Chapter 3 - Getting Started with Islandora

This chapter presumes that you have access to a running instance of Islandora, either via the online sandbox, or because you have installed a **Virtual Machine Image** on your local machine. If you have a full running instance of Islandora with solution packs installed, this chapter will show you how to perform basic functions. If you do not have access to an instance of Islandora, please review Chapter 2 - Accessing Islandora.

Note that if you are running the **Virtual Machine Image (VM)**, you will have greater access to the back-end of your Islandora installation. However, this chapter focuses on explaining the way that Islandora understands content, and providing basic instructions for how you can create, view, update, and delete (purge) content in your repository.

Both the online sandbox, and the VM come with pre-installed collections for you to interact with. These instructions presume that you are either using one of these environments OR that you have a working instance of Islandora with **Solution Packs** installed.

How Islandora Understands Content

Before accessing and using Islandora, it is important to learn how Islandora understands your content. Islandora leverages Fedora's flexible and powerful Digital Object Model, Relationships, and **Content Model** architecture. If you are familiar with these concepts, you may find this section repetitive. The following section borrows heavily from the Fedora documentation on this subject.

The following is an attempt to distill this information down to what is most relevant when getting started with Islandora. Understanding Islandora means understanding the following:

- Everything is an Object
- Objects have **Datastreams**
- Objects have Relationships to one another
- Objects have **Persistent Identifiers (PIDs)** that you will be asked to specify

Everything is an Object

Everything in Islandora's Fedora Repository is an "object," made up of "Datastreams." When using Islandora, there are several special kinds of objects that perform different functions. The major types of Islandora "objects" you will encounter are Content Model Objects, Collection Objects, and Data Objects.

Content Model Object - A Content Model Object is a "template" for a particular type of content. Fedora has its own notion of a Content Model, which you can read more about here

Islandora extends Fedora's Content Model Architecture by envisioning an Islandora-specific Content Model, a content model with an ISLANDORACM Datastream. Think of a Content Model as a template or a recipe for data in your repository.

The Content Model Object answers particular questions about an object:

a) What will happen to an object when it is placed into (ingested) into the repository?

For example, when you put in a large image of a book page, the book content model will tell Islandora to create a number of smaller images (derivatives) used in display, and will perform an Optical Character Recognition (OCR) process so that the full text content of that page is stored alongside the image, and can be searched.

b) How will this object be displayed?

A map has very different display requirements from a book. For a map, users should be able to zoom in and out of the image to view details, or get an overall look at the object. For a book, images should be navigated so that users have the sense of turning pages. The Content Model tells objects how they should appear to a user.

c) What does this object consist of?

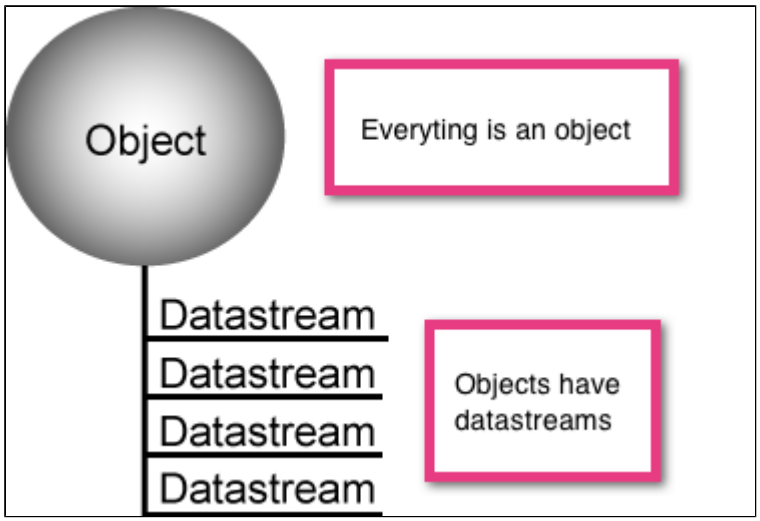
Content Models want to treat media types differently. For example, a PDF and a map might have very different data associated with them, based on the metadata conventions used for describing those kinds of objects, or what type of file it is. The Content Model object will define the list of Datastreams the associated digital object will have, as well as the list of binary files.

Collection Object - For most purposes, the term “Collection Object” can be understood as synonymous with the traditional concept of a collection. If you want to make a map collection in Islandora, you would start with a collection object that subscribes to the Map Content Model. Every data object you add to (ingest into) this collection would then be a member of that collection, because a relationship between that new data object and the collection object (the map collection) is written. Every instance of Islandora will have a root collection object. This collection object will subscribe to a “Collection” Content Model, enabling collections to be created “under” that object. All Collection objects will be affiliated with the Collection Content Model object, and that affiliation will define what types of Data Objects can belong to a Collection. That affiliation will be defined in one of the Datastreams of the collection object - the COLLECTION_POLICY stream.

