Versioning - Authorization Design

- General Information
  - Open Questions
  - Answered Questions:
- Delta Documentation
- Design
  - Players
  - RESTful Interactions
    - Enable Versioning on a LDPR
    - Check if a resource is versionable and discover the TimeMap/LDPCv
    - Check if the client can create versions
    - Creating a new version of a LDPRv
    - Access the TimeMap (LDPCv) to see what versions exist
    - Access an existing version (LDPRm)
    - Delete an existing version (LDPRm)
    - Restore an existing version (LDPRm)
  - Internal Interactions / Algorithms
    - Finding the ACL on a LDPRm (memento)
      - Memento and Security
      - Use Cases:
      - To find the ACL that relates to the LDPCv:
      - To find the ACL that relates to a LDPRm, follow this algorithm:
      - SOLID WebAC Specification:
  - Internal Representation of resources
    - LDPCv - Memento Container (TimeMap)
    - LDPRm - Memento
- Use Cases
  - Issues / Concerns:
    - URIs in LDPRm, and the "multiple parent" problem
    - Snapshot Versioning (problem ?)
  - Implementation thoughts
  - Design Implementation
  - REST Interaction questions

General Information

This is a place to record thoughts on the interaction of resource versions and WAC authorization in the context of the Fedora API alignment sprint.

JIRA issue: **FCREPO-2583** - Getting issue details... STATUS

Design Questions

Open Questions

1. What approach should be taken to resolve referential integrity issues causing changes to LDPRms?
    1. https://wiki.duraspace.org/x/TAJsBQ#VersioningDelta/SpecificationNotes-ref-int
    2. Also consider if referential integrity should be address across the repository rather than just for LDPRms
2. Is it acceptable to only create versions for individual resources rather than trees?
    1. https://wiki.duraspace.org/x/TAJsBQ#VersioningDelta/SpecificationNotes-version-single
3. How should restoring previous versions of LDPRvs work?
    1. https://github.com/fcrepo/fcrepo-specification/issues/217
4. How to make an existing LDPR versionable?
    1. PUT with an empty body may not be acceptable
5. Should it be possible to include non-LDPRm children in LDPCvs?
6. How can LDPCvs be identified?
    1. https://github.com/fcrepo/fcrepo-specification/issues/233
7. Do we need a separate memento ontology? How have people represented Memento information in RDF?
    1. What does the internal representation of mementos look like?

        1. Are mementos stored internally as separate resources LDP-RS contained by LDPCv (TimeMap resource) or could /should they be serialized versions stored as LDP-NRs? What other options would work to internally represent the mementos?
    2. Is it okay to store information in the LDPRm indicating it's archival time (and strip that out when delivering to user)?
8. For information about TimeMap and TimeGate can these resources not list them explicity and just use some known url formatting for this? We may not need the pointers to the TimeGate / TimeMap inside the resources.
    1. Timegate is LDPRv URL
    2. TimeMap is LDPRv URL + "/fcr:versions"
    3. The code that retrieves the memento can construct the "original", "timemap" and "timegate" link headers from it's own URL – is that okay to do?

Answered Questions:

- Fedora Modeshape implementation will cleanup inbound references upon deletion of an object. How do we keep the mementos intact and immutable if something they reference disappears?
    - Handling Deletion of Referenced Resources / Mementos

One way to fix this is to update the system so that links to other resources are URIs and not node references

- Should Fedora Modeshape allow snapshot (tree) versioning? What modeshape currently does in that scenario does not create actual mementos for ldp:contained resources.

The initial version of this change will not allow snapshot(tree) versioning. The first pass at spec compliance will only include versioning one resource at a time.

- Should the mementos that have ldp containment triples reference the current resource in those triples or a memento of those resources? Some of it depends on how the memento was created (as a tree or solo resource).
    - i.e., should </a/fcr:versions/vnum> ldp:contains </a/b> or ldp:contains </a/b/fcr:versions/vnum>

The discussion was to have the ldp:containment triples reference the cannonical URL of the resources.

