LDP-PCDM-F4 In Action

Due to the evolution of usage patterns for PCDM (particularly as they relate to Fedora), the structure of resources described in this guide is not necessarily the recommended structure.
However, the details and examples of how to use LDP Direct and Indirect Containers are still very informative in understanding how those constructs work.

-- Andrew Woods 2015-10-30

Fedora4 (F4) implements the Linked Data Platform (LDP) W3C Recommendation. Additionally, the Portland Common Data Model (PCDM) has increasingly become adopted as a common content modeling approach in Fedora4.

LDP defines terminology and interaction models relating to linked data resources and servers. The "action" in inter"action" models should be emphasized, because LDP introduces two new concepts that enhance the actions within a linked data server, F4 in this case. These concepts are:

- DirectContainer
- IndirectContainer

These two container types have associated behavior that are highlighted and clarified here, in this guide. Two different uses of DirectContainers are illustrated in the Book and Ordering examples, respectively, and the use of IndirectContainers is described in the Collections example.

This guide is designed to describe the details of both LDP and PCDM in the context of F4 by walking through a simple example of a single collection, consisting of a single book that implements page ordering.

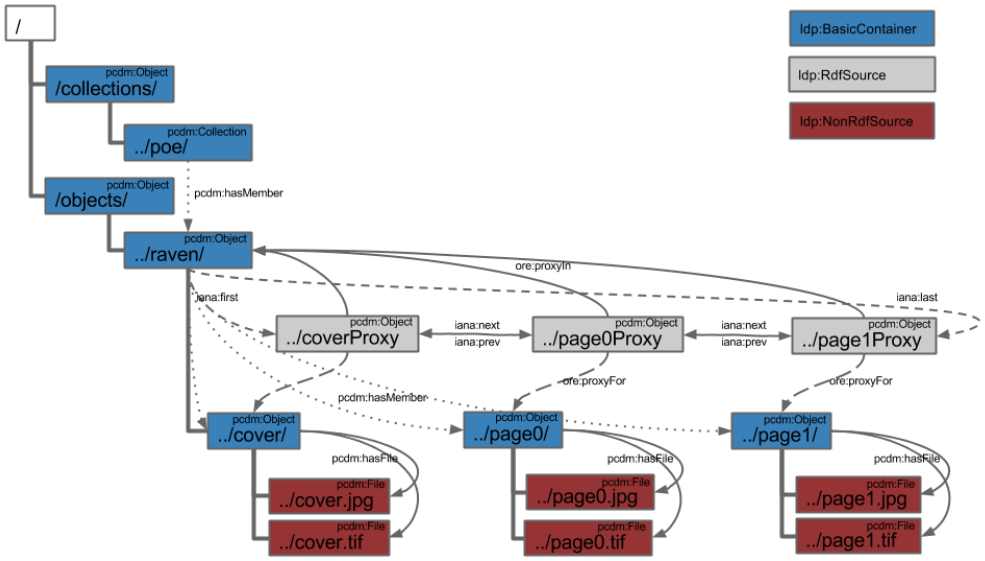
Note0: Although the following example uses specifically named resources, such as "poe" and "raven", production scenarios will likely use opaque identifiers/URLs by allowing them to be auto-generated by F4.
Note1: An easy way to stand-up an environment for executing the following REST requests is to use fcrepo4-vagrant.
Note1a: F4 is deployed in the fcrepo4-vagrant box with the context "fcrepo". If you are deploying in another environment, you may need to change the "curl" requests and turtle files (*.ttl) below to reflect a different context.
Note2: Please use your Internet browser to inspect the results of each of the steps below! http://localhost:8080/fcrepo/rest

The full slide deck of images are available for download.

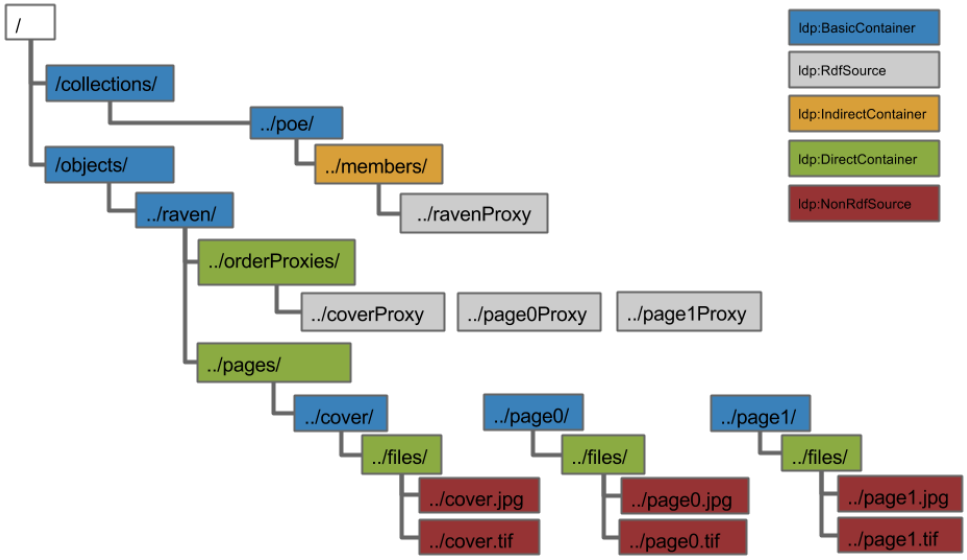
End Goal - Final State

0: Final State - Complete

This diagram depicts the logical structure of resources and relationships of a collection ("poe") that consists of a book ("raven") that consists of three pages that are ordered.



This is a diagram of the same collection, book, and pages, but depicting the directory structure and LDP resource types.



The following steps will walk through the process of creating this structure from the ground up.
The different types of LDP containers and sources will be detailed, as well as the PCDM types and relationships in the steps below.

Books In Action

1: Final State - Book

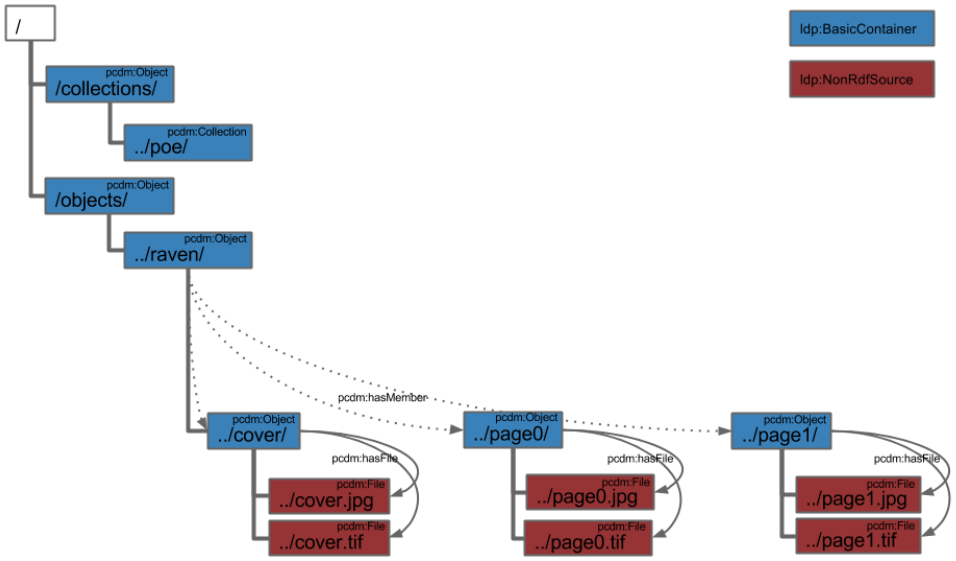
The ldp:BasicContainers are simply containers of other resources. BasicContainers can contain both other containers as well as ldp:NonRdfSources (or "binaries").
There are three PCDM types here:

- pcdm:Object
- pcdm:Collection
- pcdm:File

Additionally, there are two PCDM relationships that indicate resource membership and file membership:

- pcdm:hasMember
- pcdm:hasFile

The descriptions of these resource types and relationships may be found in the detailed Portland Common Data Model page.

