RHEL 8.x

**ActiveMQ** - Initially this will be a single server but can be made into a cluster. See (https://activemq.apache.org/clustering). ActiveMQ will need to be accessible from Islandora/Drupal and the Crayfish stack. Islandora communicates with ActiveMQ on port 61613 (61614 with SSL), Karaf communicates with ActiveMQ on port 61616 (61617 with SSL).

The amount of memory required will depend on the number of Drupal nodes, derivative generation traffic, and the number of Crayfish nodes.

Starting spec: 2CPUs and 4GB of RAM. Reference for specs from AmazonMQ https://docs.aws.amazon.com/amazon-mq/latest/developer-guide/broker-instance-types.html#activemq-broker-instance-types.

### ActiveMQ Access Requirements

- Port 61616 opened for Crayfish Nodes
- Port 61613 opened for Drupal Nodes
- SSH access for DGI

### Server Resource requirements

- 2 CPU
- 4GB Memory
- 200GB OS System Disk
- RHEL 8.x

**Solr Server** - This will be running the latest Solr 8.x which runs inside of Jetty. The requirements of this server will likely be similar to that of your existing solr server which runs 4CPU and 50GB of memory however given that the migration will take some time to complete we might want to start off smaller provided we can easily increase the resource allocations later.

### Solr Access Requirements

- Port 8983 opened for Drupal Nodes
- SSH access for DGI

### Server Resource requirements

- 4 CPU
- 24GB Memory (we likely can start off with half as much initially provided we can increase easily as we scale)
- 200GB OS System Disk
- 200GB Index disk (should be SSD ideally)
- RHEL 8.x

**IIIF/ Tomcat Server** - This server will run Cantaloupe 4.1.9+ which allows for IIIF viewing.

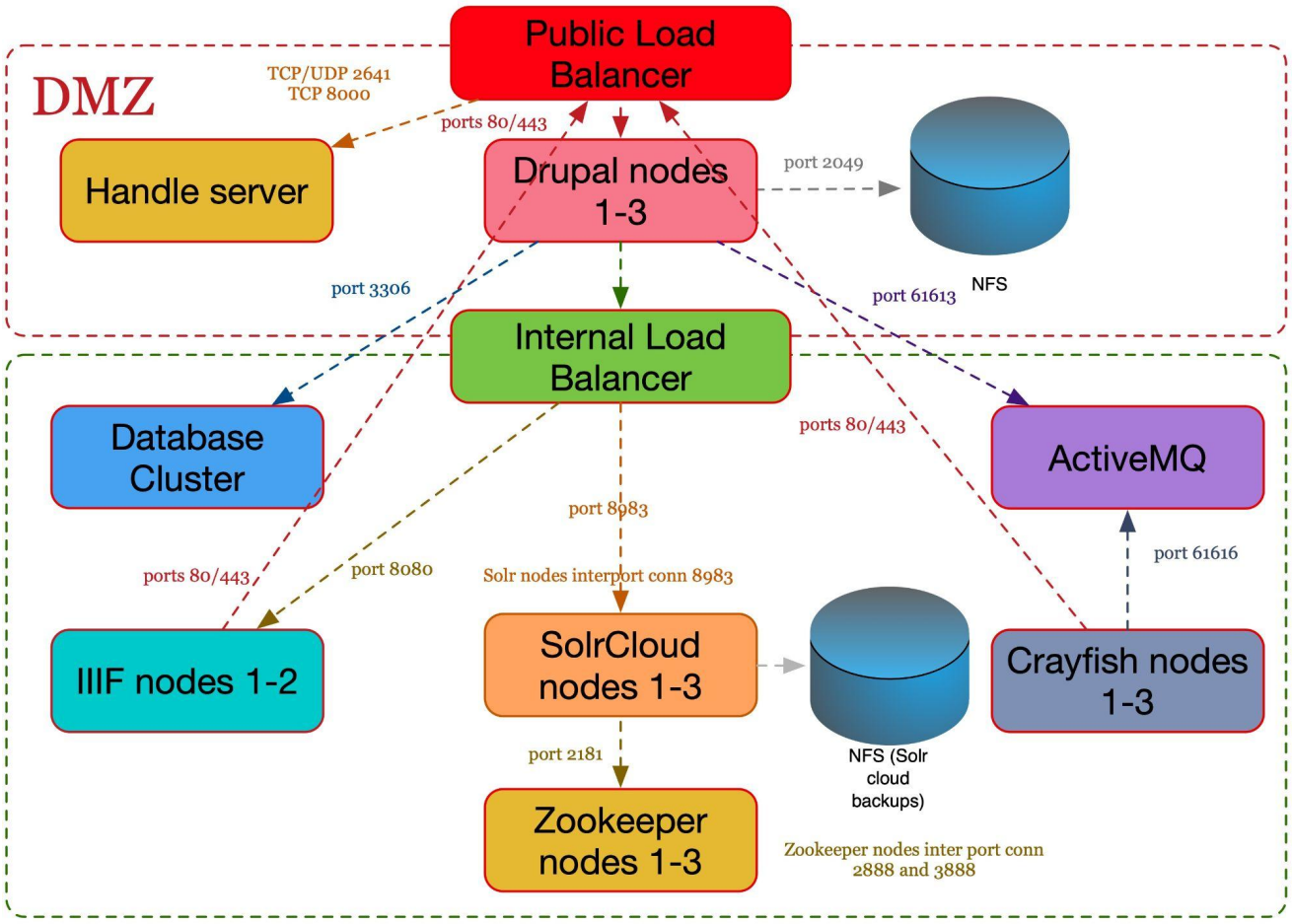
### Tomcat Access Requirements

- Port 8080 opened for Drupal Nodes
- ports 80/443 -> Load Balancer (For IIIF)
- SSH access for DGI

### Server Resource requirements

- 4 CPU
- 10GB Memory (Could drop this to )
- 200GB OS System Disk
- RHEL 8.x

## Future Architecture Options




## Later Architecture Changes

This section provides a high level explanation how the stack can be scaled as needed.

The Islandora 8 stack is designed so that parts of it can be broken out and managed separately, allowing scaling of individual stacks. Infrastructures can vary but in AWS a concept of “Autoscaling” is used that can add/remove servers depending on load. Depending on UITS infrastructure options this might not be a possibility. If autoscaling is not possible, servers will need to be added or removed manually. The ways that the servers can be broken out can vary. In some cases, more than one service can be kept running in a particular grouping. For example, keeping Karaf and Crayfish running on the same node set with an external Activemq.

This later phase illustration has the following additional concepts that weren't discussed in the Phase One diagram:

**Solr Cloud (fronted by Load balancer)** - SolrCloud is flexible distributed search and indexing, without a master node to allocate nodes, shards and replicas. Instead, Solr uses ZooKeeper to manage these locations, depending on configuration files and schemas. Queries and updates can be sent to any server. DGI usually uses 3 solr cloud servers and 3 ZooKeeper servers. It recommends using one shard for your collection then a replica for each node that you have available. In an attempt to improve performance shards can be broken out per collection, however this is a much more involved configuration that will take additional time and testing and can further complicate backup/recovery efforts.

