

October 29, 2020

Dear valkyrie,

Do I have to die to  
enter Hyrax-halla?

Sincerely,

E. Lynette Rayle



Samvera Connect

2020 Zoom Mtg

Santa Barbara, CA

93106

P.S. A journey to valkyrizing our Hyrax models

# USDA ESMIS site as test platform for migrating to Valkyrie





United States Department of Agriculture  
Economics, Statistics and Market Information System



## Search Publications

BROWSE BY  
AGENCY ▼ CATEGORY ▼ ALL PUBLICATIONS

[Home](#) [Release Calendar](#) [About](#) [Help](#) [Contact](#) [API](#)

The USDA Economics, Statistics and Market Information System (ESMIS) contains over 2,100 publications from five agencies of the U.S. Department of Agriculture (USDA). These materials cover U.S. and international agriculture and related topics. Mann Library at Cornell University developed and maintains this site.

© 2017 USDA Economics, Statistics and Market Information System

### Partners

- Agricultural Marketing Service (AMS)
- Economic Research Service (ERS)
- Foreign Agricultural Service (FAS)
- National Agricultural Statistics Service (NASS)
- World Agricultural Outlook Board (WAOB)

Albert R. Mann Library, Cornell University  
United States Department of Agriculture

### Developed & supported by:



**Albert R. Mann Library**  
Cornell University  
Ithaca, NY 14853-4301  
Email: [usda-help@cornell.edu](mailto:usda-help@cornell.edu)

1	<b>Projected Work for Migrating to Valkyrie Postgres</b>								
2									
3	Migrate AF Work Models to Valkyrie::Resource								
4	Generate a work resource								
5	Define attribute metadata								
6	Test resource attributes								
7	Handle extra custom code								
8	Custom Query to find by identifier field								
9	Custom Change Sets								
10	Test using Persister to save								
11	Test using QueryService to read								
12	Migrate Custom Indexing								
13	Verify fields from metadata yaml are indexed as specified								
14	Add custom solr fields								
15	Test solr document generation								
16	Test writing to Valkyrie Solr Core								
17	Test using WorkUpdate transaction for save to Fedora & Solr								
18	Verify same fields are generated by Valkyrie indexer								
19	New Form								
20	Use default generated form to save a work								
21	Verify fields are required based on metadata yaml								
22	Verify fields are above the fold when primary in metadata yaml								
23	Verify fields are allow multiple values based on metadata yaml								
24	Verify that form fields work as expected								
25	Verify that can select and upload a file								
26	Verify that can share the work								
27	Verify that can set relationships								
28	Verify that fields are validated based on metadata yaml								
29	Add UI customizations & re-verify								
30	Verify removed fields are removed								
31	Can create custom ChangeSets for custom validations								

Issue #4556 all fields are multi-value after roundtrip

Issue #4487 solr docs not an exact match

works in Hyrax, so likely due to a clash with a customization  
same here

Hyrax::ChangeSets do not create validations

32	Work Display View								
33	New form forwards to work view								
34	Displays work created in new form								
35	Displays work created outside of new form								
36	Displays same fields for resource as for AF object								
37	Presenters								
38	Verify that existing presenter is used								
39	May need to copy existing presenter to resource namespace								
40	SearchBuilders								
41	Determine if search builders continue to work								
42	Migrate AF Collection Models to Valkyrie::Resource								
43	Migrate AF AdminSets to Valkyrie::Resource								
44	Migrate AF FileSets to Valkyrie::Resource								
45	Migrate AF Files to use Valkyrie::Storage								
46	Refactor API to use resource controllers								
47	ActiveFedora references in app code								
48	Refactor ActiveFedora::SolrService (34)								
49	Refactor ActiveFedora::ObjectNotFoundError (7)								
50	Refactor ActiveFedora::Base.where (1)								
51	Refactor ActiveFedora::SolrQueryBuilder (1)								
52	Refactor ActiveFedora::RecordInvalid (1)								
53	Refactor ActiveFedora::Cleaner in specs (13)								
54	ActiveFedora reference in locale translations								
55	hyrax.en.yml								
56	ActiveFedora references in initializers								
57	Refactor ActiveFedora::Base.exists? in noid (1)								
58	Refactor ActiveFedora::Base.gone? in noid (1)								
59	Refactor ActiveFedora::WithMetadata in redirect_mime_type (1)								
60	Fully Valkyried Tests								
61	Read object created with ActiveFedora using Valkyrie Resource								
62	Migrate Data from ActiveFedora to Postgres								
63	Read from AF with Wings adapter								
64	Save to Postgres using Valkyrie Postgres adapter								

Issue #4559 can't display resource



# Documentation of the Valkyriorization process



[https://elrayle.github.io/samvera\\_docs](https://elrayle.github.io/samvera_docs)