- When versioning a tree of resources, what happens if one of the resources in the tree does not have the version interaction model attached to it?

Given that we will not be doing snapshot (tree) versioning at this point, this should not be an issue to consider at the present time.

Delta Documentation

For versioning: Versioning Delta/Specification Notes

For authorization: TBD

Design

Before reading through this, it would be good to review the Fedora Specification Versioning Section as well as understand the Memento Terminology.

This design relates specifically to how versioning could be done in the Modeshape Implementation of Fedora 4

Players

- **LDPR** - original resource
- **LDPRv** - same as LDPR, but it has the versioning interaction model turned on (http://fedora.info/definitions/fcrepo#VersionedResource). This implies that it *is* a TimeGate and *has* a TimeMap.
- **TimeGate** - a resource which provides access to LDPRms belonging to a LDPRv via datetime negotiation. This *is* the LDPRv.
- **LDPCv** - a LDP container that contains the LDPRms associated with a LDPRv. Since the TimeMap response is generated from this information the LDPCv and TimeMap are generally considered to be the same, but the LDPCv may contain more information than what is delivered in a TimeMap response.
- **TimeMap** - a resource from which a list of URIs of Mementos of the Original Resource is available.
- **LDPRm** - a specific version/memento of a LDPR. The LDPRv is not a LDPRm itself.

RESTful Interactions

1. ## Enable Versioning on a LDPR
    1. A PUT or POST request to create an object will make a resource versionable if it includes header Link: rel="type" with type of http://fedora.info/definitions/fcrepo#VersionedResource
        1. A LDPR will be created as a LDPRv with the versioning type.
        2. A LDPCv will be created, from which a TimeMap can be generated.
        3. A LDPRm will be generated, contained by the LDPCv.
        4. Any subsequent responses from the LDPRv will include the appropriate memento links in the header: Timegate, Timemap
    2. A PUT request to an Existing LDPR will make a resource versionable if it includes header Link: rel="type" with type of http://fedora.info/definitions/fcrepo#VersionedResource
        1. The versioning type will be added to the LDPR, making it a LDPRv.
        2. A LDPCv will be created, from which a TimeMap can be generated.
        3. A LDPRm will be generated, contained by the LDPCv.
        4. Any subsequent responses from the LDPRv will include the appropriate memento links in the header: timegate, timemap
2. ## Check if a resource is versionable and discover the TimeMap/LDPCv
    1. A HEAD request on the LDPRv will return response with Link rel="type" http://fedora.info/definitions/fcrepo#VersionedResource which indicates versioning support and a 'Link rel="timemap"' points to the URL of the LDPCv/TimeMap.
3. ## Check if the client can create versions
    1. An OPTIONS request on LDPCv/TimeMap that contains an "`Allow: POST" header` indicates that versions can be created by a client.
4. ## Creating a new version of a LDPRv
    - **Note**: when creating a new version of the LDPRv, only the single resource itself will be versioned. There is no concept of "tree" snapshots anymore.
    1. A POST request to the LDPCv with an empty body and no "Memento-Datetime" header will cause a new memento of the LDPRv to be created with current date/time.
    2. A POST request to the LDPCv with header "Memento-Datetime" and no body will create a historic verision with ~~current state of the LDPRv~~ an empty resource with the specificed date/time.
    3. A POST request to the LDPCv with header "Memento-Datetime" and a body will create a historic version with the specified body and date/time.
    4. A POST request to the LDPCv with a body and no "Memento-Datetime" header to create a version with the specified body and the current datetime.
5. ## Access the TimeMap (LDPCv) to see what versions exist
    1. A GET request to the LDPCv with the "Accept: application/link-format" header will cause the TimeMap to be returned.
    2. A GET request to the LDPCv with no "Accept:" header, or one specificying an RDF format will result in the LDPCv being returned in rdf format.
    3. The response from the GET will include a "Vary-Post: Memento-Datetime" to indicate that a client can request a specific time be associated with a memento when it's created via a POST.
6. ## Access an existing version (LDPRm)
    1. A GET request to the TimeGate Resource (the LDPRv itself) with "Accept-Datetime" header specified will return a 302 response, with a 'Location' header providing the URI of the LDPRm associated with that datetime, or the closest one if there is not an exact match.
        - example header usage: "Accept-Datetime: Thu, 31 May 2007 20:35:00 GMT"
        - See: Datetime negotiation algorithm example for Accept-Datetime negotiation details.
        - We are currently planning to follow Memento Datetime negotiation pattern 1.1, see: section 4.1.1.
    2. A GET request to LDPRm/Memento (if the LDPRm/Memento has its own URI), will result in the memento being returned if it exists.
    3. Any response from the LDPRv will include link relation headers of type "timegate" (referring to the LDPRv), "original" (also referring to this LDPRv), and "timemap" (referencing the URI of the LDPCv).
7. ## Delete an existing version (LDPRm)
    1. A DELETE request to LDPRm/Memento will result in that memento being deleted.
8. ## Restore an existing version (LDPRm)
    - **Note**: This interaction still needs to be ironed out as this is currently under discussion in Spec Issue 217
    1. A PUT request to LDPRv/TimeGate with header (can't be `Content-Location`, but something like it) pointing to the LDPRm /Memento URI to indicate the version to restore

