

How to Edit/Create Ingest Forms

Introduction

This instructions has two sections.

"How to Edit an Existing Form" is geared toward people who do not know XSLT. It provides instructions on interacting with forms to do things like copy an existing form, then delete elements from that longer form to make a shorter form, then associate that form with a specific content model. This can support setting up a workflow, and you don't need to know XPATH/XSLT to do it.

"How to Create a New Custom Form" is a tutorial that steps through creating some new form fields and using XPATH/XSLT to control how they interact with XML. This is geared towards someone who will add new fields to an existing form or who will troubleshoot issues related to how forms are interacting with XML. This wiki does not cover XPATH/XSLT, and to use this information, you need to separately learn about XPATH/XSLT because the XML forms are using XPATH/XSLT to interact with XML records on your site.

Where is Forms Management on my Islandora Site?

- 1) Click to "Islandora" in the admin menu across the top of the page.
- 2) A window will pop up in the middle of the screen. Click to "Form Builder". This takes you to (admin/islandora/xmlform).
- 3) This shows you all forms on your site. There are several built in forms that cannot be deleted (you uninstall them by deleting the form association on each). Any custom forms you have installed on your site will show up at the bottom of the list.

TITLE	TYPE	OPERATIONS					
Newspaper Issue	Built-in	Copy		View	Export		Associate
Organization EAC-CPF form	Built-in	Copy		View	Export		Associate
Organization EAC-CPF Read Only	Built-in	Copy		View	Export		Associate
PDF MODS form	Built-in	Copy		View	Export		Associate
Person EAC-CPF form	Built-in	Copy		View	Export		Associate
Person EAC-CPF Read Only	Built-in	Copy		View	Export		Associate
Place EAC-CPF form	Built-in	Copy		View	Export		Associate
Place EAC-CPF Read Only	Built-in	Copy		View	Export		Associate
Scholar MADS form	Built-in	Copy		View	Export		Associate
Thesis MODS form	Built-in	Copy		View	Export		Associate
Thesis MODS recovery form	Built-in	Copy		View	Export		Associate
Video MODS form	Built-in	Copy		View	Export		Associate
Web ARChive MODS form	Built-in	Copy		View	Export		Associate
Full MODS	Custom	Copy	Edit	View	Export	Delete	Associate
MODS Simple Entry	Custom	Copy	Edit	View	Export	Delete	Associate

This is where you administer forms.

The forms show up when you go to upload an item and add metadata, or when you go to edit metadata on an item on your site.

In order for a form to show up, you need to (1) have the form installed on your site, and (2) add a form association and "Associate" that form with the specific Content Model you will use it with.

How to Associate a Form with a Content Model

In order for a form to show up for someone uploading material or editing metadata on a site, the form has to (1) exist on the site and be in that menu at **Islandora > Form Builder** (admin/islandora/xmlform), and (2) be associated with that Content Model.

Most of this wiki page is about getting the form onto the site. A last step will always be adding the form association, and for that step, you should come back here for instructions on adding the association.

View an Existing Form Association

- 1) Click to "Islandora" in the admin menu across the top of the page.
- 2) A window will pop up in the middle of the screen. Click to "Form Builder". This takes you to (admin/islandora/xmlform).
- 3) On the right hand side of the page, click on "Associate" next to an existing form.

This takes you to the existing form association(s) for that content model.

Below is the default form association for the Audio MODS form, which is one of the built-in forms that ships with Islandora:

The screenshot shows the 'Associate Form' interface for the 'Audio MODS form'. At the top, there's a header 'Associate Form "Audio MODS form"'. Below it, a 'Home' link is visible. The main content area is divided into two sections: 'CURRENT ASSOCIATIONS' and 'ADD ASSOCIATION'.

CURRENT ASSOCIATIONS

CONTENT MODEL	TYPE	DATASTREAM ID	LABEL FIELD	TRANSFORM	SELF TRANSFORM	HAS TEMPLATE	OPERATIONS
islandora.sp-audioCModel	Built-in	MODS	['titleinfo']['title']	mods_to_dc.xsl	No Self Transform	No	Enable

ADD ASSOCIATION

Content Model *

The content model to associate with a form. If the content model has no decendents it will not show up in autocomplete.

Metadata Datastream ID *

The datastream ID where the object's metadata is stored.

Title Field

The form field for the object's label.

XSL Transform

An XSL transform for setting the Fedora object's Dublin Core metadata datastream.

Self XSL Transform

An optional transformation applied to form data prior to ingestion.

Add a New Form Association

- 1) Click to "Islandora" in the admin menu across the top of the page.
- 2) A window will pop up in the middle of the screen. Click to "Form Builder". This takes you to (admin/islandora/xmlform).
- 3) First look at the forms and find a form with an existing form association that you feel is working well and is similar to how you want your form to work. Next to this existing form, click on "Associate". This will show you the existing form association. You can use this as a template for setting up your new form association.
- 4) In another tab of the browser, go back to the main forms management screen (admin/islandora/xmlform). Next to the form you want to associate with a Content Model, click on the link to "Associate".
- 5) This pops up a window with the form associations for that form. In the "Add Association" section, click through each field:

"Content Model" = the Content Model you want this form to show up for (*Begin typing the PID of the content model you wish to associate a form with, and the name will auto-complete.*)

"Metadata Datastream ID" = The datastream that the MODS form will create or edit. (ie. MODS might be the datastream for a MODSXML file)

"Title Field" = Drop down menu lets you select any metadata form that that form is able to interact with. You select which one you would like to map to the title of the Drupal representation of that item.

"XSL Transform" = Option to map from MODS XML (if you are installing a MODS form as opposed to some other metadata schema) to Dublin Core. Select this if you are installing a MODS form. Leave it "No Transform" if you are working with any other format.

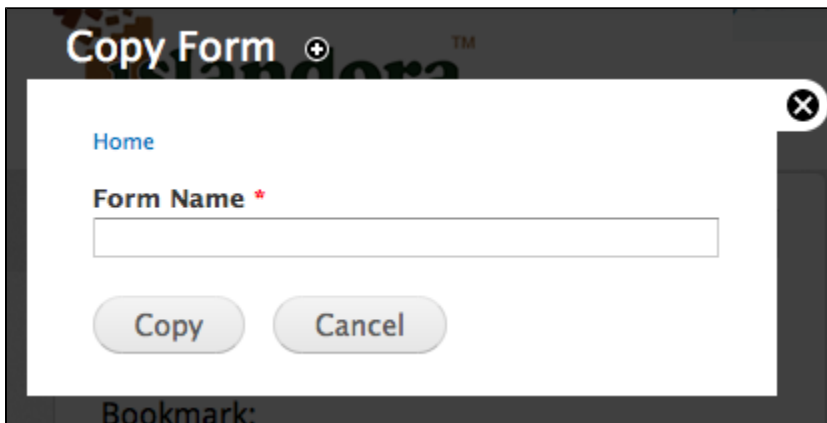
"Self XSL Transform" = Drop down menu shows transformations available to run on the XML on save. You might use this to clean up the XML, for example, to remove empty elements. You might also use this for functionality that cannot be implemented in the Form Builder's web forms, for example, to sequence elements relative to one another. Instructions on how to add a .xsl file to the list of options are here <https://github.com/Islandora/islandora/wiki/Creating-and-Working-With-XML-Forms#adding-an-xslt-transformation-to-the-list-of-available-transformations>. In general, your site or hosting company will have a standard transformation to be used for each metadata schema.

