How to Create a Custom Ingest Form

The following 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. For an overview of how Islandora handles descriptive metadata, read Chapter 14 - Descriptive Metadata and Islandora. This section will discuss how to create a new form using the Islandora Form Builder.

Introduction

The XML Forms Builder allows you to create and manipulate **xml form templates** and affiliate them with **content models.** This means you can create custom forms for users to ingest items into collection by affiliating **the content model with a form.** You may want to create a custom metadata form 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 Chapter 10: Enabling Form Creation with the XML Forms Modules, which discusses the installation and configuration of the module.
- 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* (.xsds), 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

Required Modules

Ensure that you have downloaded and enabled the following Drupal modules which are available from the Islandora.ca Download page:

- islandora_content_model_forms
- islandora_xml_forms
- php_lib
- objective_forms
- tabs

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 here

Sample XML Record

It is incredibly 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 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

When developing a form it can be useful to have an XML editor to test code 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.

XML Form Builder

XML Form Builder is a Drupal module that integrates the creation of XML based forms into Islandora. Once a form has be 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 created you will be able to create and edit your metadata.
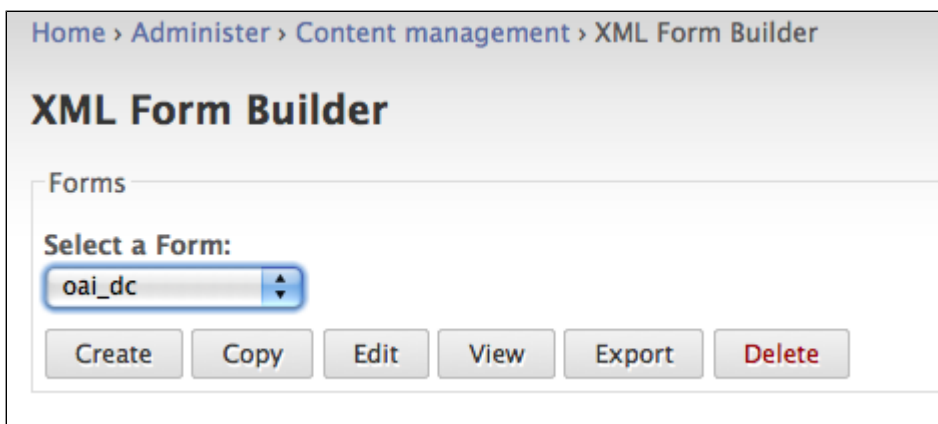
To use the XML Form Builder navigate to the module in your Islandora site:

Administer > Content Management > XML Form Builder
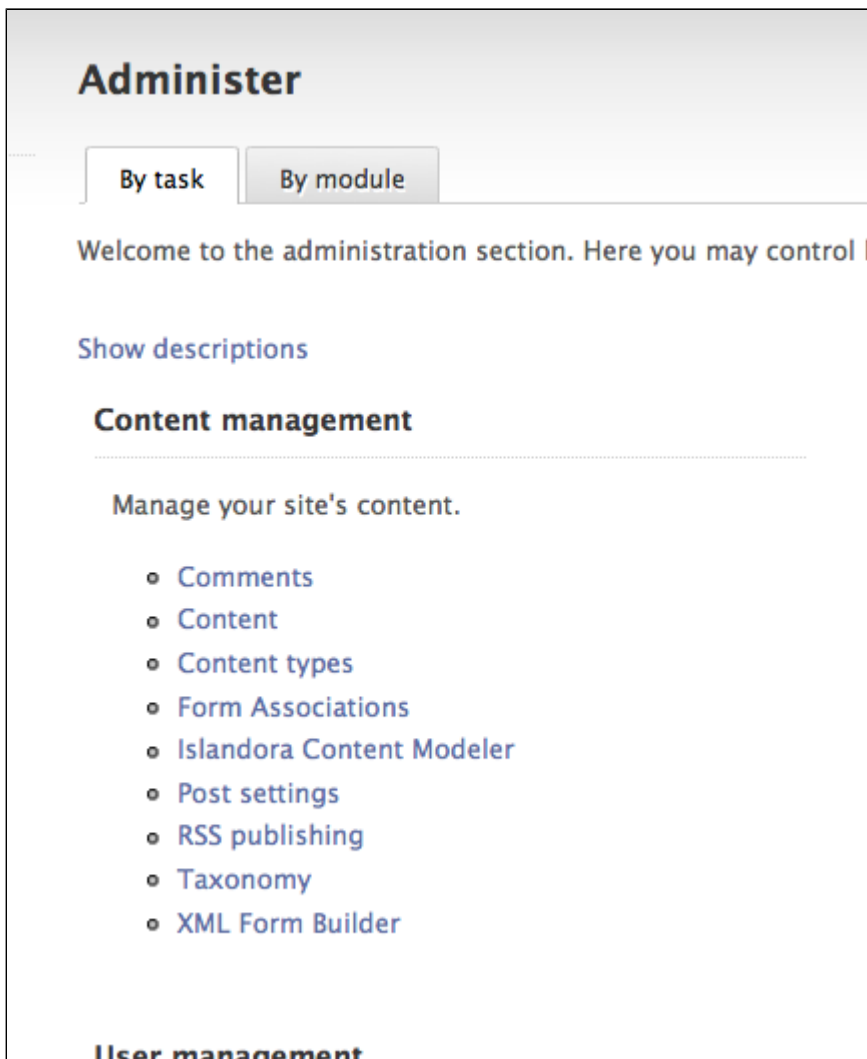
Form Builder Interface