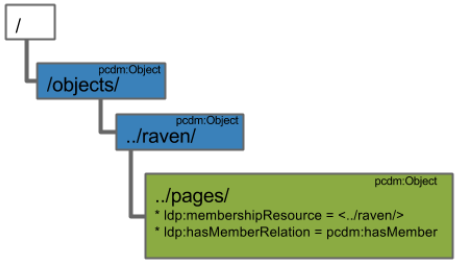


- Book - Create DirectContainer
- Book - Create Cover
- Book - Create Page0
- Book - Create Page1
- Cover - Create DirectContainer
- Cover - Create Files
- Page0 - Create DirectContainer
- Page0 - Create Files
- Page1 - Create DirectContainer
- Page1 - Create Files
- Book - Conclusion

Book - Create DirectContainer

Book - Create DirectContainer

Here we will begin to walk through the mechanics of creating the structures that will facilitate creation of the book and its pages.



First, create the top-level "objects/" pcdm:Object, which is also an ldp:BasicContainer.

```
curl -i -XPUT -H"Content-Type: text/turtle" --data-binary @pcdm-object.ttl localhost:8080/fcrepo/rest/objects/
```

Where "pcdm-object.ttl" follows:

pcdm-object.ttl

```
@prefix pcdm: <http://pcdm.org/models#>

<> a pcdm:Object .
```

Second, create the nested "raven/" pcdm:Object, which is also another ldp:BasicContainer.

```
curl -i -XPUT -H"Content-Type: text/turtle" --data-binary @pcdm-object.ttl localhost:8080/fcrepo/rest/objects
/raven/
```

Lastly, create an ldp:DirectContainer, "pages/" that will facilitate the establishment of relationships between "raven/" and its constituent pages.

```
curl -i -XPUT -H"Content-Type: text/turtle" --data-binary @ldp-direct.ttl localhost:8080/fcrepo/rest/objects
/raven/pages/
```

Where "ldp-direct.ttl" follows:

ldp-direct.ttl

```
@prefix ldp: <http://www.w3.org/ns/ldp#>
@prefix pcdm: <http://pcdm.org/models#>

<> a ldp:DirectContainer, pcdm:Object ;
  ldp:membershipResource </fcrepo/rest/objects/raven/> ;
  ldp:hasMemberRelation pcdm:hasMember .
```

An ldp:DirectContaner is an LDP construct that activates the creation of certain RDF triples when a new resource is added as a child of this container. Specifically, when a new resource is added inside of the "pages/" DirectContainer, a new triple on the ldp:membershipResource ("raven/") will be created with the predicate defined by the ldp:hasMemberRelation property ("pcdm:hasMember") and an object that is a reference to the new resource.

The auto-created triple resulting from the addition of a new child resource within "pages/" will take the form:

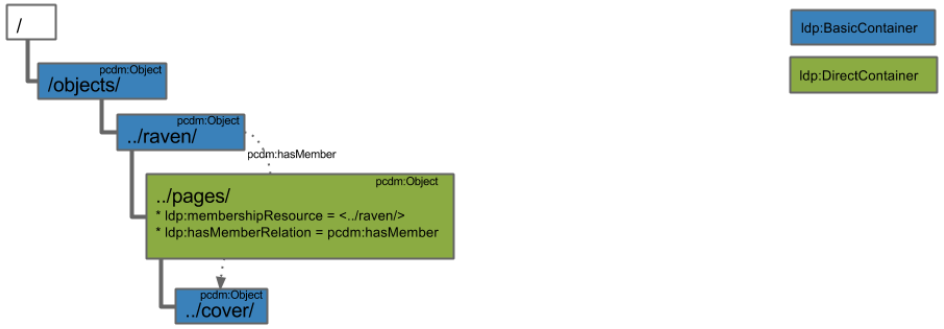
```
<http://localhost:8080/fcrepo/rest/objects/raven/> <pcdm:hasMember> <new-resource>
```

We will see this in action next!

Book - Create Cover

Book - Create cover

Create a new pcdm:Object, "cover/", that is also an ldp:BasicContainer within the "pages/" DirectContainer.




```
curl -i -XPUT -H"Content-Type: text/turtle" --data-binary @pcdm-object.ttl localhost:8080/fcrepo/rest/objects
/raven/pages/cover/
```

Where "pcdm-object.ttl" follows:

pcdm-object.ttl

```
@prefix pcdm: <http://pcdm.org/models#>

<> a pcdm:Object .
```

As described in the previous step, the addition of "cover/" automatically creates the following new triple on "raven/"

```
<http://localhost:8080/fcrepo/rest/objects/raven/> pcdm:hasMember <http://localhost:8080/fcrepo/rest/objects
/raven/pages/cover/>
```

Restating from the previous step,

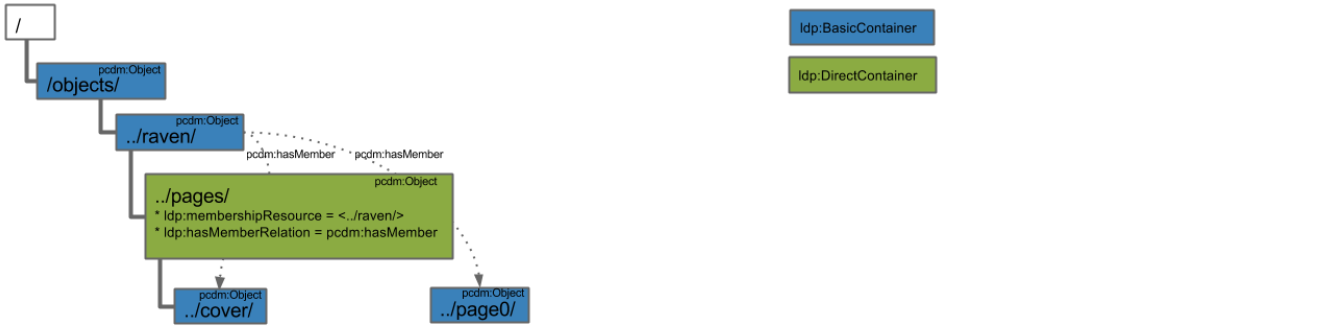
- the subject of the triple comes from the "ldp:membershipResource" defined on "pages/"
- the predicate of the triple comes from the "ldp:hasMemberRelation" defined on "pages/", and
- the object of the triple is the new resource ("cover/") that was added to the ldp:DirectContainer ("pages/")

Book - Create Page0

Book - Create Page0

In the same fashion as the previous step, adding "page0/" to the DirectContainer, "pages/" results in a new auto-generated triple on "raven/" of the form:

```
<http://localhost:8080/fcrepo/rest/objects/raven/> pcdm:hasMember <http://localhost:8080/fcrepo/rest/objects
/raven/pages/page0/>
```

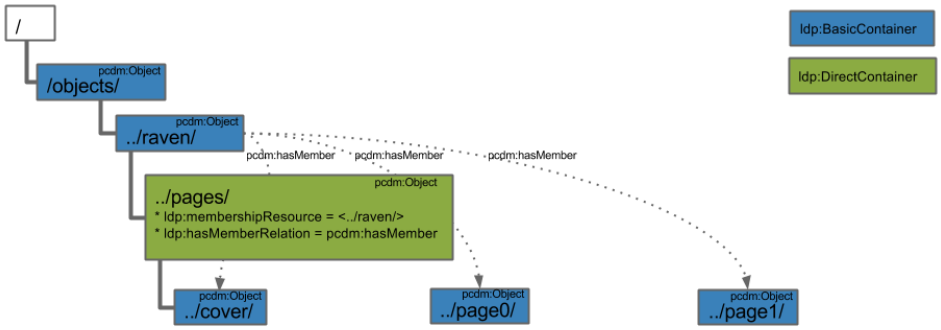



```
curl -i -XPUT -H"Content-Type: text/turtle" --data-binary @pcdm-object.ttl localhost:8080/fcrepo/rest/objects
/raven/pages/page0/
```

Book - Create Page1

Book - Create Page1

This step in creating the final page, "page1/", follows the same pattern shown in the previous two steps.