"Upload Template Document" = You can upload a sample metadata record in XML, and that will be used to prepopulate the form fields when someone creates a new item on the site. This will have no effect on editing existing metadata for an item on the site.

How to Copy an Existing Form That is Already on Your Islandora Site

When creating a new form, it is best to start by copying from an existing form, especially one already associated with the content model you plan to use the new form with. In general, it is easier to shorten an existing form by deleting form fields and elements, than it is to add new form fields and elements. Even without knowing XPath/XSLT, you can shorten a form and set up a workflow on your site which involves only seeing the fields you need to fill in. This can save significant time for people uploading content and clicking through the form on each upload.

- 1) Click to "Islandora" in the admin menu across the top of the page.
- 2) A window will pop up in the middle of the screen. Click to "Form Builder". This takes you to (admin/islandora/xmlform).
- 3) In the list of available forms, find the form you want to Copy and click "**Copy**" to the right of the form name. On the next screen you will be prompted to give your new form a name:



Make sure your name is something that you will remember.

Once you have named your form, click "**Copy**."

- 4) You will be redirected to the Form editing interface, and receive a message at the top that your form has been successfully created:



Successfully copied form "Test Audio Form".

Form Editor

Form Properties Save & Preview Save

Elements

Add Copy Paste Delete

- Root (form)
 - titleInfo (markup)
 - title (textfield)
 - subTitle (textfield)
 - name (tabs)
 - 0 (tabpanel)
 - type (select)
 - namePart (textfield)
 - role (textfield)
 - typeOfResource (select)
 - tableOfContents (textarea)
 - genre (select)
 - originInfo (fieldset)
 - dateIssued (textfield)
 - publisher (textfield)
 - country (textfield)
 - place (textfield)
 - language (textfield)
 - abstract (textarea)
 - identifier (textfield)
 - physicalDescription (fieldset)
 - form (select)
 - extent (textfield)
 - note (textarea)
 - subject (fieldset)
 - topic (tags)
 - 0 (tag)

Preview

Dashboard Content Structure Appearance People Modules Configuration Islandora

Reports Help

Add content Find content Edit shortcuts

Home » Islandora Admin » Form Builder

Preview Form

Title

Title of the work.

Sub Title

Name

1

Type

personal

5) Add the form association for the content model. Adding the association means that the new form will show up in the drop down menu when you go to edit or add metadata for that kind of Content Model.

First look at the default form association for the form you just copied. Next, go to the main forms management area, and click to "Associate" to the right of the name of the new form you just made by copying. Now, the next form will show up in a drop down menu of available forms when someone goes to add or edit metadata for that kind of content model.

How to Delete a Form from Your Islandora Site

- 1) Click to "Islandora" in the admin menu across the top of the page.
- 2) A window will pop up in the middle of the screen. Click to "Form Builder". This takes you to (admin/islandora/xmlform).
- 3) Click "Delete" to the right of the form you want to delete.

In general, it is a good idea to keep a local copy of the form you deleted. For example, if a bug in the form resulted in metadata errors, then you may want to be able to clean up metadata on your site. It also may be a good idea to delete form associations rather than delete the form. By deleting associations, you ensure that the form will not show up to people uploading and editing metadata, but you can easily bring the form back by adding the association later.

How to Install a Form from Github

Where to Find Other Peoples' Islandora Forms

Along the top of the screen at **Islandora > Form Builder**, you will also see an option to "Create Form" or "Import Form".

Creating forms from scratch requires proficiency in XPath/XSLT.

Importing forms used by other sites can be a good way to look for source material beyond the built in forms that come with an out-of-the-box Islandora install.

Islandora's Built In Forms

Some forms are already built into your Islandora site. To see those, click to "Islandora" in the admin menu across the top of the page. A window will pop up in the middle of the screen. Click to "Form Builder". This takes you to (admin/islandora/xmlform).

Here you will see all of the forms currently installed on your site. From this point you can take existing forms and copy, edit, view, export, delete, or associate existing forms on your site.

Copying an existing form, and editing the copy can be a good way to make small changes and develop a new form.

GitHub sources for Islandora Forms

Here are some sources for forms:

- Islandora Foundation's forms repository: A community repository of Islandora Ingest forms is maintained in the [Islandora Foundation's Github account](#).
- DiscoveryGarden's forms repository: https://github.com/discoverygarden/xml_forms
- Florida Virtual Campus' forms repository: https://github.com/FLVC/fl_islandora_forms

How to Install a Form from XML (ie. from Github)

- 1) Click to "Islandora" in the admin menu across the top of the page.
- 2) A window will pop up in the middle of the screen. Click to "Form Builder". This takes you to (admin/islandora/xmlform).
- 3) Locate a form on GitHub that you want to explore. Download the XML file from GitHub.
- 4) In Islandora, go to **Islandora > Form Builder** (admin/islandora/xmlform), and click "Import Form" along the top of the screen. Click through to install the form on your site. (**Please note:** Depending on the type of form and form provider you choose, there may be additional steps to configure that are out of the scope of this tutorial.)
- 5) Now, set the form association for the form you just installed. Think what content model you would like that form to appear for. On your **Islandora > Form Builder** (admin/islandora/xmlform), look next to the default form for the content model you want to associate the new form with, and click "Associate" on the default form. This will bring up a screen showing you the settings for the form association. On your new form that you want to association with that content model, go to **Islandora > Form Builder** (admin/islandora/xmlform), and click "Associate" next to the form you are working with. Set up the new form association based on the form association you pulled from the default form for that content model.

Installing a form from GitHub, lets you look at forms used by other institutions or organizations. Keep in mind, as mentioned above in the **Please note:** of step 4, that forms used by other organizations may have specific requirements or local metadata profiles that don't matter to you or cannot be covered here in this tutorial. Review any given documentation or READMEs from that provider's repository when possible. Also, keep in mind, Islandora is displaying content models on the web interface based on the MODS datastream, and the stylesheets installed on the form association are expecting MODS as input. Not all, but definitely some forms available for download from GitHub work with other metadata formats.

The main reason for examining forms used by the wider community is to find examples of how specific fields are handled. For example, if something like ORCID is not handled in a default form, you may be able to get an example implementation from another institution using Islandora.