To achieve fault tolerance, every node must have a replica of every collection. ZooKeeper will automatically distribute the replicas to all nodes. If using autoscaling, commands can be passed via API to adjust the number of replicas to match running nodes.

All Solr nodes will need a shared NFS mount with a folder that has the same UID for the "solr" user that is running the solr process. This will be used to store backups of the Solr index.

Any Solr node can work query/update messages coming from the Application Load balancer on port 8983. This should only be accessible to the Islandora/Drupal servers. Solr nodes need to be able to talk to each other on port 8983

The amount of memory each Solr node requires can vary greatly depending on the size of the index, number of indexes, shards and replicas. In the case of CTDA each Solr node would likely need to be similar in size to your existing Solr server. To quote Apache's documentation:

> *For index updates, Solr relies on fast bulk reads and writes. For search, fast random reads are essential. The best way to satisfy these requirements is to ensure that a large disk cache is available. Visit Uwe's blog entry for some good Lucene/Solr specific information. You can also utilize Solid State Drives to speed up Solr, but be aware that this is not a complete replacement for OS disk cache. See the SSD section later in this document for more details.*
>
> *In a nutshell, you want to have enough memory available in the OS disk cache so that the important (frequently accessed) parts of your index will fit into the cache. Let's say that you have a Solr index size of 8GB. If your OS, Solr's Java heap, and all other running programs require 4GB of memory, then an **ideal** memory size for that server is at least 12GB. You might be able to make it work with 8GB total memory (leaving 4GB for disk cache), but that also might NOT be enough. The really important thing is to ensure that there is a high cache hit ratio on the OS disk cache … not to achieve perfection.*

(See https://cwiki.apache.org/confluence/display/SOLR/SolrPerformanceProblems.)

**ZooKeeper** - ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. At least 3 servers are required to form a ZooKeeper ensemble (term for "cluster"). DGI uses ZooKeeper to manage its Solr Cloud installation. The ZooKeeper nodes should ideally be in the same VLAN as the Solr nodes. ZooKeeper nodes need to be able

to communicate with each other on ports 2888 and 3888. Solr nodes need to be able to talk to ZooKeeper Ensemble on ports 2181.

A general rule of thumb is that ZooKeeper nodes should have a minimum of 1 CPU and 4GB of RAM, however given the size of CTDA's current Solr index we might want to add quite a bit more resources to this. Perhaps double the minimum recommended to start and watch performance metrics as we make use of it.

**IIIF -** IIIF Cantaloupe servers that can sit behind a load balancer. This would need to be accessible from Islandora/Drupal. IIIF is run from a Tomcat install on port 8080.

Starting spec: 2 CPU and 10GB of RAM per node.

Note: This could vary if you have extremely large JP2 images.

**Load Balancer -** An internal load balancer is labelled for its intent but this could be the same device as the "Public Load balancer" depending on device configuration and network setup.

### Storage Solution

Islandora 8 can work with s3, block storage or network based storage (e.g. such as NFS). Other storage options can be considered but would need to be evaluated. Generally speaking you should expect to need at least 52TB of network based storage with room to grow (given current repository size).

The implementation plan will include steps for more accurately assessing space needs. Test collections will be migrated in a development environment and benchmarks collected from those efforts.

During the migration phase space requirements will be doubled for the period of time it takes to transit all of the CTDA collections to the I8 infrastructure.

**Note:** While DGI is aware there may be new offerings from UITS for storage of archival versions of assets, feasibility, risk assessment, testing, and estimating of any development required to implement them would need to be performed before recommending their use.

## Disaster Recovery and Failover Planning

DGI will provide a Disaster Recovery and Failover plan for CTDA's I8 implementation, working with resources available at UITS. Included will be recommendations for:

- Repository data.
- Minimum recommended retention schedules.
- Instance snapshots.
- Database restoration.
- Steps to execute.

# Meeting Core Trust Seal Requirements

As a part of the assessment the sections of the Core Trustworthy Data Repositories Requirements relevant to services DGI will provide have been reviewed, i.e. sections and/or line items of the Requirements relating to governance, project practices, or otherwise not related to the technical solutions DGI is providing are omitted. Responses to the relevant requirements are provided here by section (and point in some cases).

## 2. Licenses

**R2. The repository maintains all applicable licenses covering data access and use and monitors compliance.**

CTDA uses the rightsstatement.org vocabulary. The I8 repository provides this as a taxonomy that is targeted by an entity reference field on every repository item.

See also (if relevant) **Permissions and Access**.

## 4. Confidentiality / Ethics

**R4. The repository ensures, to the extent possible, that data are created, curated, accessed, and used in compliance with disciplinary and ethical norms.**

- The CTDA repository is not used to store any Personally Identifiable Information (PII).
- Metadata access can be permissioned by role as needed.
- Access to archival materials may be controlled by roles and group membership. See **Permissions and Access**.

## 7. Data Integrity and Authenticity

**R7. The repository guarantees the integrity and authenticity of the data.**

### Data Integrity

For this Requirement, responses on data integrity should include evidence related to the following:

- Description of checks to verify that a digital object has not been altered or corrupted (i.e., fixity checks) from deposit to use.
- Documentation of the completeness of the data and metadata.
- Details of how all changes to the data and metadata are logged.
- Description of version control strategy.
- Usage of appropriate international standards and conventions (which should be specified).

#### Responses

- The repository stores checksums for ingested files which are periodically monitored to ensure integrity. All files are virus scanned on upload.
- CTDA has developed a documented metadata application profile which all depositors must follow. This profile is enforced by all ingest channels whether through the user interface or other channels for bulk ingest. As needed the repository supports creation of views for reporting on compliance.
- From initial ingest, every change to the metadata or original file is stored as a version. Each change to the metadata logs: who made the change, when it was made, what was changed. The user interface provides tools for repository administrators to view, manage, and restore prior versions if needed.

- The CTDA metadata profile utilizes the Library of Congress Metadata Object Description Schema (MODS). The repository has the capability to share image resources via the International Image Interoperability Framework (IIIF) and provides an Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) endpoint for harvesting, which in standard form serves metadata in Dublin Core Metadata Initiative (DCMI) Metadata Terms, but is also capable to serve other standardized metadata formats if required in the future.

### Authenticity Management

Evidence of authenticity management should relate to the following questions:

- Does the repository have a strategy for data changes? Are data producers made aware of this strategy?
- Does the repository maintain provenance data and related audit trails?
- Does the repository maintain links to metadata and to other datasets? If so, how?
- Does the repository compare the essential properties of different versions of the same file? How?
- Does the repository check the identities of depositors?