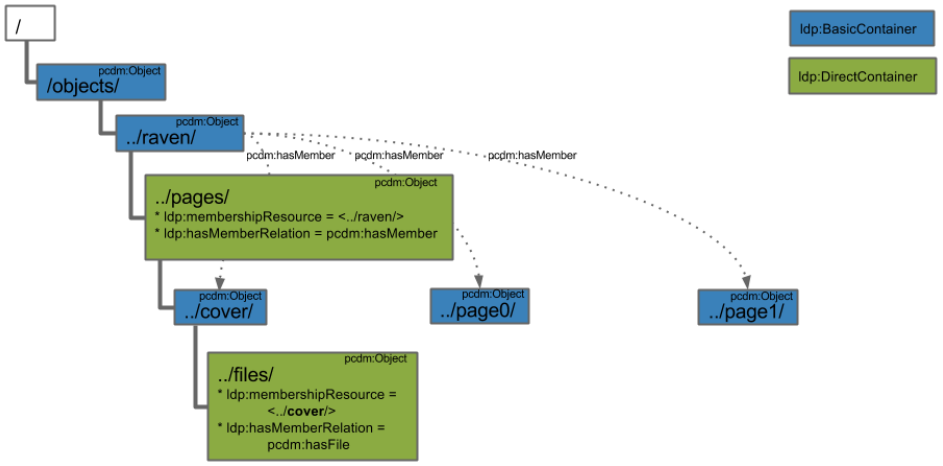



```
curl -i -XPUT -H"Content-Type: text/turtle" --data-binary @pcdm-object.ttl localhost:8080/fcrepo/rest/objects
/raven/pages/page1/
```

Cover - Create DirectContainer

Cover - Create DirectContainer

In the same way that we used an ldp:DirectContainer to facilitate the auto-generation of triples linking "raven/" to each of the pages, now use the same pattern to auto-generate the creation of triples that link each page pcdm:Object to their various file representations.



To begin with, create an ldp:DirectContainer, "files/", which is also a pcdm:Object, as a child of "cover/" as follows:

```
curl -i -XPUT -H"Content-Type: text/turtle" --data-binary @ldp-cover-direct.ttl localhost:8080/fcrepo/rest
/objects/raven/pages/cover/files/
```

Where "ldp-cover-direct.ttl" follows:

ldp-cover-direct.ttl

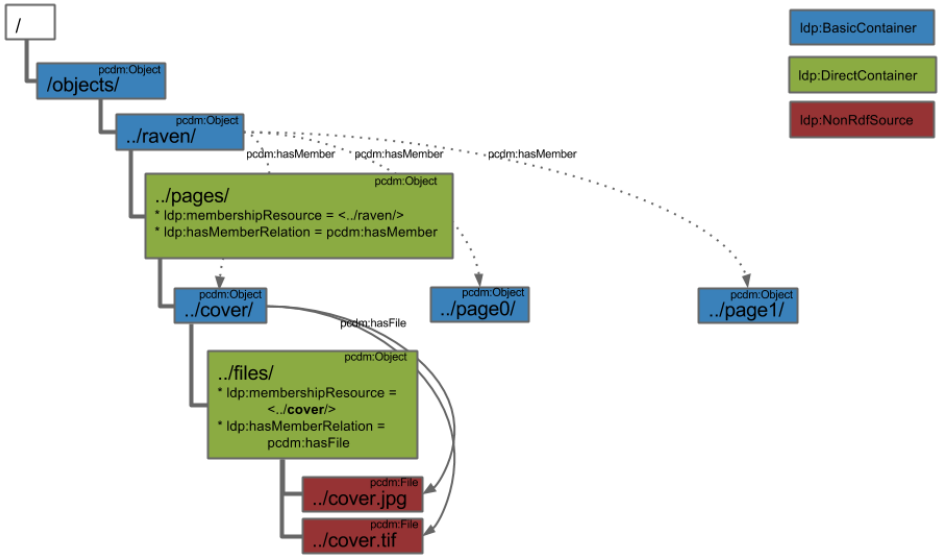
```
@prefix ldp: <http://www.w3.org/ns/ldp#>
@prefix pcdm: <http://pcdm.org/models#>

<> a ldp:DirectContainer, pcdm:Object ;
  ldp:membershipResource </fcrepo/rest/objects/raven/pages/cover/> ;
  ldp:hasMemberRelation pcdm:hasFile .
```

Now, any new resource that is added as a child of the DirectContainer "files/" will cause the auto-generation of a new triple on "cover/" that has a predicate of pcdm:hasFile and an object of the new resource.

Cover - Create Files

Cover - Create Files

Once again, we demonstrate the use of LDP in creating PCDM relationships simply as a result of repository interactions.



Add two pcdm:File resources to the DirectContainer, "files/" as follows:

```
curl -i -XPUT -H"Content-Type: image/jpeg" --data-binary @cover.jpg localhost:8080/fcrepo/rest/objects/raven
/pages/cover/files/cover.jpg
```

Where "cover.jpg" is attached.

If you perform a subsequent HTTP HEAD on this new resource, you will see there is a "Link" header of rel="describedby". Update the RDF metadata of the ldp:NonRdfSource to specify that the resource is a pcdm:File, as follows:

```
curl -i -XPATCH -H"Content-Type: application/sparql-update" --data-binary @pcdm-file.ru localhost:8080/fcrepo
/rest/objects/raven/pages/cover/files/cover.jpg/fcr:metadata
```

Where "pcdm-file.ru" follows:

pcdm-file.ru

```
PREFIX pcdm: <http://pcdm.org/models#>
INSERT {
  <> a pcdm:File
} WHERE {
}
```

Repeat for the attached TIFF, cover.tif

```
curl -i -XPUT -H"Content-Type: image/tiff" --data-binary @cover.tif localhost:8080/fcrepo/rest/objects/raven
/pages/cover/files/cover.tif
curl -i -XPATCH -H"Content-Type: application/sparql-update" --data-binary @pcdm-file.ru localhost:8080/fcrepo
/rest/objects/raven/pages/cover/files/cover.tif/fcr:metadata
```

After creating the two "cover" resources, an HTTP GET on "cover/" will include the two new triples:

```
<http://localhost:8080/fcrepo/rest/objects/raven/pages/cover/> pcdm:hasFile <http://localhost:8080/fcrepo/rest
/objects/raven/pages/cover/files/cover.jpg>
<http://localhost:8080/fcrepo/rest/objects/raven/pages/cover/> pcdm:hasFile <http://localhost:8080/fcrepo/rest
/objects/raven/pages/cover/files/cover.tif>
```

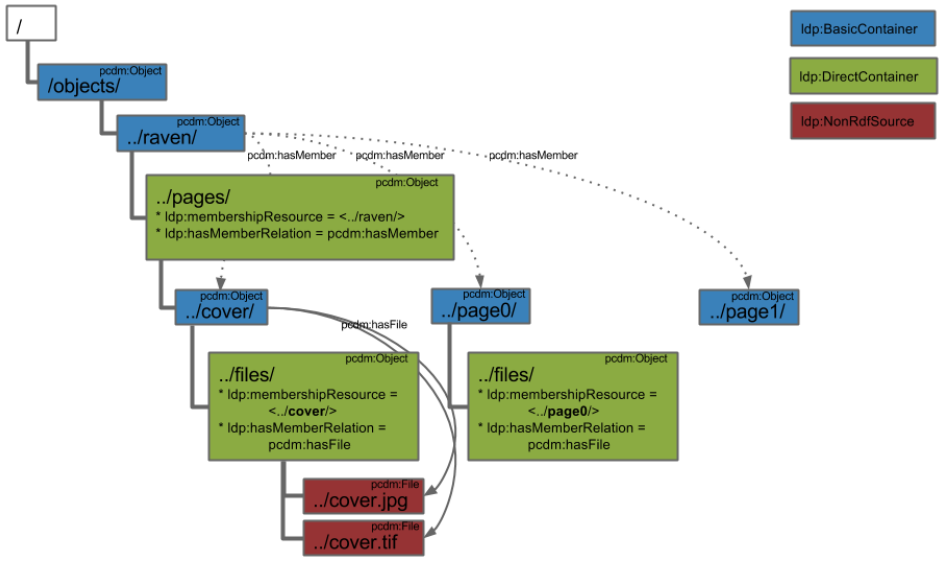
Once again,

- the subject of the triple comes from the "ldp:membershipResource" defined on "files/"
- the predicate of the triple comes from the "ldp:hasMemberRelation" defined on "files/", and
- the object of the triple is the new resource ("cover.jpg" or "cover.tif") that was added to the ldp:DirectContainer ("files/")