How to Edit an Existing Form

The following section will show you where to access forms in your Islandora installation, and how to Copy and Edit a form, and associate it with a content model. This section gives useful information for people who do not have knowledge of XPath/XSLT, and covers only minimal configuration options - how to change names of forms, how to delete fields, and how to determine what content models a form appears as an option for. The goal is that anyone can follow along with this first section, regardless of coding knowledge.

About the Form Builder Interface

1. Under "**Form Properties**" you can view the properties of your form.
2. Under "**Elements**" you can view the fields you have created for your form, as well as make changes to those fields. Clicking on an element name will allow you to view the way that element will be created, read, updated, and deleted when you interact with the form, and the resulting metadata that the form will write.
3. Click "**Save**" to save changes, and "**Save and Preview**" to save the changes and render the form in the preview screen.

The screenshot shows the Form Editor interface. On the left, the 'Elements' tree is expanded, showing a hierarchy starting from 'Root (form)' down to various fields like 'titleInfo (markup)', 'name (tabs)', and 'originInfo (fieldset)'. The 'Preview' pane on the right shows a form with a header bar containing navigation links and a user profile. Below the header, there's a 'Preview Form' section with two text input fields labeled 'Title' and 'Sub Title'.

To create a form using a new schema, you will have to create a new form from scratch, or Copy a form of the appropriate schema. This requires knowledge of both XPath and the .xsd affiliated with that schema. To create a new element, you will also have to be familiar with XPath.

However, there are several simple “tweaking” functions you can perform that do not require in-depth knowledge of XPath and your schema’s .xsd. For example, you can delete an element (one that is deemed unnecessary for a user to fill out on ingest) by selecting the element with your cursor and clicking “Delete.”

Also, you can change the label of a field, or populate the field with a default value (for example, to pre-populate an author field for a author-specific collection) by selecting the element and editing the element’s configuration properties.

The screenshot shows the 'Edit Form' interface. On the left, the 'Elements' tree is expanded, showing a list of elements including 'dc:title (textfield)', 'dc:creator (textfield)', 'dc:contributor (textfield)', 'dc:description (textarea)', 'dc:coverage (tags)', 'dc:subject (textfield)', 'dc:type (select)', 'dc:format (textfield)', 'dc:identifier (textfield)', 'dc:language (textfield)', 'dc:publisher (textfield)', and 'dc:relation (textfield)'. The 'Element Form' pane on the right shows the configuration for the 'dc:creator' element. It includes fields for 'Identifier' (set to 'dc:creator'), 'Type' (set to 'textfield'), 'Title' (set to 'Creator'), 'Description' (set to 'An entity primarily responsible for making the resource.'), 'Default Value' (empty), and a 'Required' checkbox.

Here, you can change the label users see when interacting with this form field by changing the value for the “Title” of the element. You can change the description that appears underneath the entry box by changing the text in the “Description” field. If you want the field to be pre-populated with the creator’s name, enter the name in the “Default Value” field. In addition, if you want to make this field mandatory for your form, select the “Required” checkbox, and this field will appear with an asterisk, and the users will not be able to submit this form without filling in this field. Once you have made these changes, click “Save and Preview.” You will be able to view the form as it will appear to its users.

Preview Form

Title:

A name given to the resource.

Author:

Moses, Donald


Please provide the author's name in last name, first name format.

Contributor:

An entity responsible for making contributions to the resource.

Here the default "Creator" label has been changed to "Author" and customized instructions for users appear below this field. The Author field has been pre-populated with a name to prevent unnecessary repetition for users ingesting items into this collection. However, the user can still edit this field on ingest if there is a case where the author's name is different.

Once you have tweaked this form, you will have to associate it with the appropriate content model in order for this form to appear as an option for users on ingest. To create an association between your new form and a content model, navigate back to **Islandora > Form Builder** (admin/islandora/xmlform) and select "Associate" on the form you wish to associate with a content model. You will see the following page:

Associate Form "Full MODS – Campus Voice" 

[Home](#)

CURRENT ASSOCIATIONS

CONTENT MODEL	TYPE	DATASTREAM ID	LABEL FIELD	TRANSFORM	SELF TRANSFORM	HAS TEMPLATE	OPERATIONS
islandora:newspaperIssueCModel	Custom	MODS	['originTab']['originPanel']['originInfo']['dateIssuedSet']['issueDate']	mods_to_dc.xsl	FLVC_MODS_postprocessing.xsl	Yes	Delete

ADD ASSOCIATION

Content Model *

The content model to associate with a form. If the content model has no descendants it will not show up in autocomplete.

Metadata Datastream ID *

The datastream ID where the object's metadata is stored.

Title Field

The form field for the object's label.

XSL Transform

An XSL transform for setting the Fedora object's Dublin Core metadata datastream.

Self XSL Transform

An optional transformation applied to form data prior to ingestion.

Upload Template Document

No file chosen

A sample metadata file used to prepopulate the form on ingest.

In order to fill out this form appropriately, ensure that you are familiar with the details of your solution pack - particularly the name of the content model, and the name of the metadata stream it prescribes. If you are not sure, please review the Solution Pack documentation for your installed solution packs. Alternatively, you can look at the list of forms on your site, find the default form for your desired content model, and click "Associate" next to it. This will pull up the default form association for that content model. You can recreate that on your new form.

At the top of the screen, you can see a list of all existing form associations.

Content Model:

Under "Content Model" begin typing the PID of the content model you wish to associate a form with, and the name will auto-complete:

Metadata Datastream ID:

Under “**Metadata Datastream ID**” put in the DSID of the Datastream where your content model stores its rich metadata (not the DC Datastream). You must use the existing DSID prescribed in your content model (e.g., MODS, MARCXML, etc.), or this metadata will not appear in your collection objects. If you do not know what the correct DSID is for rich metadata in your content model, please review the documentation for the relevant content model.

Title Field:

After you have entered the DSID for the Datastream, click "Do not set the label" in the “**Title Field**.”

When you do this, you will notice that there is a change in the “**Title Field**” drop-down menu. Islandora populates this drop-down field with the elements in your form, so that you can choose which element will serve as the source for the object's label. Most of the time, you will likely want the object's label to come from the “Title” element. For something like a Newspaper Issue, you might want the title of the issue to come from the “date” element instead. This label can be manually edited after ingest if ever necessary.

Select which field you would like to use for the object's label.

If you later make changes to the element that you have selected as the Title Field, be sure to go back and update this form association again, or the form will no longer function.

The next three options require a little bit more explanation:

XSL Transform:

The “XSL Transform” file you choose is used by Islandora to crosswalk the schema of your rich metadata XML to Islandora's default DC stream. In this case, where you are copying a form, you will want to select the default xsl for your content model. This information is provided in the Solution Pack section of your document. (For more information about Islandora's use of .xslts, please review the “Islandora and Metadata” section of the guide. XSLT are called from a folder within islandora_content_model_forms (in the XML Forms modules).