Tells the story of  
our journey  
moving toward  
Valkyrie in Hyrax



Valkyriorizing Hyrax Works ▲
Overview Valkyriorizing a Hyrax Work
Generating a Work Resource
Defining Attribute Metadata for the Work Resource
Dealing with extra code in pre-3.0 Active Fedora model
Custom Change Sets
Understanding Valkyrie Indexing
Creating a Valkyrie Indexer for the Work Resource
Writing Custom Queries
New and Edit Forms for the Work Resource
Appendix - Solrizer Reference
Appendix - Stumbling Blocks
Appendix - Performance

## Overview Valkyriorizing a Hyrax Work

Hyrax v3.0 ( branch: )

### Table of Contents

- [Purpose](#)
- [Process](#)

### Purpose

The goal of this effort is to convert `ActiveFedora::Base` works to `Valkyrie::Resource` works to determine how well Hyrax code can function with Valkyrie works and identify areas that still need adjustment.

This is not a polished 'how to' guide. This is a description of the process we followed to explore moving to Hyrax 3.0 using exclusively `Valkyrie::Resource` works, and later collections, filesets, and files.

- This is an anecdotal exploration of the process. Code is improving regularly. Some of the stumbling blocks may have been resolved.

### Process

- [Generating a Work Resource](#)
- [Defining Attribute Metadata for the Work Resource](#)

# Introduction to Valkyrie



<https://github.com/samvera/valkyrie/wiki/Dive-into-Valkyrie>

## Dive into Valkyrie

E. Lynette Rayle edited this page 18 seconds ago · 8 revisions

Edit

New Page

### Introduction

*Under Construction - Tutorial completed through Validating Change Sets.*

This is a demo that touches the major concepts of Valkyrie. It builds a demo app using the Postgres adapter. It does not cover everything you will want to know when building a Valkyrie based app. Refer to the [Valkyrie documentation](#) for more information.

- [Concepts of Valkyrie](#)
- [Setting up the demo app](#)
- [Setting up Postgres](#)
- [Configuring app to use Valkyrie with Postgres](#)
- [Understanding resources](#) (Define and test first resource)
  - [Defining more resources](#)
  - [Creating-relationships-between-resources](#)
  - [Saving a resource](#)
  - [Retrieving resources](#)
- [Understanding change sets](#)
  - [Defining change set properties](#)
  - [Persisting changes through synchronization](#)
  - [Defining change set validations](#)
  - [Validating change sets](#)
  - [Using change sets with forms](#)

A [completed version of the app](#) built by this tutorial is available. Commits correspond to each of the lessons that create parts of the app.

▼ Pages **38**

Find a Page...

[Home](#)

[Architecture](#)

[Benchmarks](#)

[ChangeSets and Dirty Tracking](#)

[Combining Persistence & Indexing](#)

[Community Sprint Report #2 \(September 2017\)](#)

[Concepts of Valkyrie](#)

[Configuring app to use Valkyrie with Postgres](#)

[Creating relationships between resources](#)

[Custom Indexing](#)

[Defining a New Model and Attaching it to the Valkyrie Demo App](#)

[Defining change set properties](#)

[Defining change set](#)



# Quick and Dirty Performance Analysis

The times in the table are rough wall clock times.

TOTAL = the sum of 10 runs of a command

AVERAGE = TOTAL / 10

Running Benchmarks	
----- CREATING	
10 Create Publications (ActiveFedora)	6.67ms
10 Create PublicationResources (Valkyrie)	0.04ms
----- SAVING	
10 Save Publications with ActiveFedora	682.98ms
10 Save PublicationResources with wings	5,801.27ms
10 Save PublicationResources in memory	0.50ms
10 Save PublicationResources with postgres	16.95ms
----- READING	
10 Read Publications with ActiveFedora	29.07ms
10 Read PublicationResources with wings	385.87ms
10 Read PublicationResources from memory	0.01ms
10 Read PublicationResources from postgres	1.54ms

The commands being run for the performance tests are:

*NOTE: Publication is an ActiveFedora::Base model*

```
----- CREATING
af_pub1      = Publication.new(title: ["af pub 1"])
wings_val_pub1 = PublicationResource.new(title: ["wings val pub 1"])
mem_val_pub1  = PublicationResource.new(title: ["memory val pub 1"])
pg_val_pub1   = PublicationResource.new(title: ["postgres val pub 1"])

----- SAVING
af_pub1.save
wings_id = Hyrax.persister.save(resource: wings_val_pub1).id
mem_id   = memory_persister.save(resource: mem_val_pub1).id
pg_id    = postgres_persister.save(resource: pg_val_pub1).id

----- READING
ActiveFedora::Base.find(af_pub1.id)
Hyrax.query_service.find_by(id: wings_id)
memory_query_service.find_by(id: mem_id)
postgres_query_service.find_by(id: pg_id)
```