When you start the module you are presented with an option to select a form from a list (those are the forms that come bundled with your Islandora install) and a series of buttons.

- Create
  - Select Create 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 (from the dropdown), 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.

Creating a Form

To start creating a form select Create.

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.

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

Create Form

Form Name: *

oai_dc_basic

Form Definition:

[   ] Browse...

An optional XML form definition template.

Create   Cancel

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

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

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.

Select Save once you have the properties entered. Once the form properties have been entered you are ready to add elements to your form.

Adding form fields

Some schema require elements appear in a certain order (an order based schema), that certain elements be required or not, that some elements are repeatable and others not, or that elements can be nested. Refer to your schema documentation and to the guidance in creation of records based on the schema. For the schema we are using in this tutorial, Simple Dublin Core, each of the fifteen elements is optional and may be repeated. We will create a simple form for this exercise that utilizes several Dublin Core elements. Let's create a table listing the elements, the element labels, the type of element, the content of the element, and whether they are repeatable. We'll use this information when adding elements 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 start adding elements to the form starting with the title element. We can use the information in our table to fill out the first part of Element Form.

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: Identifies this form field. It ishe Drupal form array key for this element.

Type: Used to determine the type of form field.

used as **Title:** The label of the form field as it appears on your form.

Description: The description of the form field.

Default Value: The value of the form field that will be displayed or selected initially 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 in which the path with be executed in.

document - xpath query is run from the root element of the document

parent - xpath query is run from the node created/read by its parent's form field

self - only applies 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 whether this element is inserted.

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 snipped is expected in the value field.

Value: If the type is either Element or Attribute, the name of the element or attribute is expected. If the type is XML, an XML snippet is expected where the value of the form field will be inserted where ever the string %value% is used in the XML snippet.

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.*

Checking the Create box opens up the Create dialog.

The path context is either **parent** or **document**. For the OAI DC form we will be using **document** as the path context.

Create

Path Context: document

Path: /oai_dc:dc

Schema:

Type: element

Value: dc:title

This is the path where the element will be create ... the parent of the element you are creating.

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.

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

<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.)

Read

Path Context: document

Path: /oai_dc:dc/dc:title

The path context is either **parent** or **document**. For the OAI DC form we will be using **document** as the Read path context.

This is the full path where the element will be read from.

Update

Path Context: self

Path: self::node()

Schema:

For Update the path context options are **parent**, **document**, or **self**. In this case the path context for updating the element will be **self**.

self::node() is the nodeset containing only the context node.

Delete

Path Context: self

Path: self::node()

We will use the same values for Delete as we did for Update.

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

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

Element Form

Identifier: 0

| Common Form Controls | Advanced Form Controls | More |

Type: tag

For these type of elements, the practice is to use 0 as the identifier.

*Select the **tag** type of element.*

Title:

Description:

Default Value:

Required: ☐

☑ Create

Path Context: document

Path: /oai_dc:dc

Schema:

Type: element

Value: dc:creator

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

☑ Read

Path Context: document

Path: /oai_dc:dc/dc:creator

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

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

For element that require a description or that contain a large amount of text, the textarea type form field is used.