### Responses

- See prior response describing versioning. Supplementing the versioning and audit trail, the repository provides fields for provenance, change, and other types of custodial notes.
- Digital files are linked to their metadata records, and the metadata records link to internal and external entities as follows:
    - References to terms contained in taxonomies (controlled vocabularies), which in turn link to their Authority Sources.
    - Membership in multiple collections is supported by a type of link (isMemberOf).
    - Links can be made to related items existing in the repository and or at external resources. These links are typed using DCMI Relator Terms.
- CTDA provides repository accounts only to trusted Member organization staff. All users must authenticate to have any interaction with repository content other than to search and view publicly displayable information and content.

## 8. Appraisal

**R8. The repository accepts data and metadata based on defined criteria to ensure relevance and understandability for data users.**

For this Requirement, responses should include evidence related to the following questions [*excerpted here to those relevant to DGI and/or the repository itself*].

- Does the repository have procedures in place to determine that the metadata required to interpret and use the data are provided?
- Is there any automated assessment of metadata adherence to relevant schema?
- Are checks in place to ensure that data producers adhere to the preferred formats?
- What is the process for removing items from your collection, also keeping in mind impact on existing persistent identifiers?

Responses

- CTDA has developed a documented metadata application profile which all depositors must follow. This profile is enforced by all ingest channels whether through the user interface or other channels for bulk ingest. As needed the repository supports creation of views for reporting on compliance.
- Each allowed type of repository item, e.g. Image, Audio, etc., controls the file formats (mime types) that are allowed, disallowing ingest of unapproved formats.
- Recording deletions isn't something currently done in CTDA. The option to control what user roles have permissions to delete items is present, but no Tombstoning functionality is in place. This option - which could be pursued by auditing and subsequently implementing the Drupal Tombstone module (https://www.drupal.org/docs/contributed-modules/tombstones) - might be worthy of discussion.

## 9. Documented Storage Procedures

**R9. The repository applies documented processes and procedures in managing archival storage of the data.**

For this Requirement, responses should include evidence related to the following questions:

- How are relevant processes and procedures documented and managed?
- Does the repository have a clear understanding of all storage locations and how they are managed?
- Does the repository have a strategy for multiple copies? If so, what is it?
- Are risk management techniques used to inform the strategy?
- What checks are in place to ensure consistency across archival copies?
- How is deterioration of storage media handled and monitored?

Responses

- CTDA has copies of all infrastructure, backup, and failover planning documentation.
- Yes. (See **Technical Architecture** in this report for content to draw on.)
- “
- ”
- See **Islandora 8 Infrastructure** in this report for content to draw on.
- “

## 11. Data Quality

**R11. The repository has appropriate expertise to address technical data and metadata quality and ensures that sufficient information is available for end users to make quality-related evaluations.**

Most of this would appear to be questions CTDA would respond to but a couple of points here are relevant:

- CTDA has developed a documented metadata application profile which all depositors must follow. This profile is enforced by all ingest channels whether through the user interface or other channels for bulk ingest. As needed the repository supports creation of views for reporting on compliance.
- If CTDA desired to have community comments it could be permissioned to specific roles using the Drupal (core) Comments model (https://www.drupal.org/docs/8/core/modules/comment).

## 12. Workflows

**R12. Archiving takes place according to defined workflows from ingest to dissemination.**

This would appear to be questions CTDA would respond to but the following Appendix is provided which explains I8's alignment vv. OAIS: **Appendix A1: Digital Lifecycle Curation Support in Islandora 8.**

## 13. Data Discovery and Identification

**R13. The repository enables users to discover the data and refer to them in a persistent way through proper citation.**

### Guidance

Effective data discovery is key to data sharing. Once discovered, datasets should be referenceable through full citations, including persistent identifiers to help ensure that data can be accessed into the future.
For this Requirement, responses should include evidence related to the following questions:

- Does the repository offer search facilities?
- Does the repository maintain a searchable metadata catalogue to appropriate (internationally agreed) standards?
- What persistent identifier systems does the repository use?
- Does the repository facilitate machine harvesting of the metadata?
- Is the repository included in one or more disciplinary or generic registries of resources?
- Does the repository offer recommended data citations?

### Responses

- Yes. The repository may be searched via the UI or queried via API.
- Yes. All content is indexed by Solr.
- Handle.
- Yes. An OAI-PMH service is provided. Data may also be harvested via API.
- (CTDA to respond.)
- Repository Item landing pages provide sharing tools that allow copying of the Hande for reference.

## 14. Data Reuse

**R14. The repository enables reuse of the data over time, ensuring that appropriate metadata are available to support the understanding and use of the data.**

### Guidance

Repositories must ensure that data continues to be understood and used effectively into the future despite changes in technology and the Designated Community's knowledge base. This Requirement evaluates the measures taken to ensure that data are reusable.

For this Requirement, responses should include evidence related to the following questions:

- Which metadata are provided by the repository when the data are accessed?
- How does the repository ensure continued understandability of the data?
- Are data provided in formats used by the Designated Community? Which formats?
- Are measures taken to account for the possible evolution of formats?

Responses

- All descriptive metadata fields and select technical metadata fields are provided via the repository UI. (CTDA may want to include in its response a reference to its metadata profile documentation.) OAI-PMH requests harvest DCMI Terms. API requests can serve all metadata fields for repository items and their media.
- The repository provides facilities that support monitoring for file formats which may be obsoleted. As new media formats are adopted, new digital versions of an asset may be attached to a repository item.

## 15. Technology / Technical Infrastructure

**R15. The repository functions on well-supported operating systems and other core infrastructural software and is using hardware and software technologies appropriate to the services it provides to its Designated Community.**

See all responses and references to relevant guidance from the **Islandora 8 Infrastructure** section. See also **Disaster Recovery and Failover Planning**. (It could be noted that the process being planned here to transition DAMs frameworks is evidence of CTDA's commitment to and process for continuity.)

# Islandora 8 Configurations and Custom Development

This assessment evaluated feature parity between CTDA's I7 instance and I8, both for core and custom modules and features. This section lists items that will require custom configuration and/or development to reproduce in I8. The I7 and I8 frameworks are not analogous in many respects, so here "reproduce" doesn't imply "replicate exactly", but instead means to offer a very similar feature in a useful fashion. Inclusion here doesn't necessarily mean the work will carry forward. Some features are required, others subject to CTDA's evaluation of their value. Also included are some new features CTDA has inquired about during the assessment process.

## Permissions and Access

### Member Collection Management

In the course of the migration CTDA has sought to reduce overhead and simplify the components and methodology of the implementation that allow for its Members to manage their own collections. The core requirements for member content management are that each member:

- can manage their own collections, creating, updating, and deleting repository items as they need;
- can perform bulk changes to their content;
- can perform bulk ingests to their collections;
- can control which CTDA users have anything beyond read permissions to their collections;
- can in a simple fashion brand their collections, in particular the top level;
- can offer users the ability to search only within a given collection, e.g. the Member organization's collection(s).