Page0 - Create DirectContainer

Page0 - Create DirectContainer

Here we repeat the exact steps as for the "cover/" above, but for "page0/".




```
curl -i -XPUT -H"Content-Type: text/turtle" --data-binary @ldp-page0-direct.ttl localhost:8080/fcrepo/rest
/objects/raven/pages/page0/files/
```

Where "ldp-page0-direct.ttl" follows:

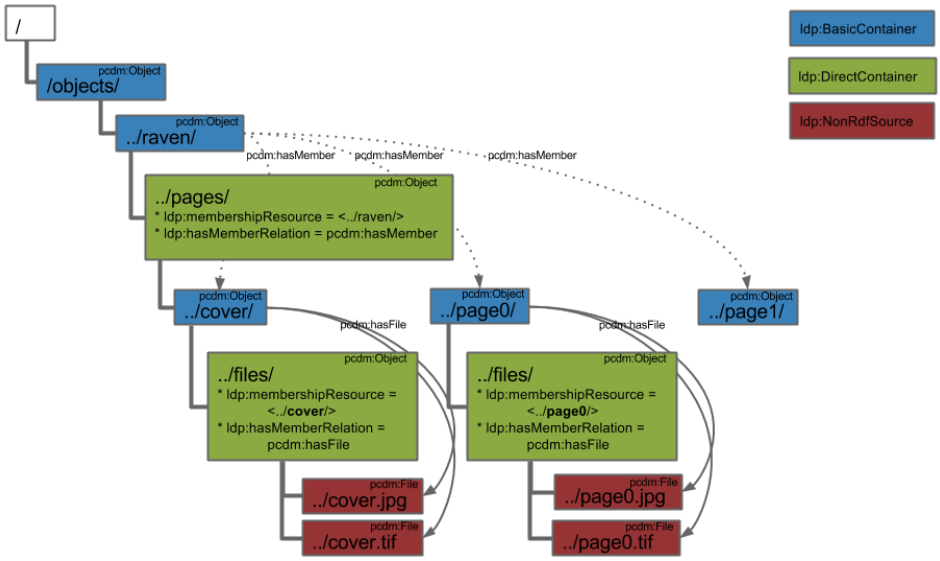
ldp-page0-direct.ttl

```
@prefix ldp: <http://www.w3.org/ns/ldp#>
@prefix pcdm: <http://pcdm.org/models#>

<> a ldp:DirectContainer, pcdm:Object ;
  ldp:membershipResource </fcrepo/rest/objects/raven/pages/page0/> ;
  ldp:hasMemberRelation pcdm:hasFile .
```

Page0 - Create Files

Page0 - Create Files

Here we add the attached page0 files (page0.jpg and page0.tif) to the newly created DirectContainer.




```
curl -i -XPUT -H"Content-Type: image/jpeg" --data-binary @page0.jpg localhost:8080/fcrepo/rest/objects/raven
/pages/page0/files/page0.jpg
curl -i -XPUT -H"Content-Type: image/tiff" --data-binary @page0.tif localhost:8080/fcrepo/rest/objects/raven
/pages/page0/files/page0.tif
```

Followed by assigning the type of pcdm:File to the respective RDF Sources found in the "Link; rel=describedby" header of each of the ldp:NonRdfSources.

```
curl -i -XPATCH -H"Content-Type: application/sparql-update" --data-binary @pcdm-file.ru localhost:8080/fcrepo
/rest/objects/raven/pages/page0/files/page0.jpg/fcr:metadata
curl -i -XPATCH -H"Content-Type: application/sparql-update" --data-binary @pcdm-file.ru localhost:8080/fcrepo
/rest/objects/raven/pages/page0/files/page0.tif/fcr:metadata
```

Where "pcdm-file.ru" follows:

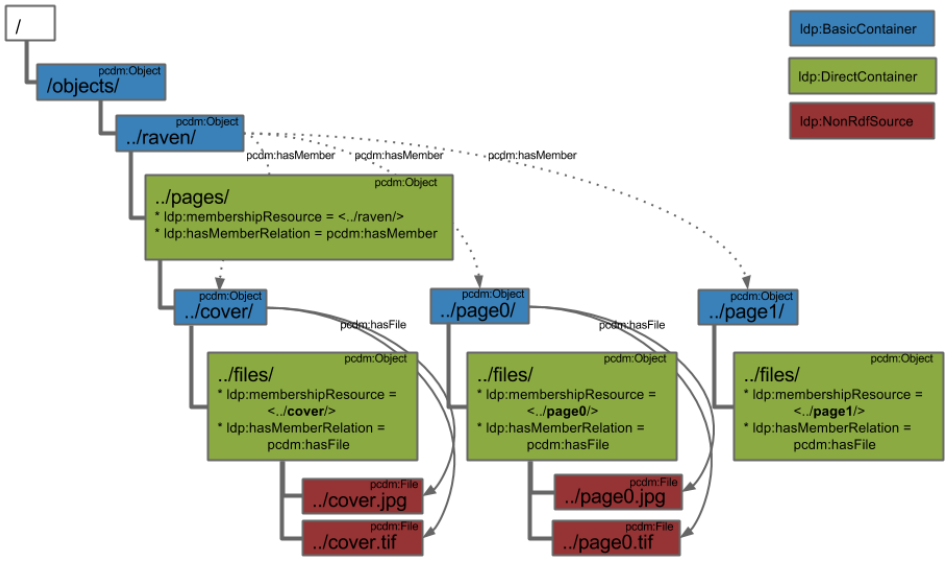
pcdm-file.ru

```
PREFIX pcdm: <http://pcdm.org/models#>
INSERT {
  <> a pcdm:File
} WHERE {
}
```

Page1 - Create DirectContainer

Page1 - Create DirectContainer

Here we repeat the exact steps as for the "page0/" above, but for "page1/".




```
curl -i -XPUT -H"Content-Type: text/turtle" --data-binary @ldp-page1-direct.ttl localhost:8080/fcrepo/rest
/objects/raven/pages/page1/files/
```

Where "ldp-page1-direct.ttl" follows:

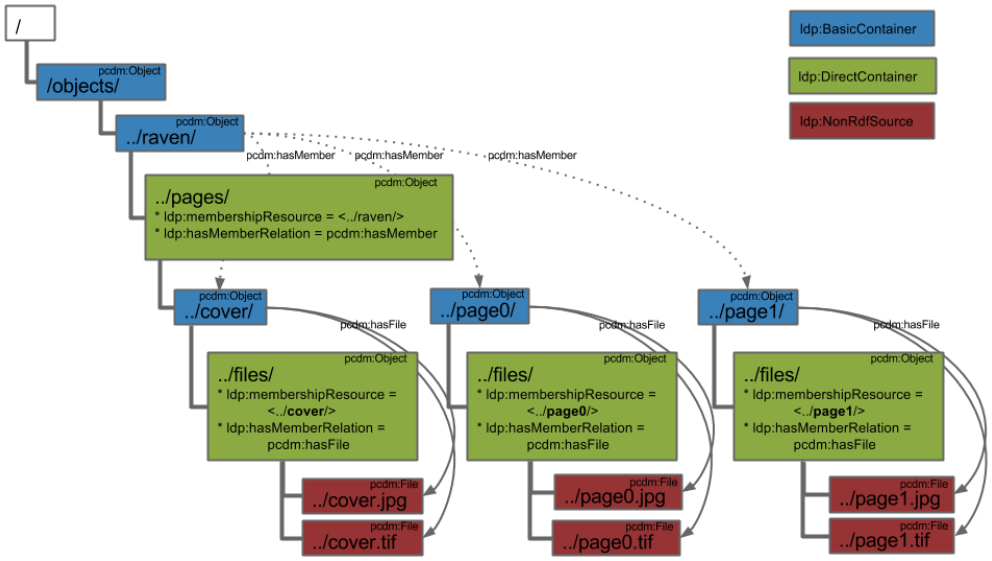
ldp-page1-direct.ttl

```
@prefix ldp: <http://www.w3.org/ns/ldp#>
@prefix pcdm: <http://pcdm.org/models#>

<> a ldp:DirectContainer, pcdm:Object ;
  ldp:membershipResource </fcrepo/rest/objects/raven/pages/page1/> ;
  ldp:hasMemberRelation pcdm:hasFile .
```