Self XSL Transform:

This is a transformation that runs on the XML you've entered in your form and cleans it before you save the form. You can use this to remove empty fields or enforce validation of a local profile.

Upload Template Document:

The “**Template Document**” is a feature that allows you to submit a form with fields already completed. Usually, this will be a form that you have created by hand in an XML editor. It is not obligatory.

Saving the new form association:

Click “**Add Association**.” Now, whenever you create an object in a collection that is associated with the content model you have used here, you will see your new form as an option to select in the drop down menu.

How to Create a New Custom Form

This section will discuss how to create a new form using the Islandora Form Builder.

This section presumes that you are using the Virtual Machine Image or are visiting <http://sandbox.islandora.ca> OR that you have installed and configured the XML Forms module, and that you have a basic knowledge of XPath and understand metadata schemas. For an overview of how Islandora handles descriptive metadata, read [Chapter 11 - Metadata in Islandora](#).

A useful tutorial on how to create a new form is here: Metadata and XML Forms, Donal Moses (UPEI), presented at Islandora Camp NY, November 20, 2013, available at <http://islandora.ca/sites/default/files/Metadata%20XMLForms%20Islandora%20Camp%20NY%202013.pdf> . This tutorial steps through the process of building a simple Dublin Core form, and includes screenshots of each step.

Introduction

The XML Form Builder allows you to create and manipulate **XML form templates** and associate them with **content models**. This allows you to create custom forms to address the needs of your particular collection, or to pre-populate repeating fields. For example, if a collection of PDFs was all written by the same author, you may wish to create a custom form for this collection that has the author's name pre-populated, so that users ingesting into this collection will not need to re-enter this information. Using custom forms you can also specify which metadata elements you wish to use to describe your object, and create validation rules for particular fields, among other features.

- If you are a developer, or somebody looking to install the XML Form Builder, you will want to review the section on the [XML Form Builder](#) module, which discusses the module's installation and configuration.
- If you are a user, then the following documentation assumes that you have some understanding of metadata schemas and XML, as well as Islandora specific concepts such as Content Models, and Collection Objects. The greater grounding you have in **XPath**, **XML Form Templates**, and **XML Schemas** (.xsd), the greater use you will be able to make of the form builder.
- Solution Packs are designed to come pre-packaged with suitable forms written in MODS. These forms can be copied, and edited or modified to suit your needs.
- Forms can be created based on any schema.

Before You Begin

Review and install the [XML Form Builder](#) module and its dependencies.

Metadata Schemas

Metadata schemas specify the names and properties of elements that can be used in an XML file that is based on that schema.

When developing an XML form in the XML Form Builder you'll need to reference the schema of the metadata format you are working with. Throughout this document we'll be using the OAI DC metadata schema as an example. The OAI DC XML format is the serialization of Simple Dublin Core metadata descriptions and we'll be using Dublin Core elements in this example. You can view/retrieve the schema from http://www.openarchives.org/OAI/2.0/oai_dc.xsd

Sample XML Record

It is helpful to have an example record on hand while developing the form. Ensure that it matches the version of the XML schema you are working with.

Sample MODS Version 3.5 Documents are available for download from the Library of Congress: <http://www.loc.gov/standards/mods/mods-guidance.html>

Sample OAI DC Record from a Fedora Repository

```

<oai_dc:dc xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/" xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/
[http://www.openarchives.org/OAI/2.0/oai_dc.xsd]">

<dc:title>Pioneer days & shanty ways</dc:title>
<dc:creator>Eldershaw, Edith V.</dc:creator>
<dc:subject>History</dc:subject>
<dc:subject>Social life and customs</dc:subject>
<dc:description>Edith V. Eldershaw.</dc:description>
<dc:description>Printed by Williams & Crue Ltd.; Summerside, P.E.I.</dc:description>
<dc:description>Contains "stories", poems, and photographs,</dc:description>
<dc:publisher>Eldershaw</dc:publisher>
\\

<dc:type>collection</dc:type>
<dc:type>ingested</dc:type>
<dc:format>electronic</dc:format>
<dc:identifier>ilives:257167</dc:identifier>
<dc:language>eng</dc:language>
<dc:coverage>Prince County (P.E.I.)</dc:coverage>
\\

<dc:coverage>Tignish (P.E.I.)</dc:coverage>
<dc:coverage>Prince Edward Island</dc:coverage>

</oai_dc:dc>

```

XML Editors (on your desktop; outside of Islandora)

When developing a metadata form it is important to have an XML editor to test you created and edited XML in. Typically these editors can help you determine the XPath of an element, whether the output you are producing is valid, etc. XML editors would include [oXygen](#) (commerical), [XPontus](#) (opensource), and there are many others.

Anytime you are working with software that writes XML, it's important to validate that XML against the schema, and also to ensure backward compatability when importing metadata that follows older versions of the schema.

XML Form Builder

XML Form Builder is a Drupal module that integrates the creation of XML based forms into Islandora. Once a form has been built, it is associated with a content model. This tutorial will take you through the process of creating a form, associating it with a content model, and implementing it with a collection. Once it has been created you will be able to create and edit your metadata.

To use the XML Form Builder navigate to the module in your Islandora site: **Islandora > Form Builder** (admin/islandora/xmlform)

Form Builder Interface

When you start the module you are presented with a list of forms (those are the forms that come bundled with your Islandora install) and a series of options to perform.

Form Builder

[Home](#) » [Administration](#) » [Islandora](#)

[+ Create Form](#) [+ Import Form](#)

TITLE	TYPE	OPERATIONS					
Audio MODS form	Built-in	Copy	View	Export	Associate		
Basic image MODS form	Built-in	Copy	View	Export	Associate		
Islandora Book MODS Form	Built-in	Copy	View	Export	Associate		
Large image MODS form	Built-in	Copy	View	Export	Associate		
Newspaper MODS form	Built-in	Copy	View	Export	Associate		
PDF MODS form	Built-in	Copy	View	Export	Associate		
Video MODS form	Built-in	Copy	View	Export	Associate		
Test Image Form	Custom	Copy	Edit	View	Export	Delete	Associate
test MODS	Custom	Copy	Edit	View	Export	Delete	Associate
TestForm	Custom	Copy	Edit	View	Export	Delete	Associate

Create Form: Select to begin the process of creating a metadata form from scratch or from an existing form definition file (an XML Form Builder form).

Copy: Copies an existing form, that you can then modify. This is probably be one of the most common methods you will use to create new forms.

Edit: Edits an existing form.

View: View an existing form. This option is useful when testing input. You can submit a form and see its XML output.

Export: Exports an existing form and allows you to save the form XML to your local computer.

Delete: Removes a custom form.

Associate: Links a form with a content model.

Creating a Form

The examples and screenshots below refer to an Islandora 6.x installation. The basic functions of the form builder are the same, although elements may look slightly different.