This assessment process has arrived at a solution supporting the first four points (access control) using the Drupal Group module (https://www.drupal.org/project/group) with some custom configurations. The branding and search requirements are addressed in the **Presentation Layer** section.

Groups supports the creation of communities within a Drupal instance. The proposed implementation will be along these lines:

- Each CTDA Member has a top level collection. CTDA administrators will be responsible for creating a Group for each Member organization and assigning this top level collection to the Member's Group.
- In the CTDA Drupal instance, a Member Curator role will be created and assigned to the Member organization user who will be responsible for administering their CTDA collections.
- CTDA administrators will grant a Member Curator the role of Group Administrator for that Member's Group.
- An Organization's Group administrator will be able to add CTDA users from their organization to their Group, and grant them access to manage their content.
- DGI will create customizations to ensure that content created is assigned membership in the correct Group, regardless of the method of creation: UI ingest, CSV ingest, or other batch process.
- The Views Bulk Edit module employed for batch changes to repository items respects Group permissions.

This implementation plan ensures that while Member organizations can manage their own content they are restricted from making alterations to any other Member's. That said, there is a bonus to this plan: Any CTDA

user can be added to a Group by the Group Administrator, so collaboration across Member groups is supported. Further, repository items can not only be a member of more than one collection, they may also be assigned to more than one Group.

### Embargoes

While there is an Embargo module for I8 it currently functions on a per repository item basis. CTDA has inquired about a feature to apply embargoes to a group of items with a single request. DGI can provide this by configuring a Bulk Views Edit Action to apply them to items that are a member of a given collection.

### Additional Controls

#### Simple Workflow

CTDA Member Curators and also control the publication status of their content. Setting the Published flag to 'N' will prevent user roles that do not have view access to unpublished content, e.g. unauthenticated (public) users.

#### Field Permissions

If for whatever reason it is desired to set permissions on certain metadata fields, e.g. as to hide them from unauthenticated users, Field Permissions can be used to do so.

## Collections Search

See **Presentation Layer**.

## Integrated Ingest Channels

There are several channels through which content is bulk ingested into CTDA's I7 instance that will need to be recreated in the I8 environment. Details of each follow.

### Connecticut State Regulations (EREGS)

The CTDA repository serves the State of CT as a partner by providing archival storage for CT State Eregulations (EREGS). A bespoke application serves to ingest submissions packages created and delivered by EREGS FileNet document management system into the CTDA Islandora repository on a scheduled basis.

DGI's proposed approach to replacing this in the new I8 environment is:

- Utilize Drupal Migrate to avoid management of CRUD operations on Drupal entities, and utilize things like entity looks up for a given data model. It's beneficial to derivative migrations such that each can live independently of the one before it.
- Use the Drupal Migrate Directory module or an extension of it as the "source plugin" to handle processing the "watched zips" directory via cron.
- Write custom source plugins that can replicate the behavior defined above in the current EREGS processing flow (ex: file skipping logic, collection identification from MODS and so forth).

### Connectic League of History Organizations (CLHO) Connector

The CLHO connector will utilize the same approach (framework) used for EREGS, adding custom source plugins.

### Connecticut Historical Society (CTHS) Ingests

#### TMS

The TMS import workflow relies on the use of CSV ingest, so from the CTDA side no work beyond that of configuring CSV for its own purposes is required; however, on the CTHS side the TMS report that creates the CSV content will need updates. This will involve consulting and testing time to communicate the new report output template to them, review the outputs, and iteratively test and refine the work. CTHS should plan on 2 Project Days for this work.

#### Koha

The Koha ingest, which imports MARC xml exported from Koha, will utilize the same approach (framework) used for EREGS, adding custom source plugins.

### General Note

All of the Migrate workflows will assign content to the correct Groups.

## DPLA OAI Export

**Removed from Scope.** Possible solutions are fully described in v. 1.0 of this report.

## Manuscript Model

There are at this time no features equivalent to those provided by the I7 Manuscript Solution Pack CTDA employs. The following features will need development in I8:

- Ability to store TEI transcripts as media of a Page model repository item.
- Display of TEI transcripts with page images (when present).
- (opt.) Have a method for batch ingest and association of the TEI transcripts.

The solution will use Islandora 8's Paged Content model, adding a TEI Transcript Media type to it. An XSL Formatter will be implemented to transform the transcript into HTML and display it in a block when the transcript is present. See **Assumptions, Risk, Constraints**.

## Multipage CSV Ingest Rows

Courtesy of some work for the Connecticut HIstorical Society, CDTA's I7 spreadsheet ingest supports ingestion of multiple pages or compound children in a single row of source data (= "single row aggregates") rather than having to prepare a row for each item. To retain this feature in I8 requires custom development that is included in the **WBS, Estimated Costs, and Timelines**.

## Watermarking

CTDA's I7 instance provides watermarking features via Islandora Watermark. It offers, among other options, the ability to select a collection whose new members should have watermarks applied on ingest. The main use cases are driven by one stakeholder:

- An image ingested by CSV has a specific watermark applied if a flag is set.
- An image ingested via the UI has a specific watermark applied if a flag is set.
- A CTDA admin can manage configurations used for watermarks.

Islandora 8 options for an equivalent were scoped as a part of this assessment. There are a couple of ways of doing this, depending whether the images should have permanent watermarks or not.

### Permanent Watermarks:

This would have to be done at time of derivative creation, which would mean either extending Houdini (I8 microservice) or creating a secondary derivative creator that functions after Houdini is done. It doesn't appear watermarking can be accomplished using the basic controls on the derivative actions. This does have the drawback of being more configuration to set up an individual collection, but it also acts as kind of a set-it-and-forget-it to watermarking, and functions closely to I7's Islandora Watermark.

### Watermarks on Request:

We can provide watermarks as an image style using Drupal Basic Watermark https://www.drupal.org/project/basic_watermark. This has the drawback of being, again, quite a bit of configuration to set up for any new collections that need to be added.

### The above resolves the described case such:

- A user filling out the form in the UI or in CSV sets the watermark flag.
- They also set the collection.
- In the background, by virtue of the collection assignment and the flag being set, actions and context are able to assign the correct watermark using one of the two above methods

### Assumptions

This is running on the assumption that, like in Drupal 7, these watermarks are intended to be applied per-collection. A further option: Something much like the 'representative image' field for collections where an existing media or file is selected as the watermark at the time of ingest, which any of these pieces could potentially use down the line. This would require a little more development than the preceding options, but may deliver something that's more usable long-term and makes more sense from a UI perspective.

## Bulk Metadata Updates

Islandora 8 now includes Views Bulk Operations and View Bulk Edit modules, which will be utilized for this. CTDA admins and Member admins will be trained in their use. At this time it is assumed after training that CTDA will be able to build their own views and relay upon Support for any assistance needed.