Page1 - Create Files

Page1 - Create Files

Finally, we add the attached page1 files (page1.jpg and page1.tif) to the newly created DirectContainer.




```
curl -i -XPUT -H"Content-Type: image/jpeg" --data-binary @page1.jpg localhost:8080/fcrepo/rest/objects/raven
/pages/page1/files/page1.jpg
curl -i -XPUT -H"Content-Type: image/tiff" --data-binary @page1.tif localhost:8080/fcrepo/rest/objects/raven
/pages/page1/files/page1.tif
```

Followed by assigning the type of pcdm:File to the respective RDF Sources found in the "Link; rel=describedby" header of each of the ldp:NonRdfSources.

```
curl -i -XPATCH -H"Content-Type: application/sparql-update" --data-binary @pcdm-file.ru localhost:8080/fcrepo
/rest/objects/raven/pages/page1/files/page1.jpg/fcr:metadata
curl -i -XPATCH -H"Content-Type: application/sparql-update" --data-binary @pcdm-file.ru localhost:8080/fcrepo
/rest/objects/raven/pages/page1/files/page1.tif/fcr:metadata
```

Where "pcdm-file.ru" follows:

pcdm-file.ru

```
PREFIX pcdm: <http://pcdm.org/models#>
INSERT {
  <> a pcdm:File
} WHERE {
}
```

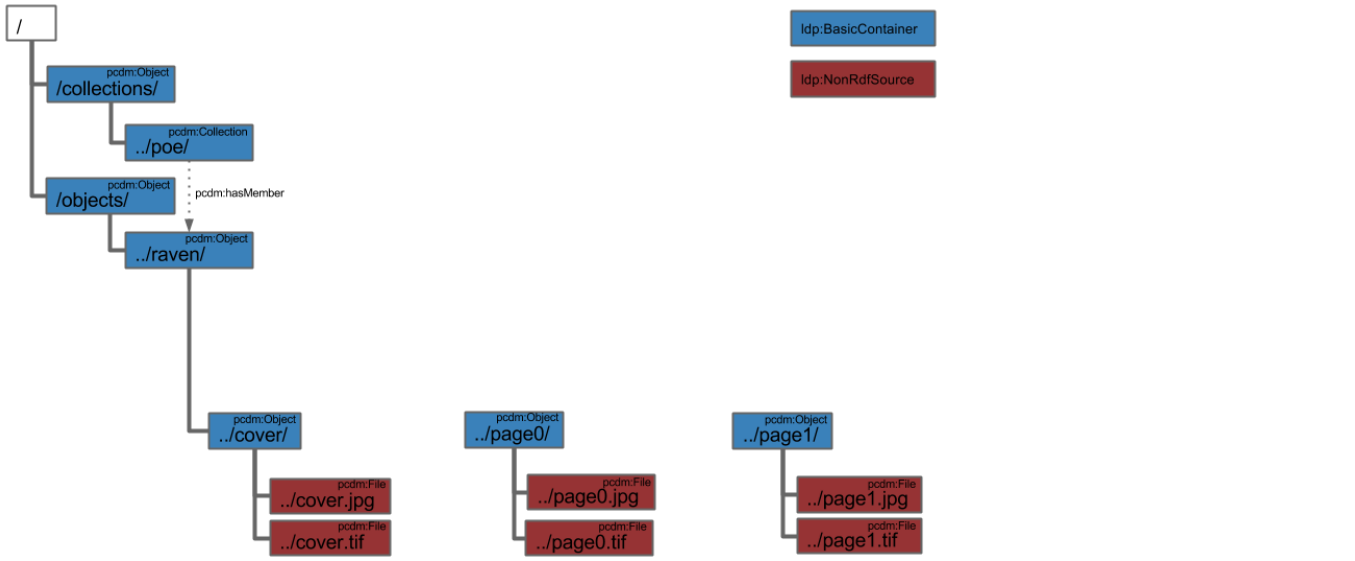
Book - Conclusion

Using LDP in conjunction with PCDM terms, we have created a book, "raven/", with its constituent pages and their file representations.

Collections In Action

2: Final State - Collection

Continuing with the previous example of modeling and creating a book with LDP, PCDM and F4, here we will detail an approach for adding that book, "raven/" to a new collection, "poe/".



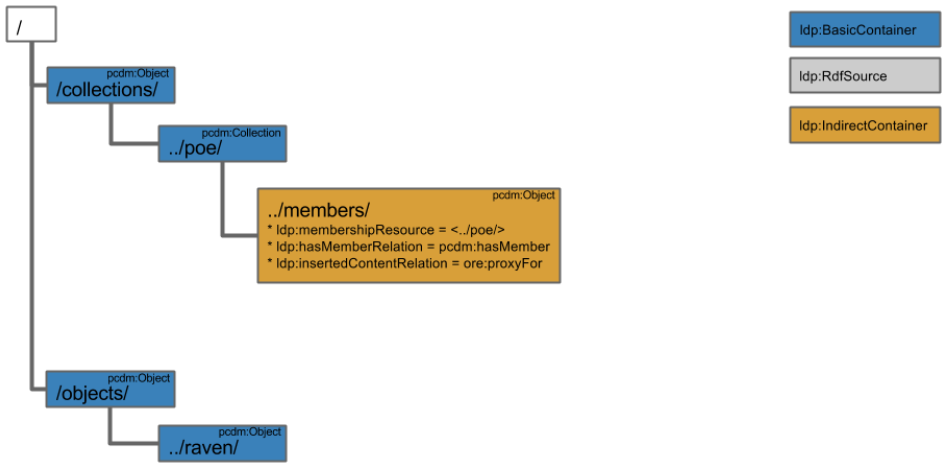
The objective in this section is to leverage LDP interaction models to not only create the appropriate pcdm:hasMember relationship between the collection "poe/" and the book "raven/", but to put the LDP structure in place for a simplified addition of new items to the "poe/" collection.

- Collection - Create IndirectContainer
- Collection - Create Raven Proxy
- Collection - Conclusion

Collection - Create IndirectContainer

Collection - Create IndirectContainer

Here we will begin to walk through the mechanics of creating the structures that will facilitate creation of the collection and its single member, in this case.