To start creating a form click "**Create Form.**"

In the Create Form dialogue enter a form name - for example we are creating a basic OAI DC form so a name like oai_dc_basic would be appropriate. If you have an existing XML Form Builder form you could upload the form definition. We'll be creating this OAI DC XML form from scratch, so we can click on "**Create**" without uploading a form definition.

Home > Administer > Content management > XML Form Builder > Create Form

Create Form

Form Name: *

Form Definition:

An optional XML form definition template.

The module should report that it successfully created a new form called oai_dc_basic.

Home > Administer > Content management > XML Form Builder > Edit Form

Edit Form

Successfully created form: oai_dc_basic.

Form Editor

Form Properties

Save & Preview Save

Elements

Add Copy Paste Delete

Form (form)

Preview

UPEI UNIVERSITY of Prince Edward ISLAND

Islandora Virtual Environment

Digital Repository Create content Administer

Home > Administer > Content management > XML Form Builder > Preview Form

Preview Form

This is the main form building/editing interface for creating XML forms and it provides methods for adding form properties, form fields and a preview pane.

Setting Form Properties

Form Editor

Form Properties

Elements

Add Copy Paste Delete

Form (form)

Properties Form

Root Element

Root Element Name: changeme

Namespace URI:

Schema

Name:

Namespaces

Add Delete

Prefix	URI

When creating an XML form the first thing you need to set in the Form Editor is the Form Properties. This is where having an example of an OAI DC record would come in handy and would provide the information you need to fill in the Form Properties. Here is the part of the record that you'll use:

```
<oai_dc:dc xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/" xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/
http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
```

It provides information about the Root Element Name, the Namespace URI, the location of the Schema, as well as the various namespaces needed.

Properties Form

Root Element

Root Element Name: oai_dc:dc

Namespace URI: http://www.openarchives.org/OAI/2.0/oai_dc/

Schema

Name: http://www.openarchives.org/OAI/2.0/oai_dc.xsd

Namespaces

Add Delete

Prefix	URI
oai_dc	http://www.openarchives.org/OAI/2.0/oai_dc/
xsi	http://www.w3.org/2001/XMLSchema-instance
dc	http://purl.org/dc/elements/1.1/

Click "**Save**" once you have the properties entered. Once the form properties have been saved you are ready to add fields to your form.

Adding form fields

Some schema specify that elements appear in a certain order (an order-based schema), that certain elements are required, that only certain elements are repeatable, or whether and how elements can be nested. Refer to your schema documentation and to any guidance it offers in the creation of records based on the schema. In the schema we are using in this tutorial, Simple Dublin Core, each of the fifteen elements is optional and may be repeated, but no nesting is allowed. We will create a simple form for this exercise that utilizes several Dublin Core elements. Let's create a table listing the element names, the form field labels, the types of form fields, the content of the elements, and whether they are repeatable. We'll use this information when adding fields to our form.

Element	Label	Type	Content	Repeatable
title	Title	textfield	The title of the work.	no
creator	Creator(s)	tags/tag	The creator(s) of the work.	yes
description	Description	textarea	A description of the work.	no
type	Type	select	A controlled list of terms	no
date	Date	datepicker	The date the work was issued or published.	no
subject	Subject(s)	tags/tag	The topic of the work.	yes
rights	Rights	textarea	Information about rights held in and over the resource.	no

Adding a textfield type form field - the dc:title element

We can use the information in our table to fill out the "Element Form" pane. We can start with the **title** element.

In this part of the form you can enter values for **Identifier**, **Type**, **Title**, **Description**, **Default Value**, and **Required**. These can be defined as follows:

Identifier: An ID for this form field. It is the Drupal form array key for this element. Typically you would just enter the name of the element you are describing.

Type: The form field type for this field (textfield, textarea, select, etc.)

Title: The label of the form field as it appears on your form. Typically you would just enter the name of the element you are describing.

Description: The description of the element that this Element Form is about.

Default Value: The value to be display or selected initially for this element if the form has not been submitted yet.

Required: Indicates whether or not the element is required. This automatically validates for empty fields, and flags inputs as required. Fields with a **Type** of "file" are not allowed to be required.

The rest of the form deals with where each element is created, read, updated, and deleted in the XML tree. This is where it would be useful to understand how XML works and to have a basic understanding of XPath.

Reviewing our OAI DC XML sample record we can determine the location/context of the title element.

```
<oai_dc:dc xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/
[http://www.openarchives.org/OAI/2.0/oai_dc.xsd]">
<dc:title>Pioneer days & shanty ways</dc:title>
```

The full XPath in this XML document for the dc:title element would be:

```
/oai_dc:dc/dc:title
```

where /oai_dc:dc is the parent element and dc:title is the child element. The **Create** dialogue is where we enter the information needed for creating an element in our form. We'll be entering information into the **Path Context**, **Path**, **Type**, and **Value** properties in our example. If you had an order-based schema, you would also fill in the **Schema** field. Here are some definitions for each of those properties:

Path Context: the context at which the XPath action will be executed.

- *document* - the XPath query is run from the root element of the document
- *parent* - the XPath query is run from the node created/read by its parent's form field
- *self* - only applies to Delete and Update actions and applies to the node selected by the Read action.

Path: An XPath to this element's parent object. This is used to determine where this element is inserted.

Note: If you specify an XPath that has multiple elements in it then it will generate multiple elements in the form. If they aren't nested or wrapped in an element then elements will appear below the Submit button. To prevent this from happening, you'll need to update the XPaths or the form itself.

Schema: An XPath to the definition of this element's parent. The XPath is executed in the schema defined in this form's properties. This is used to determine the insert order for this element.

Type: The type of node that will be created. If XML is specified, an XML snippet is expected in the **Value** field.

Value: If the **Type** is either "element" or "attribute," the name of the element or attribute is expected here. If the **Type** is "XML," an XML snippet is expected in which %value% indicates where the value of the form field will be inserted.

Review the image below and think about how your other OAI DC elements will be created based on this pattern.

- **Create** action is about selecting the parent node where the new node will be created.

The image shows a 'Create' dialog box with the following fields and annotations:

- Create** (checked checkbox): An annotation points to this checkbox with the text: "Checking the Create box opens up the Create dialog."
- Path Context:** A dropdown menu showing 'document'. An annotation points to it: "The path context is either **parent** or **document**. For the OAI DC form we will be using **document** as the path context."
- Path:** A text field containing '/oai_dc:dc'. An annotation points to it: "This is the path where the element will be create ... the parent of the element you are creating."
- Schema:** An empty text field.
- Type:** A dropdown menu showing 'element'. An annotation points to it: "This can be **element**, **attribute**, or **XML**. In this case we are creating an element. Alternately we could also create the element using XML, but a type of element makes sense here."
- Value:** A text field containing 'dc:title'. An annotation points to it: "Since we are creating an element, we enter the value of the element – dc:title. If we were creating the value using XML we could enter"

Below the 'Value' field, an example XML snippet is provided:

```
<dc:title>%value%</dc:title>
```

The rest of the form deals with where the element will be read, updated, and/or deleted from.