Data Object - Here, the term “Data Object” refers to the actual assets being stored. So, one “map” and its associated Datastreams (metadata, alternate image formats/sizes) is a “data object.” The map object will have a number of different Datastreams. These Datastreams will tell the system where the main file is located, as well as any derivatives that are used for web-display, they will contain the bibliographic record (metadata) associated with that object, and they will indicate which collection (or collections!) the map belongs to.

Objects have Datastreams

Each object has Datastreams, which are the parts of an object.

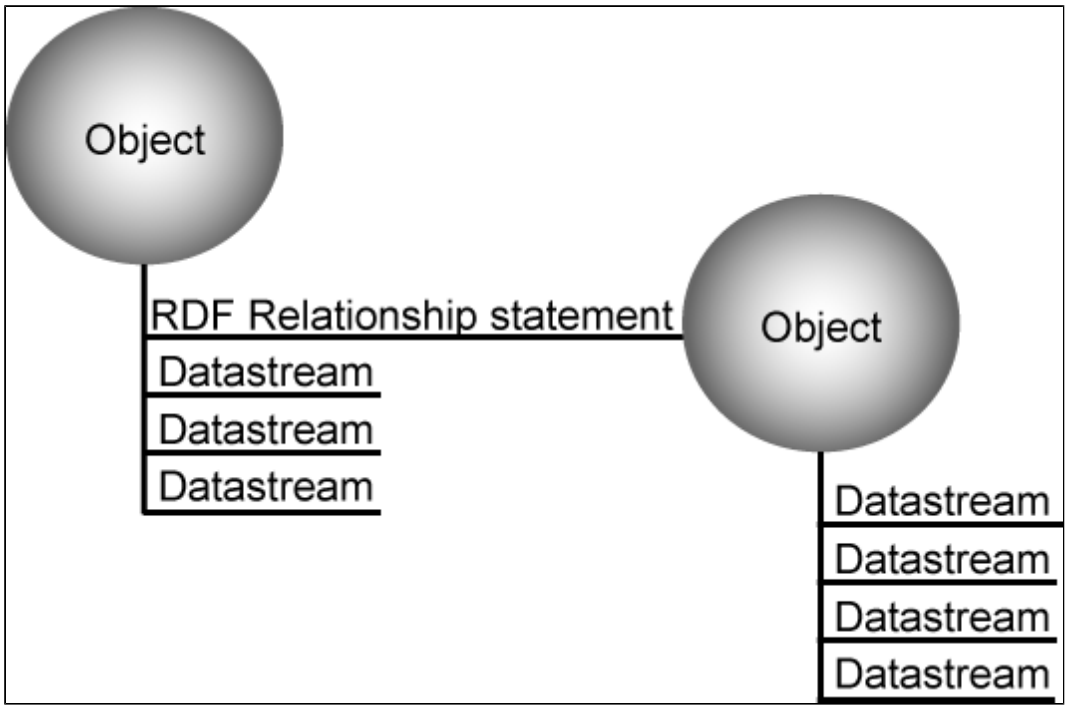


This can be confusing, since specialized “objects” and their specialized “Datastreams” perform a number of different functions, and serve a number of different purposes. Datastreams may store a reference to the binary file being stored (such as PDF, or an image), they may declare the Content Model Object affiliations of an object, and they may store the relationships that an object has to other objects in the repository. Each object in Islandora has a default DC Datastream. This Datastream is a **Dublin Core** record associated with this object. Islandora uses other metadata schemas, and crosswalks relevant data to the DC stream. For more information about how Islandora utilizes crosswalks, please review Ch apter 12 - Metadata in Islandora.

For an overview of the Datastreams you will encounter in Islandora, see APPENDIX C - Datastream Reference.

Objects have Relationships

Objects in Islandora have relationships to one another. These relationships are stored in a RELS-EXT Datastream in an object, which usually has the label “Fedora Object-to-Object Relationship Metadata.” This Datastream is written in RDF.