## Reporting

**Removed from Scope.** Possible solutions are fully described in v. 1.0 of this report.

## User Content Lists

**Removed from Scope.** Possible solutions are fully described in v. 1.0 of this report.

## Set Access Control via CSV Ingest

**Removed from Scope.** Possible solutions are fully described in v. 1.0 of this report.

## Handle Service

The new *DGI Actions Handle* module will be configured to trigger the minting of Handles for new repository items on ingest.

## Additional Configuration and Customization

- To comply with Core Trust requirements DGI will:
    - Audit and add the Drupal Media Revisions UI module. (See https://www.drupal.org/project/media_revisions_ui)
    - Enhance checksum checking features. This may be done working with existing Filehash data that is stored and/or by implementing Islandora RipRap. (See https://github.com/mjordan/islandora_riprap.)

## Unsolicited Feature Guidance

- To improve SEO CTDA should consider configuring the Drupal PathAuto module (https://www.drupal.org/project/pathauto).

# Presentation Layer

DGI provides an Islandora 8 base theme that is WCAG Level 2 AA compliant. The theme will be customized to provide additional features to the user that are not present in its base form.

- **Collection search in the page header.** Adds capability to filter search by Member top level collections to the site header.
- **Collection search**. Adds capability to filter search by Member top level collections to the main content area search form on the Home page.
- **Member Collection Branding**. Member Group admins can manage some settings on their top level collection landing page: Hero image, descriptive text, logo, and color scheme.
- **Member Repository Item Branding**. Content in a Member group will inherit the top level collection color scheme.

CTDA has expressed interest in additional theme work during the course of the assessment:

- Changes to the presentation of compound repository items.*
- Changes to the presentation of Newspapers/Periodicals.

During the course of its continuing evaluation and adoption of Islandora 8 it is inevitable that further desired changes will arise, so a budget line item is allocated for this.

*There is already a new Compound display being released soon.

# Data Migration

## Migration of Collections and Repository Items

### New Supporting Works from DGI

DGI has completed new features in its I8 release that will facilitate the migration of CTDA content (and ongoing CTDA batch ingests):

- An I8 Repository Item content type that supports most MODS informational elements.
- An I8 Spreadsheet Ingest with a Migrate configuration for the Repository Item content type.
- An I7-to-I8 migration process that has a standardized workflow, yet allows a transformation step customizable for a given adopter's use of the MODS. The framework supports iterations as it can perform rollbacks, reruns, and updates.

### Migration Workflow

- An export of all MODS and RELS datastreams from the I7 repository is analyzed for MODS markup usage, relationship types, and content model usage.
- The I7 data analysis is used to create specifications for migration work and I8 configurations:
    - MODS markup usage is examined vv. the base migration mapping of MODS elements to I8 content type fields. Any elements requiring transformation to align with the I8 content type are noted and compiled into a specification for an XSLT stylesheet.
    - Repository Item fields that will not be used are noted so they can be disabled in the new environment for UI simplification and data quality control.
    - Authorities referenced by MODS elements for subjects, names, genres, forms, etc. are listed so they can be added to Authority Source lists for taxonomies in I8.
    - Any field usage that may require adjustment of the Repository Item content type is noted.
- Based on the preceding analysis and specification work:
    - Repository Item content type adjustments are performed.
    - Authority Sources are added to I8 taxonomies.
    - A transform is written to align the I7 MODS with the MODS form the migration process anticipates.
    - In the I7 repository a collection of test objects is created with all possible permutations of MODS markup and content models.
- Testing and iterative refinement is performed.
    - First the I7 test collection is migrated to I8 and the fidelity and accuracy of the migrate configuration is verified.
    - Next, selected collections in the I7 repository are migrated to further review the results and to benchmark the performance of the process. Results will drive potential infrastructure changes (which may be temporary) to improve performance, and be used to define the timetable for the migration work as a whole.
- Based on the plan emerging from the prior step, the cutover process will begin.

### Creation of Taxonomies

In I7 metadata for subjects, names, genres, forms, etc. is not referenced from controlled vocabularies within Islandora, and stores the content on every object. The data model in I8 is far more efficient in the respect

that such terms are managed as entities in Taxonomies and referenced from Repository Items (i.e. single sourced). The migration process will populate these taxonomies in the course of it's work, putting entity references into the Repository Items.

## Remapping Persistent Identifiers

A process for remapping the CTDA handle URIs to point to the repository items in Islandora 8 will be authored. On successful ingest of a migration batch, this process will be run.

All new items ingested into the CTDA repository will have handles minted as they have in Islandora 7.

## Non-Repository Content

See **Assumptions**.

## Domain Name Management Assistance

CTDA domain names will be remapped on the Go Live to the new Islandora 8 repository. For Members using multi sites or custom domains domains will be mapped to specific top level collection URLs.

## Cutover Strategy Options

CTDA's repository contents are quite extensive and processing time for migrations are lengthy. DGI will benchmark processing time early in the Migration testing so accurate timelines can be extrapolated.

It is not realistically possible to perform the migration of all Member collections at once, nor desirable as it would be impossible to determine when a given stakeholder could become active in the new repository. Migrating each set of Member Collections one-at-a-time is the most flexible approach.

# Assumptions, Risks, and Constraints

## Assumptions

- CTDA will be responsible for migration of the non-repository content.
- Either CTDA admins or Member Group Admins will configure the hero image and test for their collections (esp. top level).
- EREGS MODS submissions comply with the current CTDA MODS usage guidelines, i.e. will not change in format.
- The format of the CTHS Koha export is unchanged.
- CTDA would want to open source work that can serve the community at large.
- Gallery Systems can implement changes to the report for CTHS that are required for I8 spreadsheet ingest.
- Fedora is not employed by the solution.
- MySQL is used for the Drupal database (not PostgreSQL).
- See **Assumptions** under **Watermarking**.
- It's recognized that dgi and community features and issues in Islandora 8 are evolving at a rapid pace. As such scope and cost may decrease if features or performance enhancements are released that were not available at the time of this writing, or increase if undiscovered issues arise. dgi will alert CTDA to any events such as this that significantly impact scope or cost.

## Risks

- Performance testing of the access control may turn up performance issues.
- Migration process performance testing may turn up performance issues.
- Use case testing of access control may turn up further customization work.
- Running migration processes on the new repository while already migrated Members are ingesting new content can lead to degraded ingest performance.
- Drupal modules planned for inclusion in the technical architecture may have issues that turn up in auditing and testing.
- Third parties involved can introduce additional time and costs potentially.

## Constraints

- Work to effect any support for EAD finding aids in I8 is not in scope.
- Migration of existing I7 bookmarks is not in scope.
- Migration of I7 usage stats is not in scope.
- There is not at this time a means to migrate embargoes between I7 and I8. These will need to be documented and reapplied as content is migrated over to I8.
- Institutional Repository features are not in scope. (During the course of the assessment work on the DGI IR offering was halted, so looking at integration of its features into CTDA will be delayed.)
- For display of TEI transcripts on Paged Content repository item landing pages: Transcripts may/may not have page breaks indicated in them and there isn't currently a way to sync them with page navigation in the image viewer. In I8 the feature will display the transcript in a scrollable manner.