- **Read** action is about selecting the node that will be used to populate the form field.
- **Update** action is about updating the node that was used to populated the form field.
- **Delete** action is about deleting the node that the Read action selected.

Note: You can Update or Delete nodes other than the node which is Read - For example with `mods:name` ... where sub-elements are created with a form field and additional nodes are automatically created with XML code. We may want to delete/update the entire `mods:name` and its sub-elements.

The screenshot shows the XML Form Builder interface with three sections: Read, Update, and Delete. Each section has a 'Path Context' dropdown and a 'Path' text field. Annotations explain the choices:

- Read:** Path Context is set to 'document'. Annotation: "The path context is either **parent** or **document**. For the OAI DC form we will be using **document** as the Read path context." Path is set to `/oai_dc:dc/dc:title`. Annotation: "This is the full path where the element will be read from."
- Update:** Path Context is set to 'self'. Annotation: "For Update the path context options are **parent**, **document**, or **self**. In this case the path context for updating the element will be **self**." Path is set to `self::node()`. Annotation: "`self::node()` is the nodeset containing only the context node."
- Delete:** Path Context is set to 'self'. Annotation: "We will use the same values for Delete as we did for Update." Path is set to `self::node()`.

Adding the creator element - an element that may have multiple values

multiple selects

Just a note that the current version of the XML Form Builder does not currently allow multiple selects. Based on the information in this [pull request](#), users "can still use unallowed elements such as 'checkbox', 'checkboxes', 'date', 'file', 'managed_file', 'password_confirm', 'radio', 'radios', 'tableselect', 'vertical_tabs', 'weight', 'button', 'image_button', 'submit'. But the builder will no longer allow you to give XML CRUD actions to these elements, also the 'select' element doesn't allow XML CRUD combined with multiple."

Users still have the option to [create their own forms](#).

In some cases you will have multiple occurrences of an element, for example a digital object may have more than one creator (multiple authors) or may have many subjects that describe it. The Form Builder has several methods dealing with this use case. In our example OAI DC form we have decided that there could be multiple creators and the schema allows that. It is a two step process:

1. create an element with a type of **tags**,
2. nest another element in the tags type element that has a type of **tag**.

This image displays the first step of the creation of the creator element.

The screenshot shows the XML Form Builder interface with the 'Form Properties' panel on the left and the 'Element Form' panel on the right. The 'Elements' list on the left shows a hierarchy: Root (form) > title (textfield) > creator (tags) > 0 (tag) > Submit (submit). The 'Element Form' panel shows the configuration for the 'creator' element:

- Identifier:** Set to 'creator'. Annotation: "The name of the element."
- Type:** Set to 'tags'. Annotation: "Select a type of **tags**. This element will hold a sub-element with a type of **tag**."
- Title:** Set to 'Creator(s)'. Annotation: "The label for the element that will appear on the form."
- Description:** Set to 'The creator(s) of the work.'. Annotation: "The description of the element that appears below the entry box for the element on the form."
- Default Value:** Empty text field.
- Required:** Unchecked checkbox.
- Actions:** Create, Read, Update, and Delete checkboxes. A red bracket groups these actions. Annotation: "None of these are required as we will set them in the **tag** 'child' element."

The second step is to create a nested **tag** type element. The image below displays the values for the properties used for this element.

The screenshot shows the 'Element Form' interface with the following fields and annotations:

- Identifier:** 0. Annotation: "For these type of elements, the practice is to use 0 as the identifier."
- Type:** tag. Annotation: "Select the **tag** type of element."
- Title:** (empty field)
- Description:** (empty text area)
- Default Value:** (empty field)
- Required:** ☐
- Create:** ☒
 - Path Context:** document. Annotation: "The same pattern is used for Create as was used in the dc:title example."
 - Path:** /oai_dc:dc
 - Schema:** (empty field)
 - Type:** element
 - Value:** dc:creator
- Read:** ☒
 - Path Context:** document. Annotation: "Read, Update, and Delete are set up in the same manner as the previous dc:title example."
 - Path:** /oai_dc:dc/dc:creator

Adding a textarea type form field - the **dc:description** element

The "textarea type" of form field should be used for elements that require a description or that contain a large amount of text.

Element Form

Identifier: Enter an identifier.

Common Form Controls | **Advanced Form Controls** | **M**

Type: Select the **textarea** type of element.

Title:

Description:

Default Value:

Required: ☐

☒ **Create**

Path Context:

Path:

Schema:

Type:

Value:

☒ **Read**

Path Context:

Path:

The same pattern is used for Create as was used in the dc:title example.

Read, Update, and Delete are set up in the same manner as the previous dc:title example.

Adding a select type form field - the **dc:type** element

In many cases you will want to provide the user with a list of controlled terms and the select form field type is used. For the type element our controlled list of terms is based on the DCMI Type Vocabulary. The Vocabulary *provides a general, cross-domain list of approved terms that may be used as values for the Resource Type element to identify the genre of a resource.*

DCMI Type Vocabulary

Term	Description
Collection	An aggregation of resources.
Dataset	Data encoded in a defined structure. Examples include lists, tables, and databases. A dataset may be useful for direct machine processing.
Event	A non-persistent, time-based occurrence. Examples include an exhibition, webcast, conference, workshop, open day, performance, battle, trial, wedding, tea party, conflagration.
Image	A visual representation other than text. Examples include images and photographs of physical objects, paintings, prints, drawings, other images and graphics, animations and moving pictures, film, diagrams, maps, musical notation.
Interactive Resource	A resource requiring interaction from the user to be understood, executed, or experienced. Examples include forms on Web pages, applets, multimedia learning objects, chat services, or virtual reality environments.
MovingImage	A series of visual representations imparting an impression of motion when shown in succession. Examples include animations, movies, television programs, videos, zoetropes, or visual output from a simulation.
PhysicalObject	An inanimate, three-dimensional object or substance. Note that digital representations of, or surrogates for, these objects should use Image, Text or one of the other types.
Service	A system that provides one or more functions. Examples include a photocopying service, a banking service, an authentication service, interlibrary loans, a Z39.50 or Web server.

Software	A computer program in source or compiled form.
Sound	A resource primarily intended to be heard. Examples include a music playback file format, an audio compact disc, and recorded speech or sounds.
StillImage	A static visual representation. Examples include paintings, drawings, graphic designs, plans and maps. Recommended best practice is to assign the type Text to images of textual materials.
Text	A resource consisting primarily of words for reading. Examples include books, letters, dissertations, poems, newspapers, articles, archives of mailing lists. Note that facsimiles or images of texts are still of the genre Text.