[https://elrayle.github.io/samvera\\_docs/valkyrie-work-appendix-performance.html](https://elrayle.github.io/samvera_docs/valkyrie-work-appendix-performance.html)



Come join us at the upcoming Developers Congress  
to help move this work forward

Developer Congress - November 16-18 2020





## Understanding Differences in Processing

*Persistence in ActiveFedora vs. Valkyrie*

## *Persisting - differences in processing*



ActiveFedora: `work_object.save`

Valkyrie: `Hyrax.persister.save(resource:  
work_resource)`



## *Persisting - differences in processing*

ActiveFedora: `work_object.save`

Valkyrie: `Hyrax.persister.save(resource:  
work_resource)`

- ActiveFedora saves the work to Fedora and to Solr
- Valkyrie saves the work to the data store only (e.g. Fedora)



Understanding Differences in Processing

*Indexing in ActiveFedora vs. Valkyrie*



## *Indexing - differences in processing*

ActiveFedora: `work_object.save`

Valkyrie: `Hyrax.index_adapter.save(resource:  
work_resource)`





## *Indexing - differences in processing*

ActiveFedora: `work_object.save`

Valkyrie: `Hyrax.index_adapter.save(resource:  
work_resource)`

- ActiveFedora saves the work to Fedora and to Solr
- The index\_adapter generates solr\_doc and saves to solr only



## Understanding Differences in Processing

*Modifying an Object in ActiveFedora vs. Valkyrie*

## *Modifying an Object - differences in processing*



ActiveFedora:

```
work_object.title = ['My New Title']
```

Valkyrie:

```
work_changeset = Hyrax::ChangeSet.for(resource: work_resource)
work_changeset.title = 'My New Title'
work_resource = work_changeset.sync if work_changeset.valid?
```

# *Modifying an Object - differences in processing*



ActiveFedora:

```
work_object.title = ['My New Title']
```

Valkyrie:

```
work_changeset = Hyrax::ChangeSet.for(resource: work_resource)
work_changeset.title = 'My New Title'
work_resource = work_changeset.sync if work_changeset.valid?
```

You can set the title directly on the work\_resource. But the change sets give you two advantages:

- validation of multiple changes before persisting
- the ability to not go through with a change if validation fails

## *Full process to make a change*



ActiveFedora:

```
work_object.title = ['My New Title']  
work_object.save
```

Valkyrie:

```
work_changeset = Hyrax::ChangeSet.for(resource: work_resource)  
work_changeset.title = 'My New Title'  
work_resource = work_changeset.sync if work_changeset.valid?  
Hyrax.persister.save(resource: work_resource)  
Hyrax.index_adapter.save(resource: work_resource)
```



## *Transactions for saving to data-store and solr*



```
work_changeset = Hyrax::ChangeSet.for(resource: work_resource)
work_changeset.title = 'My New Title'
Hyrax::Transactions::UpdateWork.new.call(work_changeset)
```

## *Transactions for saving to data-store and solr*



```
work_changeset = Hyrax::ChangeSet.for(resource: work_resource)
work_changeset.title = "new_title"
Hyrax::Transactions::UpdateWork.new.call(work_changeset)
```

- Sets modified\_date
- Sets uploaded\_date
- Validates the change set
- Syncs the change set to a resource
- Save the resource
- Calls `Hyrax.publisher.publish('object.metadata.updated'...)`  
which will call the indexer and save to solr



Understanding Differences in Processing

*Querying in ActiveFedora vs. Valkyrie*

```
Valkyrie:                                     resource =
Hyrax.query_service.find_by(id: work_id)
```

## Querying - differences in processing

```
ActiveFedora:  object = ActiveFedora::Base.find(work_id)
```

```
Valkyrie: resource =  
Hyrax.query_service.find_by(id: work_id)
```

Many other methods for querying in ActiveFedora that often search in Solr and return the object from Fedora. SearchBuilders are a major way of creating custom queries for Solr.