- Estimates for the following features are given without full scoping and as such may change when researched in the Implementation Planning phase:
    - Action for Bulk Application of Embargoes
    - Extracted Text Hit Highlighting for Paged Content

# Appendices

# A1: Digital Lifecycle Curation Support in Islandora 8

Islandora 8 provides a robust, secure, and flexible environment for meeting preservation and digital curation requirements. It's easiest to illustrate by discussing the points of how Islandora supports the OAIS Reference Model functional entities and processes. It should be easy enough to extrapolate a mapping of these supporting features to other frameworks from these examples.

What follows is more focused on Islandora 8's technical support rather than practices external to the repository (which vary between implementers). Nonetheless it should make clear how the technical features provide the appropriate and necessary points of engagement and interaction to support those as well.

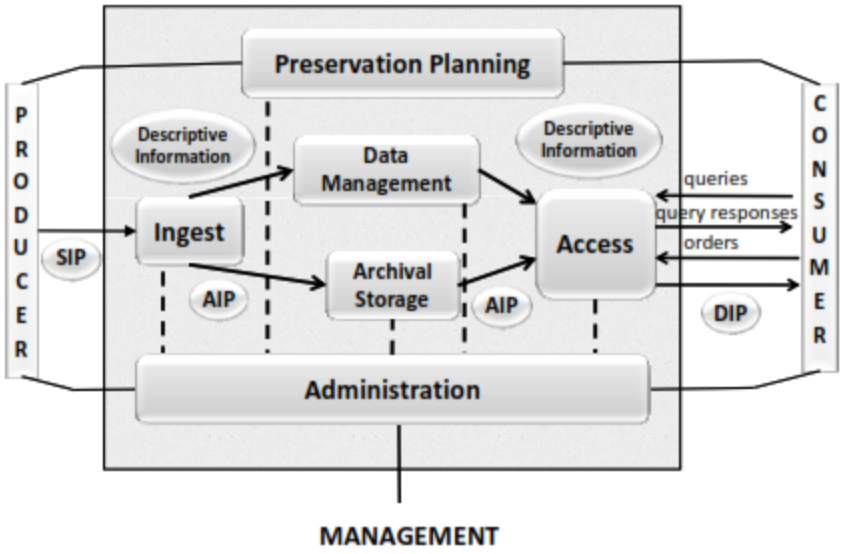



Figure 4-1: OAIS Functional Entities

**Ingest.** In standard form, Islandora provides three endpoints for ingest: Repository Item submission forms, Bulk ingest submission forms, and Application Programming Interfaces (APIs). Requests via submission forms or bulk ingest requests (which intake digital files and corresponding metadata in csv form) are subject to validation by the Content Type definition in use. The Content Type definition is configured per the implementer's metadata application profile (MAP). APIs are only used for custom processes, therefore features for validation are subject to a specific implementation. Upon ingest (or on modification) all components of the inbound Repository Item(s) are immediately placed in Archival Storage.

**Archival Storage.** Any time a new Repository Item is ingested, or an existing one modified, version(s) of all ingested or modified components are stored using Discoverygarden Islandora 8 AWS Preservation Storage. See Appendix A for a whitepaper providing full details of its permanent storage and recovery capabilities.

Implementers using other storage options can configure component storage and versioning per their offering constraints and requirements.

**Data Management.** The following features support data management:

- ❏ Population of the database and preservation storage via the endpoints described in **Ingest**.
- ❏ Interfaces for managing entity definitions such as Content Types and Taxonomies used to define Descriptive and Administrative Metadata.
- ❏ Management and validation of entity references to maintain their integrity.
- ❏ All metadata is indexed and exposable to views.
- ❏ Reports and Views may be defined to present content and/or related information stored in the database.
- ❏ Bulk metadata tools.

**Administration.** A full set of features supporting administrative requirements is present:

- ❏ Capability to solicit agreements for deposit.*
- ❏ Rights and License taxonomy support.
- ❏ Administrative metadata.
- ❏ Content validation (see **Ingest** above).
- ❏ Content moderation.
- ❏ Reporting, as for: Repository content, usage statistics, file formats, duplicate files, etc.

* May require mild customization depending on the use case.

**Preservation.** This is entirely a practical matter but the reader should review the other functional entities for their supporting features.

**Access.** Islandora provides a set of functions for providing and controlling access. Content discovery and interactions

- ❏ A highly scalable and performant search powered by Solr.
- ❏ Configurable search results display.
- ❏ Configurable Repository Item display (landing pages).
- ❏ Capability to make downloads of digital files or search results available.

Access controls include capability to:

- ❏ Integrate with Single Sign-On systems.
- ❏ Define user roles representing classes of users.
- ❏ Define Groups of content that can have members with varying levels of permissions.
- ❏ Permission all metadata fields by user role.
- ❏ Permissions access to Repository Items by such means as: Taxonomy Terms, Publication Status.
- ❏ Embargo Repository Items, and/or (just) their Digital File(s) by: IP, user role.
- ❏ Allow mediated access (as by request).

# A2: Drupal Modules Deployed with DGI Islandora 8

Note: This list is as of 2021-08-21. Updates are ongoing.