The RDF statements in this Datastream will indicate, for example, what collection object(s) an object belongs to, what content model object it subscribes to, and what its Persistent Identifier (PID) is. These statements take the following form:

<subjectFedoraObject> <typeOfRelationship> <targetFedoraObject>

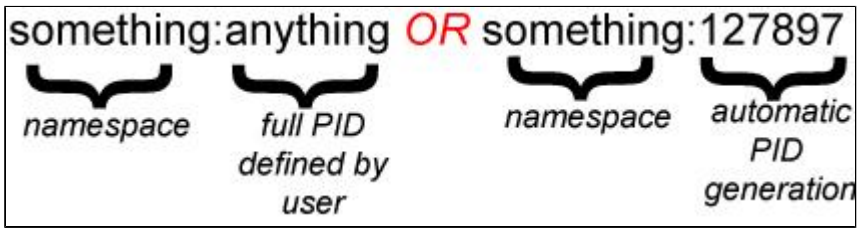
So, for example, the relationship between an object and the collection it belongs to might look like this:

<myMapImage> <isMemberOf> <myMapCollection>

A default set of common relationships is defined in the Fedora Relationship Ontology. More information on digital object relationships in general is available from the Digital Object Relationships section of the FedoraCommons documentation. References to these documents are also provided in the selected reading section of this guide.

Objects have Unique (Persistent) Identifiers

Each object in an Islandora/Fedora repository has a unique (persistent) identifier called a PID (Persistent Identifier). No two PIDs in your repository can be the same. PIDs look like this:



In the first case, the entire PID has been specified by a user. In the second case, the PID is being automatically created based on the PID **namespace prefix** (which is the first half of the PID, or the portion before the colon).

PID namespace prefixes do not need to be meaningful, they just need to be unique. However, we often make these namespaces meaningful in order to help us sort and find items in our repository. For example, a namespace might refer to the type of content (such as audio:) or it may refer to the name of an institution or collection (upei: OR stmatthews:). As you navigate Islandora, you will see PID namespaces mentioned frequently, and you will be prompted to choose PID namespace prefixes or PID namespaces.

Another way of understanding the PID is as the basis for your repository's Uniform Resource Identifiers (URIs) which uniquely identify items in your repository. To learn more about PIDs and PID namespace prefixes, visit Fedora Commons documentation; specifically, the section on Fedora Identifiers.

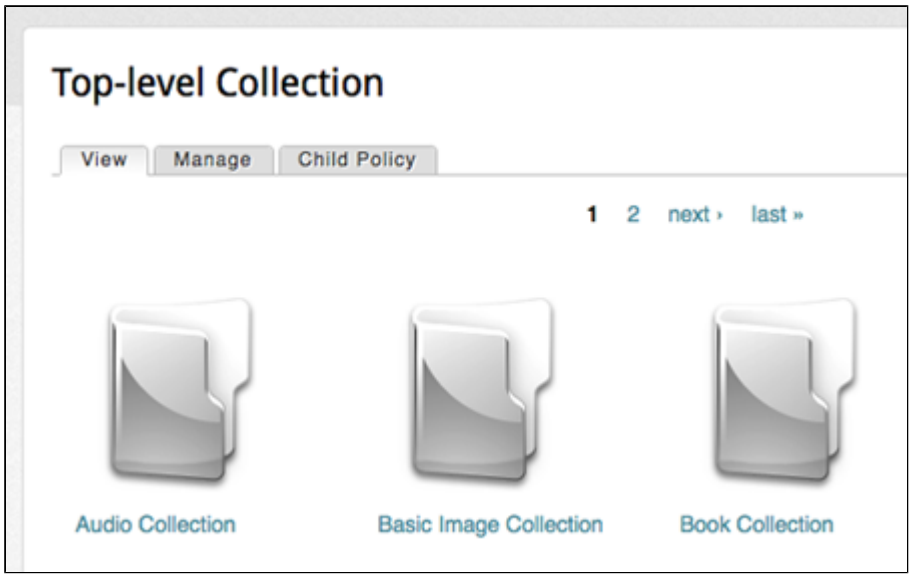
Managing and Building Collections in Islandora

Once you have logged into your Islandora site as an administrator, you can view a list of all current collections by clicking the **Digital Repository** link on the top menu (see screenshot below).

You should see any collections that are installed as part of a **Solution Pack,** and they will appear with a default “icon” folder, or a thumbnail image. These tutorials assume that you have installed one or more Solution Packs. Please review Chapter 5 - Islandora Modules if you are not sure if you have Solution Packs installed. If no solution packs are installed, you will not be able to create collections of content in Islandora.

The following tutorials will guide you through the basics of managing and building your Islandora collections:

- How to Add an Item to the Digital Collection
- How to Edit an Object's Metadata
- How to Purge an Object
- How to Make Changes to the Datastreams of an Object
- How to Create a New Islandora Collection
- How to Batch Ingest Files



Advanced Collection Management

The Collection Manager allows you to create new child collections (detailed above) but it also provides a number of advanced collection management functions. These functions will be explained in the following tutorials:

- How to Manage Collection Policies
- How to Purge Multiple Objects from a Collection