## Querying - differences in processing

```
ActiveFedora:  object = ActiveFedora::Base.find(work_id)
```

```
Valkyrie: resource =
Hyrax.query_service.find_by(id: work_id)
```

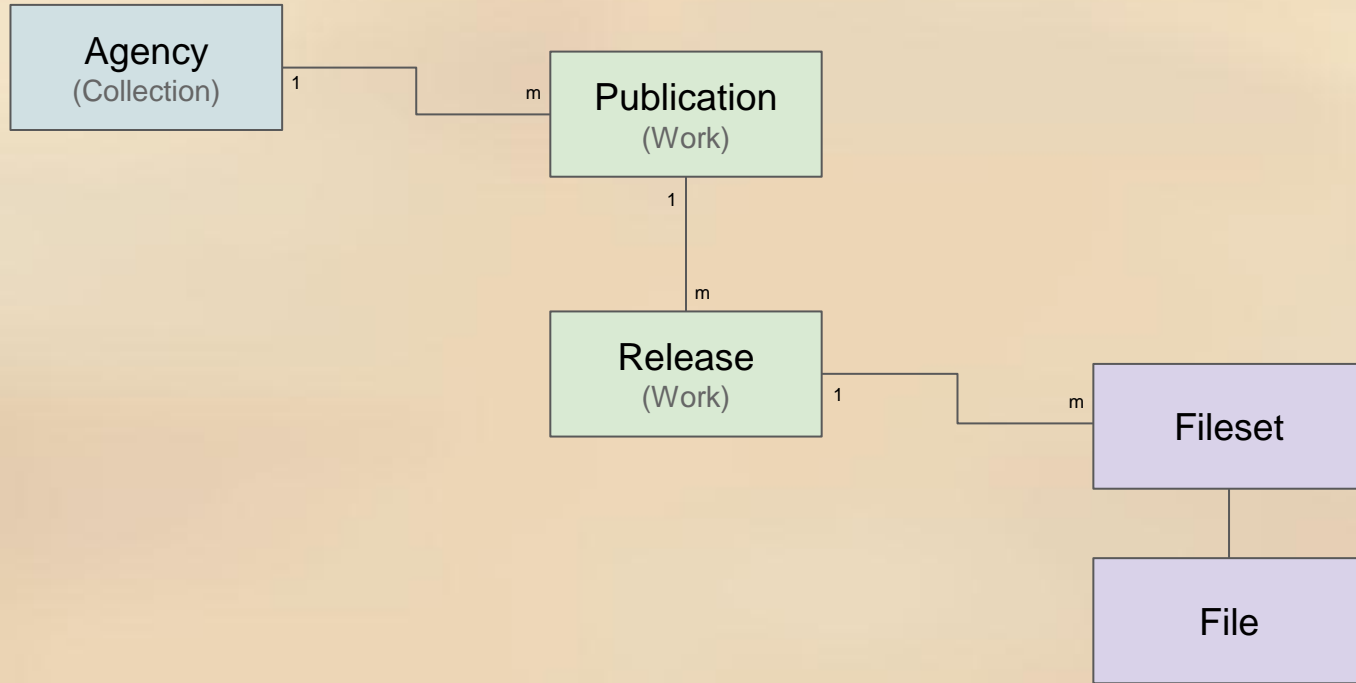
Many other methods for querying in ActiveFedora that often search in Solr and return the object from Fedora. SearchBuilders are a major way of creating custom queries for Solr.

Valkyrie query service supports many other methods. CustomQueries are a major way of extending Valkyrie's ability to search.