First, create the top-level "collections/" pcdm:Object, which is also an ldp:BasicContainer.

```
curl -i -XPUT -H"Content-Type: text/turtle" --data-binary @pcdm-object.ttl localhost:8080/fcrepo/rest
/collections/
```

Where "pcdm-object.ttl" follows:

pcdm-object.ttl

```
@prefix pcdm: <http://pcdm.org/models#>

<> a pcdm:Object .
```

Second, create the nested "poe/" pcdm:Collection, which is also another ldp:BasicContainer.

```
curl -i -XPUT -H"Content-Type: text/turtle" --data-binary @pcdm-collection.ttl localhost:8080/fcrepo/rest
/collections/poe/
```

Where "pcdm-collection.ttl" follows:

```
@prefix pcdm: <http://pcdm.org/models#>

<> a pcdm:Collection .
```

Lastly, create an ldp:IndirectContainer, "members/" that will facilitate the establishment of relationships between "poe/" and the collection members.

```
curl -i -XPUT -H"Content-Type: text/turtle" --data-binary @ldp-indirect.ttl localhost:8080/fcrepo/rest
/collections/poe/members/
```

Where "ldp-indirect.ttl" follows:

ldp-indirect.ttl

```
@prefix ldp: <http://www.w3.org/ns/ldp#>
@prefix pcdm: <http://pcdm.org/models#>
@prefix ore: <http://www.openarchives.org/ore/terms/>

<> a ldp:IndirectContainer, pcdm:Object ;
  ldp:membershipResource </fcrepo/rest/collections/poe/> ;
  ldp:hasMemberRelation pcdm:hasMember ;
  ldp:insertedContentRelation ore:proxyFor .
```

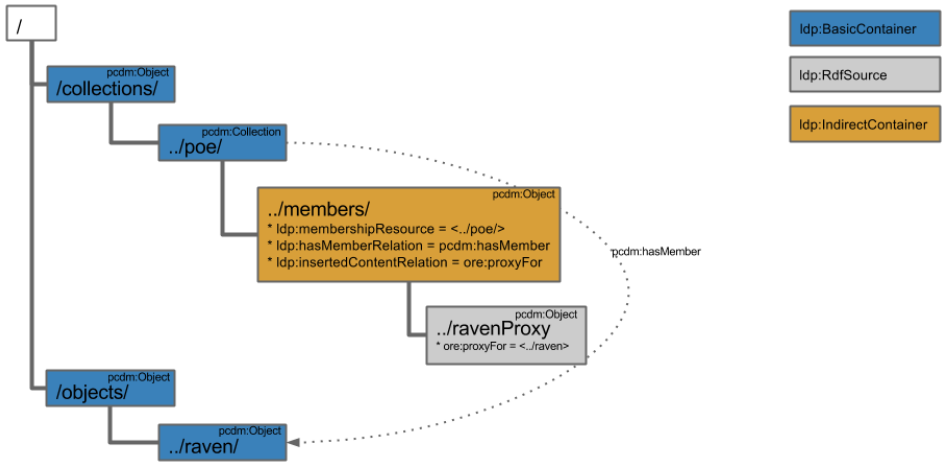
Similar to the previously described ldp:DirectContainer, an ldp:IndirectContainer is an LDP construct that also activates the creation of certain RDF triples when a new resource is added as a child of this container.
Just like with a DirectContainer, when a new resource is added inside of the "members/" IndirectContainer, a new triple on the ldp:membershipResource ("poe/") will be created with the predicate defined by the ldp:hasMemberRelation property ("pcdm:hasMember").
However, the difference from a DirectContainer is that the object of the created triple is not the newly added child, but instead the resource defined by the ldp:insertedContentRelation property (ore:proxyFor, in this case) found on the newly added child of this container.

We will see this in action next!

Collection - Create Raven Proxy

Collection - Create Raven Proxy

Create a new pcdm:Object, "ravenProxy/", that is an ldp:RdfSource within the "members/" IndirectContainer.




```
curl -i -XPUT -H"Content-Type: text/turtle" --data-binary @pcdm-raven-proxy.ttl localhost:8080/fcrepo/rest
/collections/poe/members/ravenProxy
```

Where "pcdm-object.ttl" follows:

pcdm-raven-proxy.ttl

```
@prefix pcdm: <http://pcdm.org/models#>
@prefix ore: <http://www.openarchives.org/ore/terms/>

<> a pcdm:Object ;
  ore:proxyFor </fcrepo/rest/objects/raven/> .
```

As mentioned in the previous step, the addition of "ravenProxy/" automatically creates the following new triple on "poe/".

```
<http://localhost:8080/fcrepo/rest/collections/poe/> pcdm:hasMember <http://localhost:8080/fcrepo/rest/objects
/raven/>
```

The ldp:IndirectContainer defines the creation of this triple as follows:

- the subject of the triple comes from the "ldp:membershipResource" defined on "members/"
- the predicate of the triple comes from the "ldp:hasMemberRelation" defined on "members/", and
- the object of the triple is the resource defined by the ldp:insertedContentRelation property (ore:proxyFor) found on the newly added child resource, "ravenProxy".

Collection - Conclusion

Using LDP in conjunction with PCDM terms, we have created a collection, "poe/", with its single member, "raven/".

Ordering In Action

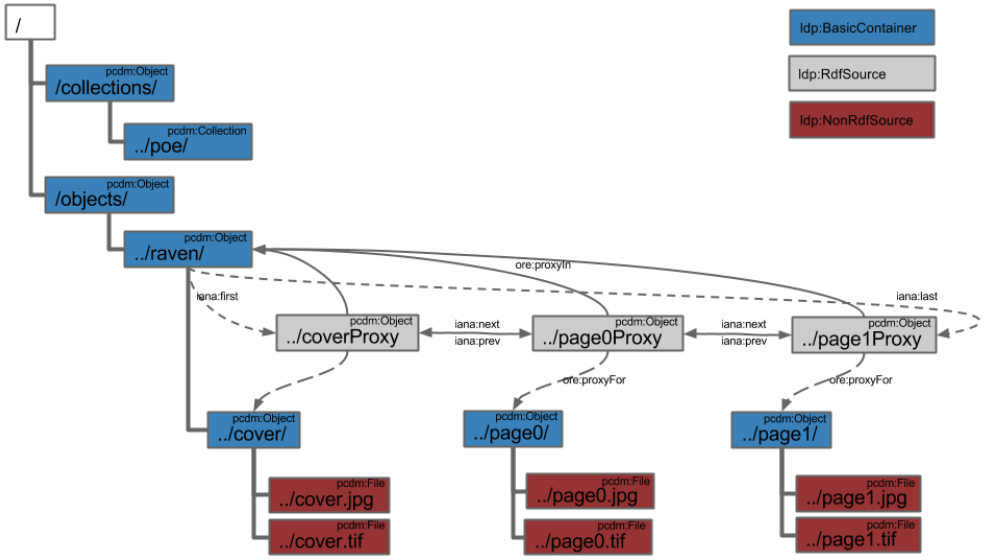
3: Final State - Ordered Pages

This final example will both illustrate a second use of ldp:DirectContainers as well as detail the PCDM recommendation for how to handle ordering of resources.

The addtional predicates/relationships that will be used in this example are:

- ore:proxyIn
- ore:proxyFor
- iana:first
- iana:next
- iana:prev
- iana:last

...all of which are further described in the Portland Common Data Model.



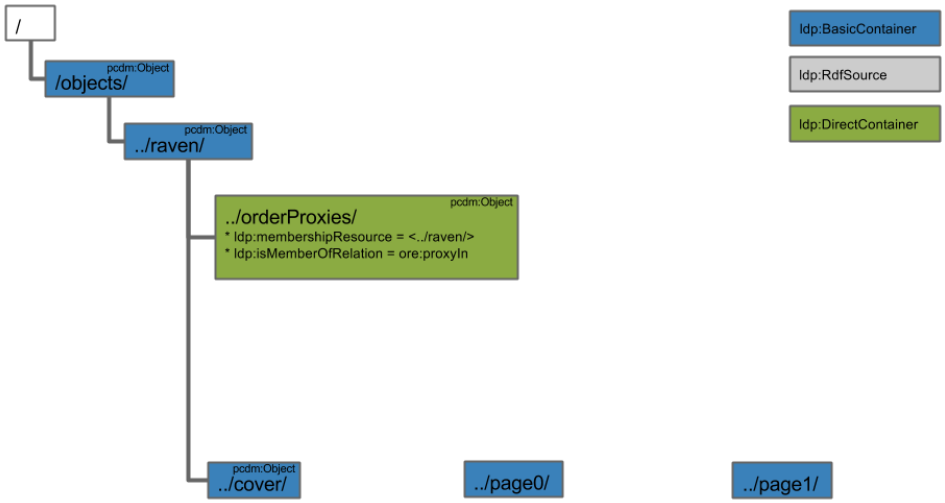
- Ordering - Create DirectContainer
- Ordering - Create Cover Proxy
- Ordering - Create Page0 Proxy
- Ordering - Create Page1 Proxy
- Ordering - Create Next and Prev
- Ordering - Create First and Last
- Ordering - Conclusion

Ordering - Create DirectContainer

Ordering - Create DirectContainer

As in the book example, begin with creating an ldp:DirectContainer, "orderProxies/", as a child of the book, "raven/", resource. This new DirectContainer will facilitate the auto-creation of triples that will define the membership relationship between the book, "raven/", and the proxies. Then, the new proxy resources within this DirectContainer will be used to establish an ordering of the books pages.