---

Internal Interactions / Algorithms

Finding the ACL on a LDPRm (memento)

Memento and Security

Memento has very little to say about security, mainly just that it is up to the server in terms of how access to previous versions work (most likely we want it to behave the same way it behaved at the point of the snapshot, but that is something that needs to be decided), and what memento headers to expose during authentication:

https://tools.ietf.org/html/rfc7089#section-7

There are three separate entities when looking at ACLs.

1. LDPRv - the original resource. Potentially has it's own ACL or has one via inheritance.
2. LDPCv - is a full LDPR in and of itself and should have it's own ACL because it's a means of discovery for finding information about mementos.
3. LDPRm - is a full LDPR as well, but the existing ACL that it references should maybe not be it's ACL anymore, so that an admin can further change azn to the mementos w/o affecting the original LDPR.

Use Cases:

1. user has access to the LDRPv but not the mementos, but can they see the TimeMap?
    1. This imples that the LDPRv has different ACL then the LDPRm's. The LDPCv would probably have a different one as well.
2. user has access to the LDPRv and the mementos - they can see everything
3. user has access to the mementos, but not the LDPRv. They can see the TimeMap and mementos.

To find the ACL that relates to the LDPCv:

1. First look at the LDPCv for the memento to see if that particular LDPCv has an ACL.
2. Otherwise follow the pattern specified by the SOLID WebAC specification for finding an ACL based on the original LDPRv, as outlined below.

To find the ACL that relates to a LDPRm, follow this algorithm:

1. First look at the LDPCv for the LDPRm to see if it has an access control triple for memento items associated with it ('memento: accessControl'). If so, stop there and honor that ACL as it will apply to all mementos the LDPCv contains.
2. Otherwise follow the pattern specified by the SOLID WebAC specification for finding an ACL based on the original LDPRv, as outlined below.

SOLID WebAC Specification:

This is a slightly modified version of how the SOLID WebAC recommends finding the ACL.

1. Use the original document's (LDPR/LDPRv) own ACL resource if it exists (in which case, stop here).
2. Otherwise, look for authorizations to inherit from the ACL of the (LDPRv) document's container. If those are found, stop here.
3. Failing that, check the LDPRv container's *parent* container to see if *that* has its own ACL file, and see if there are any permissions to inherit.
4. Failing that, move up the container hierarchy for the LDPRv until you find a container with an existing ACL file, which has some permissions to inherit.
   The *root container* of a user's account MUST have an ACL resource specified. (If all else fails, the search stops there.)
    1. For fedora, there is no root container for a user - but there is a default ACL applied to the server overall. Should this algorithm fail to find an ACL at the root of the LDPRv's tree, it shall default to this system wide default ACL.