Migrating ActiveFedora::Base model to Valkyrie::Resource model

# Background on our models



# ActiveFedora::Base model

```
# Generated via
# `rails generate hyrax:work Publication`
# rubocop:disable Metrics/ClassLength
class Publication < ActiveFedora::Base
  # This must come after the WorkBehavior because it finalizes the metadata
  # schema (by adding accepts_nested_attributes)
  include ::Hyrax::WorkBehavior

  self.indexer = PublicationIndexer

  # Change this to restrict which works can be added as a child.
  self.valid_child_concerns = [:Release]
  validates :title, presence: { message: 'Your work must have a title.' }

  # self.human_readable_type = 'Publication'

  ### START esmis publication metadata properties

  # title inherited from BasicMetadata

  # previous title
  property :previous_title, predicate: :RDF::Vocab::DC::replaces do |index|
    index.as :stored_searchable, :facetable
  end

  # authors
  property :authors, predicate: :RDF::Vocab::MARC::Relators::aut do |index|
    index.as :stored_searchable, :facetable
  end

  # description inherited from BasicMetadata

  # agency identifier (e.g. slug_id, other string assigned by agency), identifier inherited from CC BasicMetadata
  validates_with IdentifierValidator

  # resource_type inherited from CC BasicMetadata

  # System uniq identifier is the URI for the resource

  # External Identifier: ISSN
  property :issn, predicate: :RDF::Vocab::BIBO::issn, multiple: false do |index|
    index.as :stored_searchable
  end

  # Publishing Agency (get from agency)

  # Location of Publication

  # Contact name (a person)
  property :contact_name, predicate: :RDF::Vocab::VCARD::hasFN, multiple: false do |index|
    index.as :stored_searchable
  end

  # Contact office (an organization)
  property :contact_organization, predicate: :RDF::Vocab::VCARD::hasOrganizationName, multiple: false do |index|
    index.as :stored_searchable
  end

  # Contact Email
  property :contact_email, predicate: :RDF::Vocab::VCARD::hasEmail, multiple: false do |index|
    index.as :stored_searchable
  end

  # Contact Phone
  property :contact_phone, predicate: :RDF::Vocab::VCARD::hasTelephone, multiple: false do |index|
    index.as :stored_searchable
  end

  # ESMIS Keywords - Should this be a SKOS concept?
  property :keywords, predicate: :RDF::Vocab::DC::subject do |index|
    index.as :stored_searchable, :facetable
  end

  # Frequency of Release
  property :frequency, predicate: :RDF::URI.new("http://daregistry.info/Elements/u/P60F36"), multiple: false do |index|
```

```
# Status
property :status, predicate: :RDF::URI.new("http://daregistry.info/Elements/u/P60I29"), multiple: false do |index|
  index.as :stored_searchable, :facetable
end

property :upcoming_releases, predicate: :RDF::Vocab::DC::available do |index|
  index.as :stored_searchable, :facetable
end

# Local namespace for subscribable field
property :subscribable, predicate: :RDF::URI.new("http://usda.mannlib.cornell.edu/ns#subscribable"), multiple: false do |index|
  index.as :stored_sortable, :facetable
end

# Local namespace for related_publication field
property :related_publication, predicate: :RDF::URI.new("http://usda.mannlib.cornell.edu/ns#related_pubid"), multiple: true do |index|
  index.as :symbol
end

### END esmis publication metadata properties

# This must be included at the end, because it finalizes the metadata
# schema (by adding accepts_nested_attributes)
include ::Hyrax::BasicMetadata

### START esmis publication helper methods
def self.find_by_identifier(identifier)
  results = Publication.where(identifier: [identifier])
  raise ActiveFedora::ObjectNotFoundError, "Publication with identifier '#{identifier}' not found." unless results.count.positive?
  results.first
end

# Get the current agency for the publication
# @returns [Collection] the current agency or nil if not assigned
def agency
  cols = member_of_collections
  agency = nil
  return nil if cols.blank?
  cols.each do |col|
    agency = col if col.agency?
  end
  agency
end

# Get the current division for the publication
# @returns [Collection] the current division or nil if not assigned
def division
  cols = member_of_collections
  division = nil
  return nil if cols.blank?
  cols.each do |col|
    division = col if col.division?
  end
  division
end

# Change the agency for the publication
# @param [Collection] the new agency
def agency=(new_agency)
  return if current_agency?(new_agency)
  remove_from_current_agency
  add_to_agency(new_agency)
end

# Change the division for the publication
# @param [Collection] the new agency
def division=(new_division)
  return if current_division?(new_division)
  remove_from_current_division
  add_to_division(new_division)
end

# rubocop:disable Metrics/CyclomaticComplexity
# Check if the publication is already in new_agency
# @return [Boolean] true if already in new_agency OR publication is not in an agency and new_agency is also nil; otherwise, false
```





# *Major components of the ActiveFedora::Base model*

- property definitions
  - move to metadata yaml file
- validations
  - move to change sets
- helper methods
  - primarily used for navigating and maintaining the model structure
  - move to module include (at least for now)





Migrating ActiveFedora::Base model to Valkyrie::Resource model

*Generate the Work Resource*

## Generating a Work Resource



```
$ rails generate hyrax:work_resource PublicationResource
  info GENERATING VALKYRIE WORK MODEL: PublicationResource
create app/controllers/hyrax/publication_resources_controller.rb
create config/metadata/publication_resource.yaml
create app/models/publication_resource.rb
create spec/models/publication_resource_spec.rb
create app/forms/publication_resource_form.rb
insert config/initializers/hyrax.rb
create app/indexers/publication_resource_indexer.rb
create app/views/hyrax/publication_resources/_publication_resource.html.erb
create spec/views/publication_resources/_publication_resource.html.erb_spec.rb
```

## *What was generated*



- controller
- metadata yaml
- model (resource)
- indexer
- form
- view

## *What was generated*



- controller → spec
- metadata yaml
- model (resource) → spec
- indexer → spec
- form
- view → spec

## The Generated Resource



```
# frozen_string_literal: true

# Generated via
# `rails generate hyrax:work_resource PublicationResource`
class PublicationResource < Hyrax::Work
  include Hyrax::Schema(:basic_metadata)
  include Hyrax::Schema(:publication_resource)
end
```