Source

When adding a select form field in the Form Builder there are two steps:

1. Add the information about the element you are creating
2. Add the terms that will display in the select list for users

1. When adding information about the element we will use the same method that we have used previously.

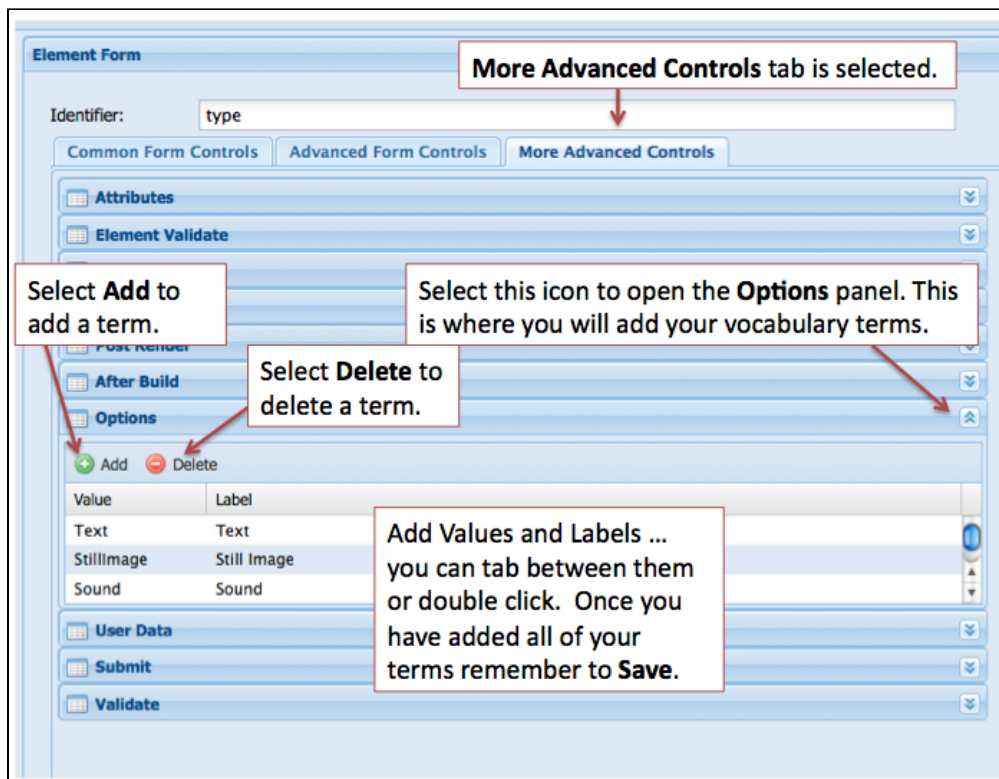
The screenshot shows the 'Element Form' interface with the following fields and annotations:

- Identifier:** A text field containing 'type'. An annotation box says 'Enter an identifier.' with an arrow pointing to the field.
- Common Form Controls / Advanced Form Controls / More Advanced Controls:** Three tabs at the top. An annotation box says 'Select the **select** type of form element.' with an arrow pointing to the 'select' option in the 'Type' dropdown.
- Type:** A dropdown menu showing 'select'.
- Title:** A text field containing 'Type'.
- Description:** A text area containing 'Select the type of element.'
- Default Value:** A text field containing 'Text'.
- Required:** A checkbox that is unchecked.
- Create:** A checkbox that is checked.
- Path Context:** A dropdown menu showing 'document'.
- Path:** A text field containing '/oai_dc:dc'.
- Schema:** A text field that is empty.
- Type:** A dropdown menu showing 'element'.
- Value:** A text field containing 'dc:type'.

Additional annotations:

- A box on the right side of the form says 'The same pattern is used for Create as was used in the dc:title example.' with an arrow pointing to the 'Create' checkbox.
- A large box at the bottom says 'Once all the necessary properties have been filled out, **Save** your changes and then select the **More Advanced Controls** to add the terms that will appear in your select dropdown menu.' with an arrow pointing to the 'More Advanced Controls' tab.

2. To add terms to your select form field, click on the "More Advanced Controls" tab in the "Element Form" pane. Review the image below and enter your terms in the Options panel.



Adding a datepicker type form field - the `dc:date` element

For this example we are using the datepicker type of form field. You will want to review your existing metadata as another type of form field may be more appropriate.

Element Form

Identifier: Enter an identifier.

Common Form Controls | **Advanced Form Controls** | **More Advanced Controls**

Type: Select the **datepicker** type of element.

Title:

Description:

Default Value:

Required: ☐

☒ **Create**

Path Context:

Path: The same pattern is used for Create as was used in the dc:title example.

Schema:

Type:

Value:

☒ **Read**

Path Context:

Path: Read, Update, and Delete are set up in the same manner as the previous dc:title example.

The remaining two elements for our sample OAI DC XML form, [dc:subject](#) (tags, tag) and [dc:rights](#) (textarea), can be built in the same manner as fields of a similar type that we have already created.

Adding a file upload type form field - pdf file uploader

One additional element that can be added for convenience is a file upload type form field. This will allow you to browse for a PDF document on your local machine as part of the Islandora ingest process. The image below illustrates what properties need to be filled in.

Note: the Identifier for this form field type **must** be **ingest-file-location**.

Element Form

Identifier: When creating a file upload form field, the identifier must be ingest-file-location.

Common Form Controls **Advanced Form Controls** **More Advanced Controls**

Type: Select the file type of element.

Title:

Description:

Default Value:

Required: ☐

☐ Create

☐ Read

☐ Update

☐ Delete

This isn't a element that will appear in our XML, so we don't need any of these actions.

Edit Form

Form Editor

Form Properties Save & Preview Save

Elements

Add Copy Paste Delete

- Root (form)
 - ingest-file-location (file)
 - title (textfield)
 - creator (tags)
 - 0 (tag)
 - description (textarea)
 - type (select)
 - date (datepicker)
 - subject (tags)
 - 0 (tag)
 - rights (textarea)
 - Submit (submit)

We've created all of the form fields that were included in this example.

Here's the preview version of the form with our fields displayed.

Preview

UPEI UNIVERSITY of Prince Edward ISLAND **Islandora Virtual Environment**

Digital Repository **Create content** **Administer**

Home > Administer > Content management > XML Form Builder > Preview Form

Preview Form

Browse for your PDF.

Title: *

This is the title.

Creator(s)

The creator(s) of the work.