Given this, the following is then true:

- LDPCv's can have an ACL applied to them. If the LDPCv does not have a memento, then it uses the same memento as the LDPRv.
- LDPRv and it's mementos can have different ACLs.

If we are preserving ACLs as part of a version rather than using the original resource's, then for resources without an assigned or inherited ACLs Fedora would need to record the Default ACL at the time of the snapshot somehow. 5.3 Inheritance and Default ACLs

- Similarly, an inherited ACL would also need to be recorded for the snapshot.

Internal Representation of resources

These are just some notes written down largely to think through things - there might be a better way of doing this type of work.

An alternate design might be to serialize each memento into a binary resource and have that contained by the LDPCv. The key thing is to figure out how you store the archival date of each memento and retrieve that info to produce the TimeMap. Should it be in the memento itself? In the LDPCv? If so, how is that stored?

Here's an example of a LDPRv - what signifies that it is a LDPRv is that a request on the LDPRv returns memento related 'Link' headers in the reponse (and possibly a VersionedResource (spec issue 233) header as well) . These 'Link' headers point to the TimeMap and TimeGate for this resource . The current behavior is that a 'hasVersions' triple is returned when a LDPR is requested.

LDPRv

```
$ curl http://localhost:8080/rest/xyz


HTTP/1.1 200 OK
Date: Mon, 18 Sep 2017 14:58:26 GMT
Link: <http://localhost:8080/rest/xyz>; rel="original timegate"
Link: <http://localhost:8080/rest/xyz/fcr:versions>; rel="timemap"; from="
Fri, 8 Sep 2017 21:35:19 GMT"; until="Mon, 23 Sep 2017 15:41:04 GMT";

@prefix premis:  <http://www.loc.gov/premis/rdf/v1#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix fedora:  <http://fedora.info/definitions/v4/repository#> .
@prefix ldp:  <http://www.w3.org/ns/ldp#> .

<http://localhost:8080/rest/xyz>
        rdf:type               fedora:Container ;
        rdf:type               fedora:Resource ;
        rdf:type               ldp:RDFSource ;
        rdf:type               ldp:Container ;
        fedora:lastModifiedBy  "bypassAdmin"^^<http://www.w3.org/2001
/XMLSchema#string> ;
        fedora:createdBy       "bypassAdmin"^^<http://www.w3.org/2001
/XMLSchema#string> ;
        fedora:lastModified    "2017-09-18T20:01:33.501Z"^^<http://www.w3.
org/2001/XMLSchema#dateTime> ;
        fedora:created         "2017-09-15T21:19:49.731Z"^^<http://www.w3.
org/2001/XMLSchema#dateTime> ;
        fedora:writable        "true"^^<http://www.w3.org/2001
/XMLSchema#boolean> ;
        fedora:hasParent       <http://localhost:8080/rest> ;
        ldp:contains           <http://localhost:8080/rest/xyz/abc> ;
```

LDPCv - Memento Container (TimeMap)

Below were just a few initial thoughts on what the internal structure of a LDPCv & LDPRm might look like.