Note: This example assumes the previous creation of "/objects/raven/" and the cover and page resources from the Book example in this series.




```
curl -i -XPUT -H"Content-Type: text/turtle" --data-binary @ldp-ordering-direct.ttl localhost:8080/fcrepo/rest
/objects/raven/orderProxies/
```

Where "ldp-ordering-direct.ttl" follows:

ldp-ordering-direct.ttl

```
@prefix ldp: <http://www.w3.org/ns/ldp#>
@prefix pcdm: <http://pcdm.org/models#>
@prefix ore: <http://www.openarchives.org/ore/terms/>

<> a ldp:DirectContainer, pcdm:Object ;
  ldp:membershipResource </fcrepo/rest/objects/raven/> ;
  ldp:isMemberOfRelation ore:proxyIn .
```

An ldp:DirectContaner is an LDP construct that activates the creation of certain RDF triples when a new resource is added as a child of this container.

Like the "pages/" DirectContainer in an earlier example, the "orderProxies/" includes the ldp:membershipResource property ("raven/"). However, it is important to point out that the "orderProxies/" DirectContainer *does not* have the ldp:hasMemberRelation property defined, but instead uses ldp:isMemberOfRelation of "ore:proxyIn".
By using ldp:isMemberOfRelation, the auto-created triple resulting from the addition of a new child resource within "orderProxies/" will take the form:

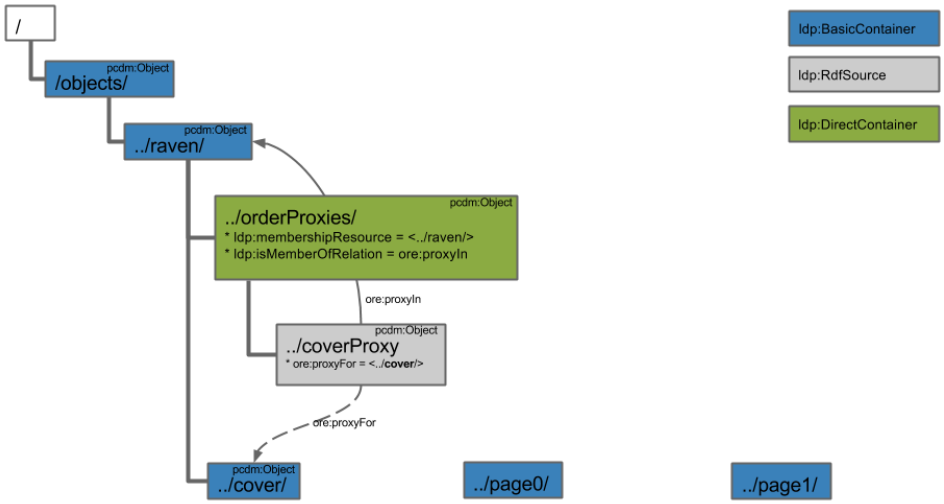
```
<new-resource> <ore:proxyIn> <http://localhost:8080/fcrepo/rest/objects/raven/>
```

We will see this in action next!

Ordering - Create Cover Proxy

Ordering - Create Cover Proxy

Create a new pcdm:Object, "coverProxy/", that is also an ldp:RdfSource within the "orderProxies/" DirectContainer.




```
curl -i -XPUT -H"Content-Type: text/turtle" --data-binary @ldp-cover-proxy.ttl localhost:8080/fcrepo/rest
/objects/raven/orderProxies/coverProxy
```

Where "ldp-cover-proxy.ttl" follows:

ldp-cover-proxy.ttl

```
@prefix pcdm: <http://pcdm.org/models#>
@prefix ore: <http://www.openarchives.org/ore/terms/>

<> a pcdm:Object ;
  ore:proxyFor </fcrepo/rest/objects/raven/pages/cover> .
```

As described in the previous step, the addition of "coverProxy" automatically creates the following new triple on "coverProxy"

```
<http://localhost:8080/fcrepo/rest/objects/raven/orderProxies/coverProxy> ore:proxyIn <http://localhost:8080
/fcrepo/rest/objects/raven>
```

Restating from the previous step,

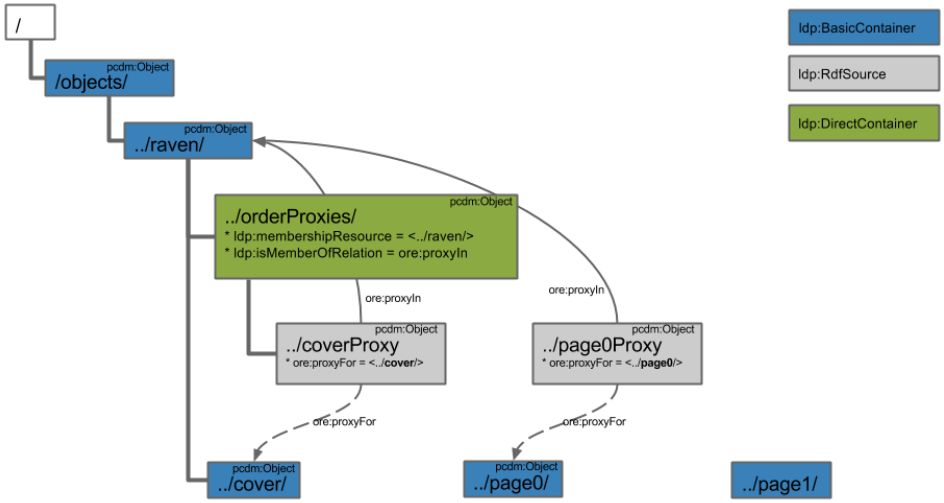
- the subject of the triple is the new resource ("coverProxy") that was added to the ldp:DirectContainer ("orderProxies/")
- the predicate of the triple comes from the "ldp:isMemberOfRelation" defined on "orderProxies/", and
- the object of the triple comes from the "ldp:membershipResource" defined on "orderProxies/"

Ordering - Create Page0 Proxy

Ordering - Create Page0 Proxy

In the same fashion as the previous step, adding "page0Proxy" to the DirectContainer, "orderProxies/" results in a new auto-generated triple on "page0Proxy" of the form:

```
<http://localhost:8080/fcrepo/rest/objects/raven/orderProxies/page0Proxy> ore:proxyIn <http://localhost:8080
/fcrepo/rest/objects/raven>
```




```
curl -i -XPUT -H"Content-Type: text/turtle" --data-binary @ldp-page0-proxy.ttl localhost:8080/fcrepo/rest
/objects/raven/orderProxies/page0Proxy
```

Where "ldp-page0-proxy.ttl" follows:

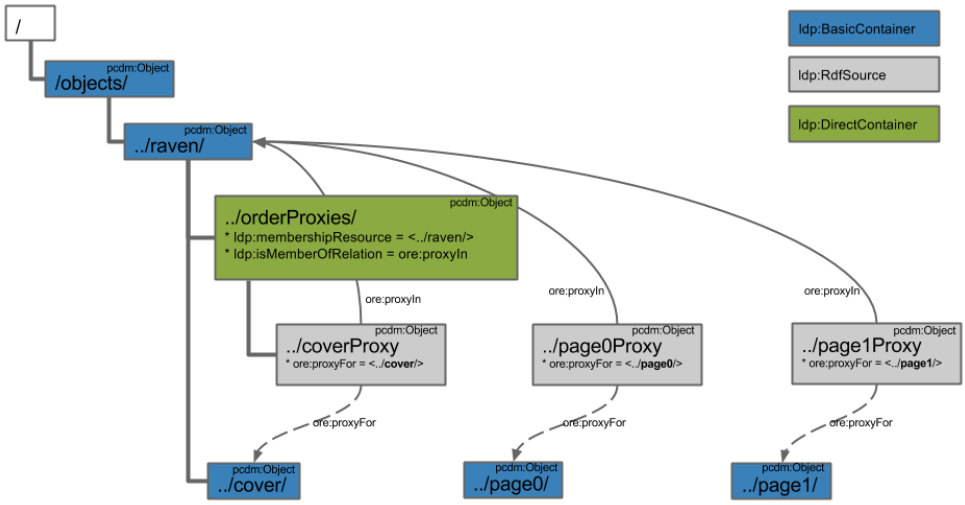
ldp-page0-proxy.ttl

```
@prefix pcdm: <http://pcdm.org/models#>
@prefix ore: <http://www.openarchives.org/ore/terms/>

<> a pcdm:Object ;
  ore:proxyFor </fcrepo/rest/objects/raven/pages/page0/> .
```