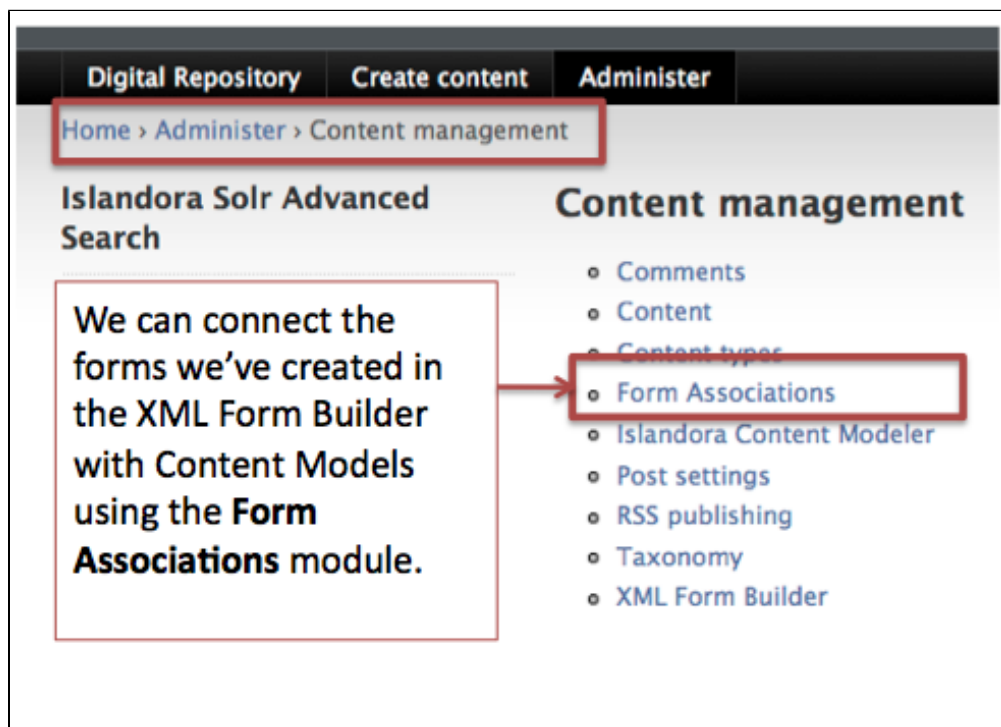
Description:

Form Associations Module

Connecting the Form to a Content Model

Once you've completed and tested the form you've created in the XML Form Builder module, you can connect that form to an existing content model using the Form Associations module. You can navigate to the module using this path:

Administer > Content management > Form Associations



In this example we will associate the form we created - **oai_dc_basic** - with the **islandora:sp_strict_pdf** content model. Review the image below to fill out the Form Associations dialog.

Form Associations Dialog

Form Associations

Content Model Datastream ID Title Field Form Transform Has Template Remove

Add Association

Content Model: *

islandora:sp_strict_pdf

The content model to associate with a form.

If the content model has no descendants it will not show up in a

Metadata Datastream ID: *

DESCMD

The datastream ID of where the objects metadata is stored.

Form Name:

oai_dc_basic

The name of the form to associate with the content model.

Title Field: *

['title']

The form field that you want to use for the objects label.

XSL Transform:

dc_no_transform.xsl

A xsl transform for setting the Fedora Object's Dublin Core me

Upload Template Document:

Browse...

A sample metadata file used to prepopulate the form on ingest.

Add Association

Enter the PID of the Content Model you are associating your form with.

Select the Datastream ID that the form will create on ingest. In our example we are creating a DC metadata datastream.

Select the form you created with the XML Form Builder.

Select the form field that holds the title information.

Select the file that transforms your metadata schema to DC. Since our form creates DC, we can select the dc_no_transform.xsl file.

Click on Add Association.

Navigating to a Collection

Once the association has been successfully created, you can try ingesting new objects into a collection that has the [islandora:sp_strict_pdf](#) content model associated with it. Alternatively, access the Islandora demo VM which has a PDF collection associated with the [islandora:sp_strict_pdf](#) content model. Navigate to the PDF collection using the image as a guide and try creating new PDF objects there.

Digital Repository Create content Administer


Home > Digital repository > Digital Repository

Islandora Solr Advanced Search


Title [dropdown]
[text input]
and [dropdown]
Title [dropdown]
[text input]
and [dropdown]
Title [dropdown]
[text input]
search

View Add Object Details


Select **Digital Repository**, then the **Basic PDFs** collection.




Basic PDFs



Basic Images



Newspapers Collection



Specimens

Once in the collection (you can tell where you are by the breadcrumb), select the Add tab to add a new PDF to the collection.

Adding a PDF to the Collection

Digital RepositoryCreate contentAdminister

Home > Digital repository > Basic PDFs > Digital Repository

Digital Repository

ViewAddObject


Select Add to ingest a new item into the collection.

Evergreen 3.8 Documentation
Book Review

Evergreen 3.8 Documentation
Book Review

Evergreen 3.8 Documentation
Book Review

Installing, Configuring
and Using

islandora™

The Robust
Open-Source
Digital Asset
Management System

Evergreen Documentation

Islandora Manual

Objects already in the Basic PDFs collection.

You will be presented with a dialog requesting two pieces of information: the content model and the form to use for ingest.

Selecting the Associated Form


Home > Digital repository > Islandora PDF Demo Collection > Digital Repository

Digital Repository

View Add Object Details


Ingest digital object into *islandora:pdf_collection* Step #1

Content models available:

ADD PDF  We'll be adding a PDF.

Content models define datastream composition, relationships between this and other content models. Additional information may be found [here](#).

Select form:

oai_dc_basic: (DESCMD)  Select the oai_dc_basic form.

Select the form to populate the metadata of the new object.

Next Select Next to move on to the form.

Entering Metadata


You will be presented with the form that we created during this tutorial. You'll need to fill it in. Once you've completed your data entry, submit the form.

Home > Digital repository > Connecting Research Data and Indigenous Communities > Digital Repository

Digital Repository

Description Read Online Object Details

View Edit Select Edit to edit your metadata.

 View Document

MetaData

title	Connecting Research Data and Indigenous Communities
creator	Kirsten Thorpe
creator	Elizabeth Mulhollann
subject	Systems and Information Theory
subject	Information Systems
description	In this poster, we propose to demonstrate the workflow and program of consultation developed by the Connecting Research Data and Indigenous Communities Data Archive (ATSIDA) to support the digital return of research data to Indigenous Australian communities for preservation and reuse in both the research community and by the general public.
date	06/13/2011
type	Text
identifier	islandora:21
rights	Rights held by the authors. Contact elizabeth.mulhollann@uts.edu.au for permission to reuse.

Digital Repository

Description

Read Online

Object Details

View | Edit |

Browse...

Browse for your PDF.

Title: *

Connecting Research Data and Indigenous Communities

This is the title.

Creator(s)

Kirsten Thorpe



The creator(s) of the work.

Elizabeth Mulhollann



Description:

In this poster, we propose to demonstrate the workflow and program of consultation developed by the Aboriginal and Torres Strait Islander Data Archive (ATSIDA) to support the digital return of research data to Indigenous Australian communities, while also facilitating data preservation and reuse in both the research community and by the general public.

The description of the work.

The same form used to create, can also be used to edit the metadata.