## The Generated Resource



```
# frozen_string_literal: true

# Generated via
# `rails generate hyrax:work_resource PublicationResource`
class PublicationResource < Hyrax::Work
  include Hyrax::Schema(:basic_metadata)
  include Hyrax::Schema(:publication_resource)
end
```

## *The Final Resource model*



```
# frozen_string_literal: true

# Generated via
# `rails generate hyrax:work_resource PublicationResource`
class PublicationResource < Hyrax::Work
  include Hyrax::Schema(:basic_metadata)
  include Hyrax::Schema(:publication_resource)
  include PublicationExtras
end
```





Migrating ActiveFedora::Base model to Valkyrie::Resource model

*Define Attribute Metadata*

# ActiveFedora::Base model

```
# Generated via
# `rails generate hyrax:work Publication`
# rubocop:disable Metrics/ClassLength
class Publication < ActiveFedora::Base
  # This must come after the WorkBehavior because it finalizes the metadata
  # schema (by adding accepts_nested_attributes)
  include ::Hyrax::WorkBehavior

  self.indexer = PublicationIndexer

  # Change this to restrict which works can be added as a child.
  self.valid_child_concerns = [:Release]
  validates :title, presence: { message: 'Your work must have a title.' }

  # self.human_readable_type = 'Publication'

  ### START esmis publication metadata properties

  # title inherited from BasicMetadata

  # previous title
  property :previous_title, predicate: :RDF::Vocab::DC::replaces do |index|
    index.as :stored_searchable, :facetable
  end

  # authors
  property :authors, predicate: :RDF::Vocab::MARC::Relators::aut do |index|
    index.as :stored_searchable, :facetable
  end

  # description inherited from BasicMetadata

  # agency identifier (e.g. slug_id, other string assigned by agency), identifier inherited from CC BasicMetadata
  validates_with IdentifierValidator

  # resource_type inherited from CC BasicMetadata

  # System uniq identifier is the URI for the resource

  # External Identifier: ISSN
  property :issn, predicate: :RDF::Vocab::BIBO::Issn, multiple: false do |index|
    index.as :stored_searchable
  end

  # Publishing Agency (get from agency)

  # Location of Publication

  # Contact name (a person)
  property :contact_name, predicate: :RDF::Vocab::VCARD::hasFN, multiple: false do |index|
    index.as :stored_searchable
  end

  # Contact office (an organization)
  property :contact_organization, predicate: :RDF::Vocab::VCARD::hasOrganizationName, multiple: false do |index|
    index.as :stored_searchable
  end

  # Contact Email
  property :contact_email, predicate: :RDF::Vocab::VCARD::hasEmail, multiple: false do |index|
    index.as :stored_searchable
  end

  # Contact Phone
  property :contact_phone, predicate: :RDF::Vocab::VCARD::hasTelephone, multiple: false do |index|
    index.as :stored_searchable
  end

  # ESMIS Keywords - Should this be a SKOS concept?
  property :keywords, predicate: :RDF::Vocab::DC::subject do |index|
    index.as :stored_searchable, :facetable
  end

  # Frequency of Release
  property :frequency, predicate: :RDF::URI::new('http://rdaregistry.info/Elements/u#P60538'), multiple: false do |index|
```

```
# Status
property :status, predicate: :RDF::URI::new('http://rdaregistry.info/Elements/u#P60129'), multiple: false do |index|
  index.as :stored_searchable, :facetable
end

property :upcoming_releases, predicate: :RDF::Vocab::DC::available do |index|
  index.as :stored_searchable, :facetable
end

# Local namespace for subscribable field
property :subscribable, predicate: :RDF::URI::new('http://usda.mannlib.cornell.edu/ns#subscribable'), multiple: false do |index|
  index.as :stored_sortable, :facetable
end

# Local namespace for related_publication field
property :related_publication, predicate: :RDF::URI::new('http://usda.mannlib.cornell.edu/ns#related_pubid'), multiple: true do |index|
  index.as :symbol
end

### END esmis publication metadata properties

# This must be included at the end, because it finalizes the metadata
# schema (by adding accepts_nested_attributes)
include ::Hyrax::BasicMetadata

### START esmis publication helper methods
def self.find_by_identifier(identifier)
  results = Publication.where(identifier: [identifier])
  raise ActiveFedora::ObjectNotFoundError, "Publication with identifier '#{identifier}' not found." unless results.count.positive?
  results.first
end

# Get the current agency for the publication
# @returns [Collection] the current agency or nil if not assigned
def agency
  cols = member_of_collections
  agency = nil
  return nil if cols.blank?
  cols.each do |col|
    agency = col if col.agency?
  end
  agency
end

# Get the current division for the publication
# @returns [Collection] the current division or nil if not assigned
def division
  cols = member_of_collections
  division = nil
  return nil if cols.blank?
  cols.each do |col|
    division = col if col.division?
  end
  division
end

# Change the agency for the publication
# @param [Collection] the new agency
def agency=(new_agency)
  return if current_agency?(new_agency)
  remove_from_current_agency
  add_to_agency(new_agency)
end

# Change the division for the publication
# @param [Collection] the new agency
def division=(new_division)
  return if current_division?(new_division)
  remove_from_current_division
  add_to_division(new_division)
end

# rubocop:disable Metrics/CyclomaticComplexity
# Check if the publication is already in new_agency
# @return [Boolean] true if already in new_agency OR publication is not in an agency and new_agency is also nil; otherwise, false
```