Element Form

Identifier: description ← Enter an identifier.

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

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

Title: Description

Description: The description of the work.

Default Value:

Required: ☐

☑ Create

Path Context: document

Path: /oai_dc:dc

Schema:

Type: element

Value: dc:description

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

☑ Read

Path Context: document

Path: /oai_dc:dc/dc:description

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.

InteractiveResource	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.

Element Form

Identifier: type   ←   Enter an identifier.

Common Form Controls   Advanced Form Controls   More Advanced Controls   ←

Type: select

Select the **select** type of form element.

Title: Type

Description: Select the type of element.

Default Value: Text

Required: ☐

☑ Create

Path Context: document

Path: /oai_dc:dc

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

Schema:

Type: element

Value: dc:type

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.

2. To add terms to your select form field you need to click on the More Advanced Controls tab in the Element Form pane. Review the image below and enter your terms in the Options panel.
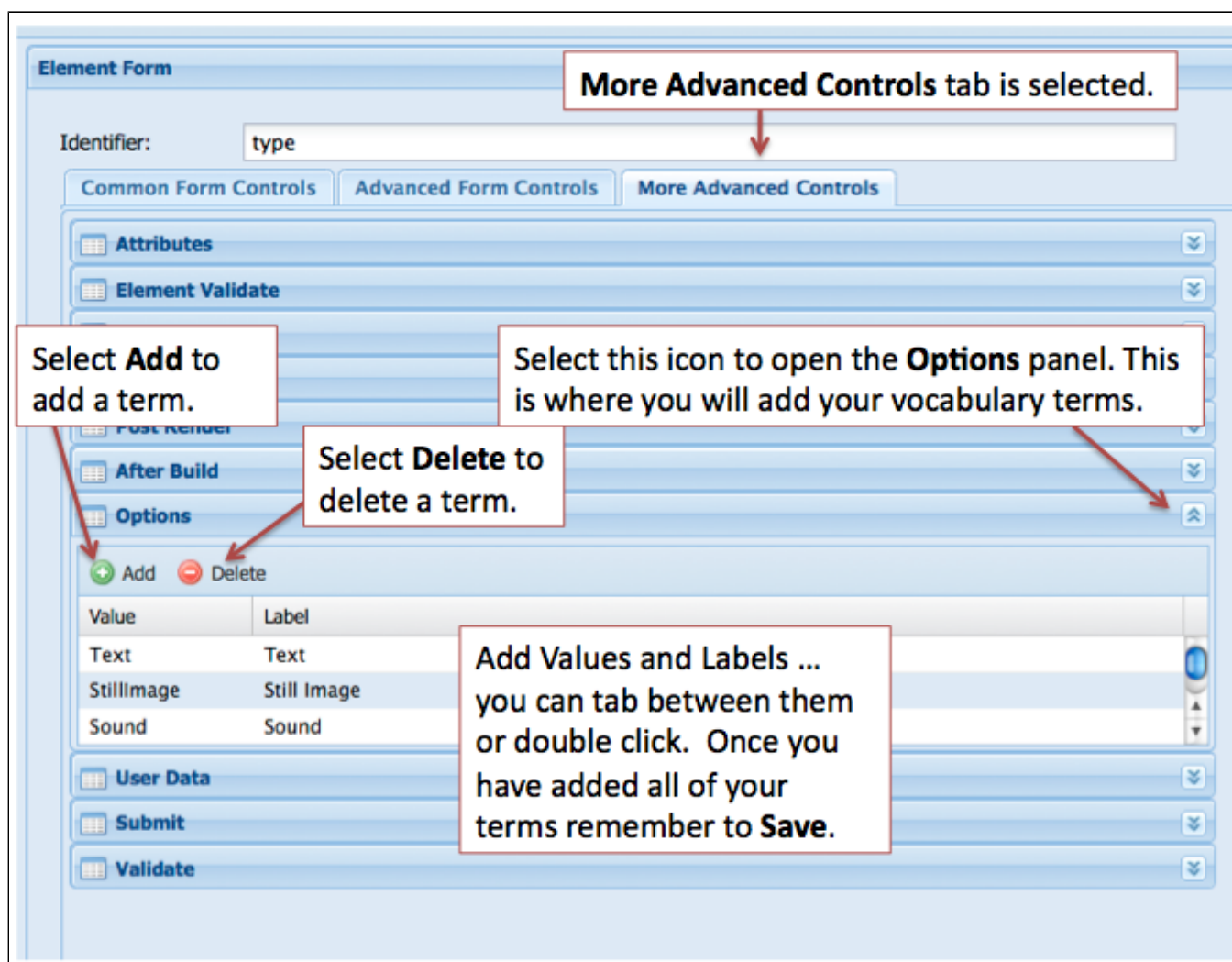
Adding a datepicker type form field - the dc:date element

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: date → Enter an identifier.