composer/installers": "^1.9",
discoverygarden/basic_ingest": "8.x-dev",
discoverygarden/breadcrumbs": "8.x-dev",
discoverygarden/content_sync_helper": "8.x-dev",
discoverygarden/dgi_actions": "8.x-dev",
discoverygarden/dgi_header": "2.x-dev",
discoverygarden/dgi_i8_base": "2.x-dev",
discoverygarden/dgi_i8_helper": "8.x-dev",
discoverygarden/dgi_migrate": "dev-master as 8.x-dev",
discoverygarden/dgi_standard_oai": "dev-8.x-dev",
discoverygarden/dgi_standard_spreadsheet_ingest": "8.x-dev",
discoverygarden/icons_facets_widget": "1.x-dev",
discoverygarden/islandora_spreadsheet_ingest": "dev-8.x-2.x as 2.x-dev",
drupal/addtoany": "*",
drupal/admin_toolbar": "3.0.0",
drupal/adminimal_admin_toolbar": "*",
drupal/adminimal_theme": "*",
drupal/better_exposed_filters": "^4",
drupal/betterlogin": "*",
drupal/block_class": "*",
drupal/blockgroup": "*",
drupal/bootstrap": "^3",
drupal/cas": "^1.7",
drupal/clamav": "*",
drupal/conditional_fields": "^1.0@alpha",
drupal/config_filter": "*",
drupal/config_inspector": "*",
drupal/config_override": "1.0.0-beta2",
drupal/config_override_warn": "*",
drupal/config_split": "*",
drupal/config_update": "*",
drupal/console": "^1.0.2",
drupal/content_browser": "*",
drupal/content_lock": "*",
drupal/content_sync": "dev-8.x-2.x-dgi as 2.x-dev",
drupal/context": "4.0-beta2",
drupal/context_groups": "*",
drupal/copyright_footer": "^1.7",
drupal/core-composer-scaffold": "^8.9",
drupal/core-recommended": "^8.9",
drupal/core-vendor-hardening": "^8.9",
drupal/ctools": "^3.6",
drupal/custom_add_another": "*",
drupal/diff": "*",
drupal/disable_user_1_edit": "^1.4",
drupal/embed": "*",
drupal/entity_browser": "^2",
drupal/entity_browser_enhanced": "*",
drupal/entity_embed": "*",
drupal/entity_reference_facet_link": "dev-master as 1.x-dev",
drupal/entity_route_context": "^1.3",
drupal/eu_cookie_compliance": "*",
drupal/eva": "*",
drupal/expand_collapse_formatter": "*",
drupal/facets": "^1.8",
drupal/features": "*",
drupal/field_group": "^3.1",
drupal/field_group_table": "^1.0@beta",
drupal/field_permissions": "*",
drupal/file_download_link": "^1.1",
drupal/file_mdm": "*",
drupal/filehash": "*",
drupal/flysystem": "*",
drupal/flysystem_s3": "*",
drupal/fontawesome": "*",
drupal/fontawesome_menu_icons": "^1.9",
drupal/geolocation": "^3.0",
drupal/gin": "3.0.0-alpha33",
drupal/gin_login": "^1.0@RC",
drupal/gin_toolbar": "1.0.0-beta14",
drupal/imagemagick": "*",
drupal/jwt": "1.0.0-beta5",
drupal/key": "*",
drupal/libraries": "*",
drupal/mail_edit": "*",
drupal/maxlength": "*",
drupal/memcache": "^2.3",

drupal/migrate_directory": "dev-8.x-1.x-dgi as 1.x-dev",
drupal/migrate_plus": "^4.2",
drupal/migrate_source_csv": "^2.1",
drupal/migrate_spreadsheet": "*",
drupal/migrate_tools": "*",
drupal/name": "dev-1.x as 1.x-dev",
drupal/node_edit_protection": "*",
drupal/nodeviewcount": "^1.0@alpha",
drupal/paragraphs": "1.x-dev",
drupal/pathauto": "^1.8",
drupal/pdf": "*",
drupal/prepopulate": "*",
drupal/range_slider": "^1.3",
drupal/request_data_conditions": "*",
drupal/rest_oai_pmh": "^1.0@beta",
drupal/restui": "*",
drupal/revision_log_default": "*",
drupal/scss_compiler": "1.x-dev@dev",
drupal/search_api": "^1.19",
drupal/search_api_solr": "^4.1",
drupal/search_autocomplete": ">=1.0 <1.2.0 || >1.2.0 <2.0 || >2.0.0 <3.0",
drupal/simple_sitemap": "^3.8",
drupal/simplesamlphp_auth": "^3.2",
drupal/sitemap": "*",
drupal/sophron": "*",
drupal/taxonomy_menu": "*",
drupal/token": "*",
drupal/user_current_paths": "^2.0",
drupal/verf": "^1.0",
drupal/view_mode_selector": "*",
drupal/views_autocomplete_filters": "*",
drupal/views_autosubmit": "^1.2",
drupal/views_base_url": "^1.0",
drupal/views_bootstrap": "*",
drupal/views_bulk_edit": "*",
drupal/views_bulk_operations": "*",
drupal/views_exposed_filter_blocks": "*",
drupal/views_field_view": "*",
drush/drush": "^10.3",
fsulib/embargoes": "8.x-1.x-dev",
iqb/substream": "dev-master",
islandora/controlled_access_terms": "^1.1.0",
islandora/islandora": "dev-8.x-1.1.1-hotfixes as 1.1.1",
islandora/islandora_defaults": "^1.1.0",
islandora/jsonld": "*",
islandora/openseadragon": "^1.1.0",
league/flysystem-aws-s3-v3": "1.0.25",
stomp-php/stomp-php": "4.*"
behat/mink-selenium2-driver": "dev-master as 1.3.x-dev",
drupal/core-dev":"^8.9",
drupal/devel": "*",
drupal/devel_php": "^1.3",
drupal/drupalmoduleupgrader": "8.x-1.x-dev",
kint-php/kint": "*",
phpdocumentor/reflection-docblock": "2.0.*"

# A3: Known Limitations of Fedora Implementation

Fedora 3.x supported rebuilding indexes on the fly if the Fedora database was ever out of sync from the data in NFS. This is particularly important in disaster recovery scenarios.

Fedora 5 can have issues scaling with larger repositories that rely on incremental backups of large detached NFS mounts. Database dumps will likely not be in sync with the incremental NFS backups so having an export of your repository becomes needed. In Fedora5 documentation they describe the backup/restore method which essentially is a full export of your repository (https://wiki.lyrasis.org/display/FEDORA51/Backup+and+Restore).

While this would work fine on smaller repositories this could represent a scaling issue with large repositories and end up using a lot more disk space than necessary to maintain the exported backup. It is possible that other more granular export/import measures could be explored such as using https://github.com/fcrepo-exts/fcrepo-import-export; however, it still doesn't allow you to rebuild your indexes from scratch in a disaster scenario.

Fedora 6 once again allows indexes that can be easily rebuilt based on whatever data exists in the Oxford Common Filesystem Layout (OCFL) file system https://wiki.lyrasis.org/display/FEDORA6x/Rebuild+Fedora+Indices. OCFL is a significant step forward for Fedora and has a lot of promise.

Fedora 6 however is still quite new. They have a number of known issues that can be found https://jira.lyrasis.org/browse/FCREPO-3260?filter=15700. Early adoption in a production environment of considerable scale - particularly in light of a solution already meeting requirements - presents implementation risk and introduces complexity.

Perhaps one of the most concerning limitations would be that of restoring from OCFL. Islandora 8 does not store the repository item metadata in XML (as in Islandora 7). It is stored in the Drupal database. The use case to restore an Islandora 8 repository from Fedora cannot be met at this time. A lossless means to crosswalk and restore descriptive metadata and versioning information back to the Drupal information model simply does not exist.

## Additional Considerations

### Multisites

We currently do not have the means to set up Drupal multi sites with Fedora and Islandora8. This is an active discussion point in the community with a number of solutions being explored. Multisites can be done without Fedora.

### Milliner JSON-LD representation of objects

Milliner (https://github.com/Islandora/Crayfish/tree/dev/Milliner) uses JSON-LD representation of the specified Drupal entity and inserts it in Fedora. It is possible that some desired Drupal information might not be making it into Fedora if the corresponding configuration is not in place. Clients should review information

going into Fedora to ensure that desired information is being stored. Further customizations might be required.