Ordering - Create Page1 Proxy

Ordering - Create Page1 Proxy

This step in creating the final page, "page1Proxy", follows the same pattern shown in the previous two steps.




```
curl -i -XPUT -H"Content-Type: text/turtle" --data-binary @ldp-page1-proxy.ttl localhost:8080/fcrepo/rest
/objects/raven/orderProxies/page1Proxy
```

Where "ldp-page1-proxy.ttl" follows:

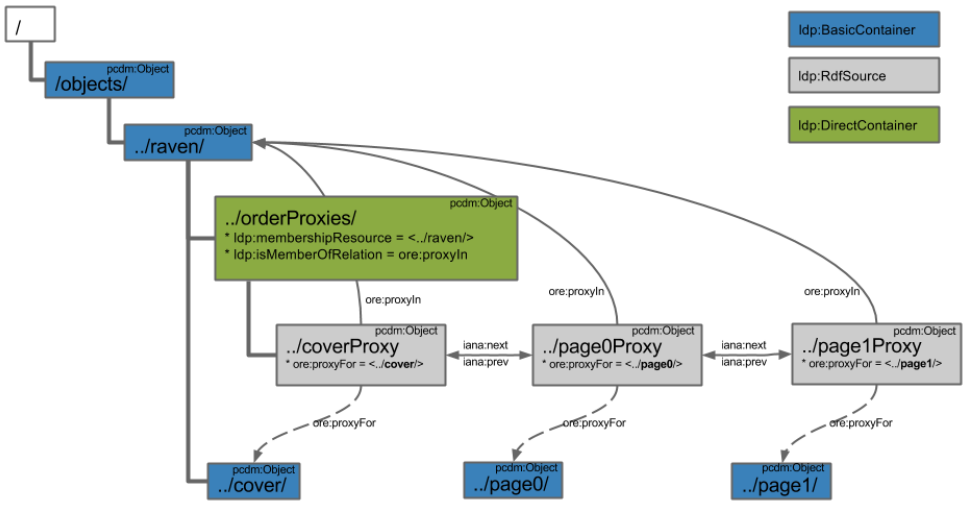
ldp-page1-proxy.ttl

```
@prefix pcdm: <http://pcdm.org/models#>
@prefix ore: <http://www.openarchives.org/ore/terms/>

<> a pcdm:Object ;
  ore:proxyFor </fcrepo/rest/objects/raven/pages/page1/> .
```

Ordering - Create Next and Prev

Ordering - Create Next and Prev

Now that the proxies have been created, and associated with the book ("raven/") and the proxied resources, we can actually use the proxies to establish ordering, per the PCDM recommendations.
First, establish the order among the proxies with iana:next and iana:prev.



1) Establish "coverProxy" has iana:next of "page0Proxy":

```
curl -i -XPATCH -H"Content-Type: application/sparql-update" --data-binary @iana-cover-proxy.ru localhost:8080
/fcrepo/rest/objects/raven/orderProxies/coverProxy
```

Where "iana-cover-proxy.ru" follows:

iana-cover-proxy.ru

```
PREFIX iana: <http://www.iana.org/assignments/relation/>

INSERT {
  <> iana:next </fcrepo/rest/objects/raven/orderProxies/page0Proxy>
} WHERE {
}
```

2) Establish both:

- "page0Proxy" has iana:prev of "coverProxy", and
- "page0Proxy" has iana:next of "page1Proxy":

```
curl -i -XPATCH -H"Content-Type: application/sparql-update" --data-binary @iana-page0-proxy.ru localhost:8080
/fcrepo/rest/objects/raven/orderProxies/page0Proxy
```

Where "iana-page0-proxy.ru" follows:

iana-page0-proxy.ru

```
PREFIX iana: <http://www.iana.org/assignments/relation/>

INSERT {
  <> iana:next </fcrepo/rest/objects/raven/orderProxies/page1Proxy> .
  <> iana:prev </fcrepo/rest/objects/raven/orderProxies/coverProxy>
} WHERE {
}
```

3) Establish "page1Proxy" has iana:prev of "page0Proxy":

```
curl -i -XPATCH -H"Content-Type: application/sparql-update" --data-binary @iana-page1-proxy.ru localhost:8080
/fcrepo/rest/objects/raven/orderProxies/page1Proxy
```

Where "iana-page1-proxy.ru" follows:

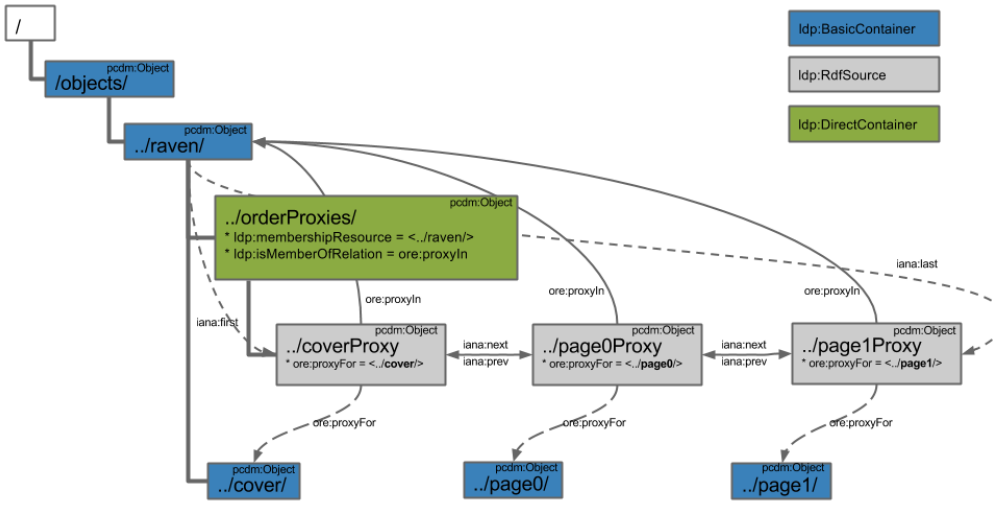
iana-page1-proxy.ru

```
PREFIX iana: <http://www.iana.org/assignments/relation/>

INSERT {
  <> iana:prev </fcrepo/rest/objects/raven/orderProxies/page0Proxy>
} WHERE {
}
```

Ordering - Create First and Last

Ordering - Create First and Last

Finally, the very last step is to define from the book's perspective, the iana:first and iana:last pages of "raven/".



Establish both:

- "raven/" has iana:first of "coverProxy", and
- "raven/" has iana:last of "page1Proxy":

```
curl -i -XPATCH -H"Content-Type: application/sparql-update" --data-binary @iana-raven.ru localhost:8080/fcrepo
/rest/objects/raven/
```

Where "iana-raven.ru" follows:

iana-raven.ru

```
PREFIX iana: <http://www.iana.org/assignments/relation/>

INSERT {
  <> iana:first </fcrepo/rest/objects/raven/orderProxies/coverProxy> .
  <> iana:last </fcrepo/rest/objects/raven/orderProxies/page1Proxy>
} WHERE {
}
```

Ordering - Conclusion

Using LDP in conjunction with PCDM terms, we have established the ordering of pages within the book, "raven/".