Common Form Controls | Advanced Form Controls | More Advanced (

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

Title: Date

Description: Date the item was created, issued, or published.

Default Value:

Required: ☐

☑ Create

Path Context: document

Path: /oai_dc:dc

Schema:

Type: element

Value: dc:date

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

☑ Read

Path Context: document

Path: /oai_dc:dc/dc:date

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 Indentifier for this form field type **must** be **ingest-file-location**.

Element Form

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

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

Type: file   ˅   ←   Select the **file** type of element.

Title:

Description: Browse for your PDF.

Default Value:

Required: ☐

— ☐ Create

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

— ☐ Update

— ☐ Delete

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...

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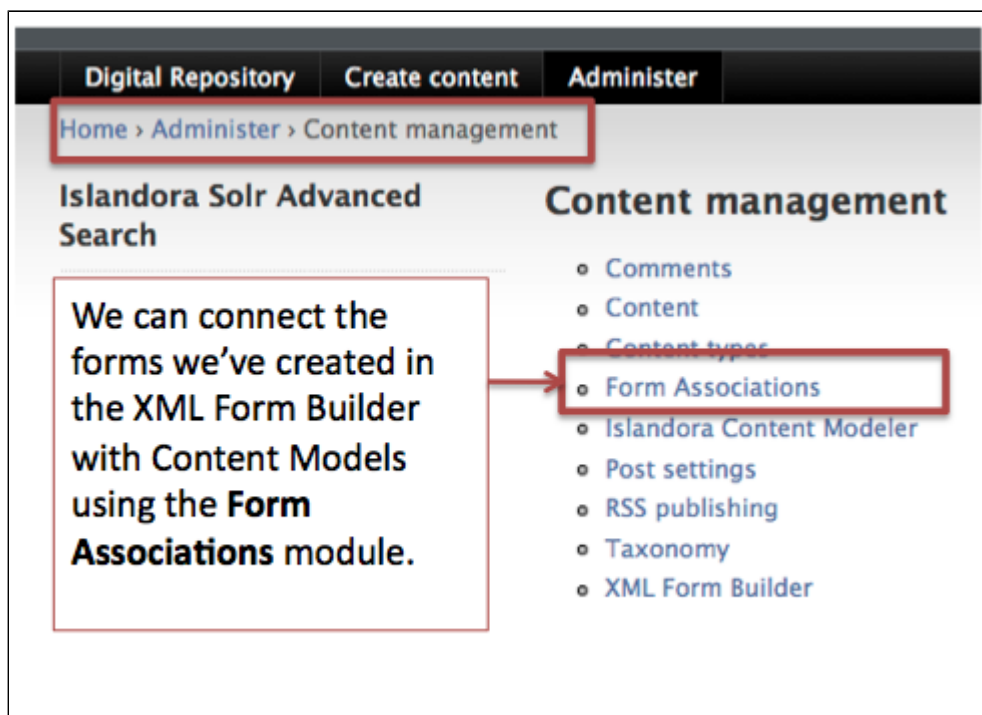
In this example we will associate the form we created - **oai_dc_basic** - with the **islandora:sp_strict_pdf** content model. Review the image below to fill out the Form Associations dialog.

Form Associations Dialog

Form Associations

Content Model	Datastream ID	Title Field	Form	Transform	Has Template	Remov

Add Association

Content Model: *

`islandora:sp_strict_pdf`

The content model to associate with a form.

If the content model has no decendants 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. In this case, the Islandora demo VM has a PDF collection associated with the islandora:sp_strict_pdf content model. Navigate to the PDF collection using the image as a guide.

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

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

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.

Digital Repository

| Description | Read Online | Object Details |

View | Edit

Choose edit form.

Forms:

oai_dc_basic: (DESCMD) ⇕ ← Select your form.

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

Next ← Click on Next.

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 a
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 genera
public.]

The description of the work.

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