# ActiveFedora::Base property statement



```
# External identifier: ISSN
property :issn, predicate: ::RDF::Vocab::BIBO.issn, multiple: false do |index|
  index.as :stored_searchable
end
```



## ActiveFedora::Base property statement

```
# External identifier: ISSN
property :issn, predicate: ::RDF::Vocab::BIBO.issn, multiple: false do |index|
  index.as :stored_searchable
end
```

### Key parts:

- property name (e.g. `:issn`)
- multiple (e.g. `false`)
- index.as (e.g. `:stored_searchable`)
- predicate (e.g. `::RDF::Vocab::BIBO.issn`)

## The Final Resource model



```
# frozen_string_literal: true

# Generated via
# `rails generate hyrax:work_resource PublicationResource`
class PublicationResource < Hyrax::Work
  include Hyrax::Schema(:basic_metadata)
  include Hyrax::Schema(:publication_resource)
  include PublicationExtras
end
```

# The Generated Metadata yaml



```
# Simple yaml config-driven schema which is used to define model attributes,  
# index key names, and form properties.  
#  
# Attributes must have a type but all other configuration options are optional.  
#  
# attributes:  
#   attribute_name:  
#     type: string  
#     multiple: false  
#     index_keys:  
#       - "attribute_name_sim"  
#     form:  
#       required: true  
#       primary: true  
#       multiple: false  
#  
# @see config/metadata/basic_metadata.yaml for an example configuration  
#  
# Generated via  
# `rails generate hyrax:work_resource <%= class_name %>`
```

```
attributes: {}
```

# The Final Metadata yaml



attributes:

...

issn:

type: string

multiple: true # TODO: Change this to false when Hyrax #4556 is resolved.

index\_keys:

- "issn\_tesim"

form:

required: false

primary: false

multiple: false

# predicate: ::RDF::Vocab::BIBO.issn

...





*Not covered by the property to attribute migration*

Relationships defined using

- directly\_contains
- indirectly\_contains

Do not have an equivalent structure in Valkyrie



## *Relationships through ID reference*

Include an attribute to hold the id of the related resource

Example in ReleaseResource

ActiveFedora:           indirectly\_contains :member\_of\_publication

Becomes:           publication\_id:  
                  type: id  
                  multiple: false  
                  index\_keys:  
                  - "publication\_id\_ssim"  
                  form:  
                  required: false  
                  primary: false  
                  multiple: false  
                  # predicate: ::RDF::Vocab::BF2.partOf



Migrating ActiveFedora::Base model to Valkyrie::Resource model

*Handle extra custom code*



## *Module include for custom methods*

```
# frozen_string_literal: true

# Generated via
# `rails generate hyrax:work_resource PublicationResource`
class PublicationResource < Hyrax::Work
  include Hyrax::Schema(:basic_metadata)
  include Hyrax::Schema(:publication_resource)
  include PublicationExtras
end
```



## *Types of Extra Custom Code*

- Navigation methods
  - get collection from publication (work)
  - get releases from publication (child works)
  - get publication from release (parent work)
- Maintaining relationships between models
  - add/remove publication in collection
  - add/remove release in a publication
  - set release's parent publication
- Finding a publication by identifier field
  - implemented with a custom query
  - requires publication to be indexed (implemented later in the process)



## *Extra Custom Code - Solutions*

- Navigation methods
  - use existing Valkyrie Custom Queries defined in Hyrax
- Maintaining relationships between models
  - create custom Valkyrie Change Sets to make the change, validate, and save
- Finding a publication by identifier field
  - create Custom Query



Migrate custom indexing

*Add custom indexing to generated indexer*



# Migration for Indexing



Copy custom indexing

FROM: `generate_solr_document` method

TO: `to_solr` method

## The generated indexer



```
# frozen_string_literal: true

# Generated via
# `rails generate hyrax:work_resource ReleaseResource`
class ReleaseResourceIndexer < Hyrax::ValkyrieWorkIndexer
  include Hyrax::Indexer(:basic_metadata)
  include Hyrax::Indexer(:release_resource)

  # Uncomment this block if you want to add custom indexing behavior:
  #   def to_solr
  #     super.tap do |index_document|
  #       index_document[:my_field_tesim] = resource.my_field.map(&:to_s)
  #       index_document[:other_field_ssim] = resource.other_field
  #     end
  #   end
end
```