### TN are being stored in the public file system

When using Fedora, TNs and likely other derivatives don't seem to be able to be stored in the private file system. Generally, with most community configurations we have the original binary being stored in Fedora and the TN and derivatives being stored in the Drupal public file system. This may be undesirable if some TN/derivatives are intended to be private. This will require additional development time to fix. In our DGI non-Fedora installs these files are stored in S3 repo-service or in the local file system using Flysystem.

### Older version of Apache Karaf

We attempted to get the Fedora related Crayfish components working with the latest Karaf that DGI uses in our non-Fedora installations but it does not work. Both ISLE and the community ansible playbook are still on version 4.0.8 which does work.

The issue with version 4.0.8 is that it is vulnerable to https://www.cvedetails.com/cve/CVE-2018-11786/

As long as the box remains properly firewalled the risks are somewhat mitigated.

# A4: Technology Stack Components

**NOTE:** For any standard OS packages assume latest available from apt/yum repos, unless specified otherwise.

**Flysystem** - Flysystem is a PHP file system abstraction layer, offering consistent interfaces (via stream wrappers) to various storage providers from PHP such as AWS S3 and Fedora. We make use of Flysystem to connect to S3 or local storage.

- https://www.drupal.org/project/flysystem - 8.x-1.0+
- https://www.drupal.org/project/flysystem_s3 - 2.0.0-rc1+
- https://flysystem.thephpleague.com/v1/docs/adapter/local/
- https://flysystem.thephpleague.com/v1/docs/adapter/aws-s3-v3/
- https://www.drupal.org/docs/8/modules/islandora/developer-documentation/flysystem

**Karaf** - Apache Karaf open source OSGi runtime environment. This is used to run Crayfish connectors and islandora-connector-derivative. For non-Fedora we use the latest Karaf 4.x +. (See https://karaf.apache.org/.)

**Alpaca** - Event-driven middleware based on Apache Camel that synchronizes a Fedora repository with a Drupal instance. We don't use most of Alpaca unless Fedora is being used. For non-Fedora installations we only make use of the islandora-connector-derivative microservice which runs in Karaf alongside the Crayfish connectors. Islandora-connector-derivative service receives requests from Drupal when it wants to create derivatives and passes that request along to a microservice in Crayfish. When it receives the derivative file back from the microservice, it passes the file back to Drupal. (See https://github.com/Islandora/Alpaca.)

**Crayfish** - Collection of microservices/connectors that run in Karaf and make use of Apache/PHP. The microservices used can vary depending on if Fedora and Blazegraph are being used.

- **Houdini** - ImageMagick as a microservice.
  - imagemagick
- **Hypercube** - Tesseract as a microservice.
  - tesseract-ocr
  - poppler-utils
- **Homarus** - FFmpeg as a microservice.
  - ffmpeg
- **Gemini** - A path mapping microservice to align resources in Drupal and Fedora. (slated to be replaced). Used only with Fedora install.
- **Milliner** - Microservice that converts Drupal entities into Fedora resources. Used only with Fedora install.
- **Recast** - Microservice that remaps Drupal URIs to add Fedora to Fedora links based on associated Drupal URIs in RDF. Used only with Fedora install.

**Apache / PHP** - Apache web server and PHP are needed for Drupal and Crayfish. We make use of the following Apache and PHP packages:

- php (7.4+)
- php-cli
- php-curl
- php-gd
- php-pgsql
- php-xml
- php-mbstring
- apache (2.4.x+)

**Database** - A database server is required for the Drupal database. On a standalone single server install we make use of postgresql-10 package. In the case of CTDA we may make use of MySQL.

**JMS broker** - A JMS broker that can receive STOMP messages is needed to receive derivative creation requests from Islandora. These requests are picked up by islandora-connector-derivative and sent to Crayfish which generates the derivatives. We make use of activemq latest 5.x.

**Solr** - Solr is an open-source enterprise-search platform, written in Java, from the Apache Lucene project. We use the https://www.drupal.org/project/search_api from Drupal to create an index within the latest Solr 8.x+. On a larger installation, we may make use of Solr Cloud and ZooKeeper to create a scalable, fault tolerant search platform. This however will use a significant number of resources it may make sense to start out with a single solr server.

**Cantaloupe** - IIIF image server used to generate tiles used with JP2 viewing. We make use of cantaloupe 4.x+ (See https://cantaloupe-project.github.io/.)

**Tomcat / Jetty** - Tomcat and/or Jetty are web servers used to run Java applications. We currently use the latest Tomcat 9.x for Cantaloupe, Blazegraph, and Fedora. Solr makes use of Jetty.

**Drupal** - Drupal is a widely used web content management system that runs in Apache (or other types of web servers that supports PHP). It needs to run on a “Front-end” system that has HTTP/HTTPs exposed to its desired user base. By default we will redirect all HTTP to HTTPS. Drupal has its own database (or multiple databases if it is setup to use multisites) that it will need to access on a Postgres database server. Latest 8.x Drupal release will be used in this install. DGI is actively working on an update to utilize Drupal 9.

**Triplestore -** A triplestore (also referred to as a Resource index or RDF database) is a purpose-built database for the storage and retrieval of triples through semantic queries. A triple is a data entity composed of subject-predicate-object, like "Bob is 35" or "Bob knows Fred".

The Triplestore is NOT required by Islandora in order to function. As you ingest and modify objects in Islandora will update the triplestore accordingly then in turn it will be able to run SPARQL queries against the triplestore in order to retrieve meaningful information about your data. This install will utilize Blazegraph version 2.x

**Islandora** - Islandora is an open-source framework that provides the necessary tools to turn a Drupal website into a fully-functional Digital Assets Management System. It is a suite of modules/dependencies that operate within the Drupal ecosystem. This install will make use of the latest Islandora 8 release (currently

1.1.1). Discoverygarden currently operates on a fork and is looking to work with the community to integrate our improvements upstream. We do not plan on using the fork long term.

**Content-sync -** In I8, metadata is not generally kept in files, but is in the nodes and media entities in the Drupal DB.

DGI has greatly expanded the functionality of the Drupal Content Synchronization module to support import and export individual components of assets.

Built-in, content_sync has the ability to perform various imports and exports, e.g full site, single item. Full site operations will definitely be useful, but the included entailment of single items to all their dependencies is powerful; however, they do not expose it for easy use. In order to get things to the level needed, DGI created a content_sync_helper module, which of particular importance exposes an action which can be used to export an entity, which can then be added into I8's index/derivative/delete workflows in order to have content exported on mutation synchronously, *maintaining the consistency of the "external" storage*.

When any modification is made to metadata it will export the changes out to "**repo-meta**" using content sync. This functionality can be configured at /admin/structure/context.

**Note:** Content-sync can make ingest jobs take longer. In some cases, you may want to disable it. For example, you could disable content-sync before a large mass migration. Once the large job is complete, you would then re-enable it and do a full export. Then it can keep things in sync going forward.