## The generated indexer



```
# frozen_string_literal: true

# Generated via
# `rails generate hyrax:work_resource ReleaseResource`
class ReleaseResourceIndexer < Hyrax::ValkyrieWorkIndexer
  include Hyrax::Indexer(:basic_metadata)
  include Hyrax::Indexer(:release_resource)

  # Uncomment this block if you want to add custom indexing behavior:
  #   def to_solr
  #     super.tap do |index_document|
  #       index_document[:my_field_tesim] = resource.my_field.map(&:to_s)
  #       index_document[:other_field_ssim] = resource.other_field
  #     end
  #   end
end
```

# ActiveFedora Indexer



```
class ReleaseIndexer < Hyrax::WorkIndexer
  include Hyrax::IndexesBasicMetadata

  def generate_solr_document
    super.tap do |solr_doc|
      # START tap87 - custom ESMIS indexing behavior
      solr_doc['release_datetime_dtsi'] = object.release_datetime.to_datetime
      publication = object.publication
      if publication
        solr_doc['publication_id_ssi'] = publication.id
        solr_doc['agency_acronym_ssim'] = publication.agency_acronym
        solr_doc['division_acronym_ssim'] = publication.division_acronym
      end
      # END custom ESMIS indexing behavior
    end
  end
end
```

# The Final Indexer



```
class ReleaseResourceIndexer < Hyrax::ValkyrieWorkIndexer
  include Hyrax::Indexer(:basic_metadata)
  include Hyrax::Indexer(:release_resource)

  ### BEGIN tap87: Uncommented this block to add custom ESMIS indexing behavior
  def to_solr
    super.tap do |solr_doc|
      # START tap87 - custom ESMIS indexing behavior
      solr_doc['release_datetime_dtsi'] = object.release_datetime.to_datetime
      publication = resource.publication
      if publication
        # solr_doc['publication_id_ssi'] = publication.id
        solr_doc['agency_acronym_ssim'] = publication.agency_acronym
        solr_doc['division_acronym_ssim'] = publication.division_acronym
      end
      # END custom ESMIS indexing behavior
    end
  end
end
```

# The Final Indexer



```
class ReleaseResourceIndexer < Hyrax::ValkyrieWorkIndexer
  include Hyrax::Indexer(:basic_metadata)
  include Hyrax::Indexer(:release_resource)

  ### BEGIN tap87: Uncommented this block to add custom ESMIS indexing behavior
  def to_solr
    super.tap do |solr_doc|
      # START tap87 - custom ESMIS indexing behavior
      solr_doc['release_datetime_dtsi'] = object.release_datetime.to_datetime
      publication = resource.publication
      if publication
        # solr_doc['publication_id_ssi'] = publication.id
        solr_doc['agency_acronym_ssim'] = publication.agency_acronym
        solr_doc['division_acronym_ssim'] = publication.division_acronym
      end
      # END custom ESMIS indexing behavior
    end
  end
end
```



Migrate form customizations

*Exploring forms and display of works*

# The generated form class



```
# frozen_string_literal: true

# Generated via
# `rails generate hyrax:work_resource PublicationResource`
#
# @see https://github.com/samvera/hyrax/wiki/Hyrax-Valkyrie-Usage-Guide#forms
# @see https://github.com/samvera/valkyrie/wiki/ChangeSets-and-Dirty-Tracking
class PublicationResourceForm < Hyrax::Forms::ResourceForm(PublicationResource)
  include Hyrax::FormFields(:basic_metadata)
  include Hyrax::FormFields(:publication_resource)

  # Define custom form fields using the Valkyrie::ChangeSet interface
  #
  # property :my_custom_form_field

  # if you want a field in the form, but it doesn't have a directly corresponding
  # model attribute, make it virtual
  #
  # property :user_input_not_destined_for_the_model, virtual: true
end
```



## *What works in forms?* (with all our customizations removed)



- Form views are generated based on the metadata yaml
  - Required fields are marked required
  - Primary fields are above the fold
  - Multi-valued fields have links to add more values
- Can set fields
- Can upload files
- In Hyrax, can share with users and set relationships to collections
  - Not working in my app, but I suspect I didn't remove all related customization

*How about display of the work*



This currently raises an exception for which there is an open issue.



*This is where I am in the process so far!*

Come join us at the upcoming Developers Congress  
to help move this work forward

Developer Congress - November 16-18 2020