But first, here's a sample TimeMap in 'application/link-format' MIME type straight from the Memento Spec:

```
 HTTP/1.1 200 OK
   Date: Thu, 21 Jan 2010 00:06:50 GMT
   Server: Apache
   Content-Length: 4883
   Content-Type: application/link-format
   Connection: close

    <http://a.example.org>;rel="original",
    <http://arxiv.example.net/timemap/http://a.example.org>
      ; rel="self";type="application/link-format"
      ; from="Tue, 20 Jun 2000 18:02:59 GMT"
      ; until="Wed, 09 Apr 2008 20:30:51 GMT",
    <http://arxiv.example.net/timegate/http://a.example.org>
      ; rel="timegate",
    <http://arxiv.example.net/web/20000620180259/http://a.example.org>
      ; rel="first memento";datetime="Tue, 20 Jun 2000 18:02:59 GMT"
      ; license="http://creativecommons.org/publicdomain/zero/1.0/",
    <http://arxiv.example.net/web/20091027204954/http://a.example.org>
       ; rel="last memento";datetime="Tue, 27 Oct 2009 20:49:54 GMT"
       ; license="http://creativecommons.org/publicdomain/zero/1.0/",
    <http://arxiv.example.net/web/20000621011731/http://a.example.org>
      ; rel="memento";datetime="Wed, 21 Jun 2000 01:17:31 GMT"
      ; license="http://creativecommons.org/publicdomain/zero/1.0/",
    <http://arxiv.example.net/web/20000621044156/http://a.example.org>
      ; rel="memento";datetime="Wed, 21 Jun 2000 04:41:56 GMT"
      ; license="http://creativecommons.org/publicdomain/zero/1.0/",
    ...
```

A possible LDPCv is below. One issue to work through is that the LDPCv might have an ACL that applies to it, and then perhaps there is a separate ALC that applies to all the mementos. How do you indicate that other ACL? Check out the section on the algorithm for finding an ACL. You may end up deciding on a simplier authorization setup.

Note that there is no memento ontology - we may want to make one up...?

TimeMap (LDPCv)

```
@prefix acl:  <http://www.w3.org/ns/auth/acl#> .
@prefix iana:  <http://www.iana.org/assignments/relation/> .
@prefix ldp:  <http://www.w3.org/ns/ldp#> .
@prefix memento:  <http://example.com/memento#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix time: <http://www.w3.org/2006/time#> .

</path/to/resource/xyz/fcr:versions> a ldp:Container ;
   acl:hasAccessControl </path/to/acls> ;    # this is for the LDPCv
itself, for the TimeMap retrieval
   prov:startedAtTime "2017-09-08T21:35:19Z"^^xsd:dateTime ;  # first
memento
   prov:endedAtTime   "2017-09-11T15:41:04Z"^^xsd:dateTime ;  # last
memento

   memento:hasAccessControl </path/to/acls> ;
```

```
   memento:hasOriginalResource  </path/to/orig/resource/xyz> ;  # how else
can we represent this? is this a given based on url?
   memento:hasTimeGate </path/to/orig/resource/xyz> ;           # how else
can we represent this? is this a given based on url?

   iana:first </path/to/resource/xyz/fcr:versions/12344> ;
   iana:last </path/to/resource/xyz/fcr:versions/12347> ;
   ldp:contains </path/to/resource/xyz/fcr:versions/12344>, </path/to
/resource/xyz/fcr:versions/12345>,
      </path/to/resource/xyz/fcr:versions/12347>, </path/to/resource/xyz
/fcr:versions/12346> .
```

LDPRm - Memento

The timemap needs to have the archival time in it - so that needs to be stored some where. One thought was to store that data in the memento (as a hidden property that the user doesn't know about). Not sure if that's a good idea or not. Also, should the memento returned contain information about the next / prev memento? Does it still qualify as a memento if the resource has that data in it?

Memento (LDPRm)

```
@prefix acl:  <http://www.w3.org/ns/auth/acl#> .
@prefix iana:  <http://www.iana.org/assignments/relation/> .
@prefix ldp:  <http://www.w3.org/ns/ldp#> .
@prefix memento:  <http://example.com/memento#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix time: <http://www.w3.org/2006/time#> .

</path/to/resource/xyz/fcr:versions/12345> a ldp:RDFSource , prov:
InstantaneousEvent;
    prov:atTime "2012-04-30T20:40:40"^^xsd:dateTime;
    memento:hasTimegate </path/to/orig/resource/xyz> ;    # how else can
we represent this? is this this a given based on url?
    memento:hasOriginalResource </path/to/orig/resource/xyz> ;   # how
else can we represent this? Is this a given based on url?
    iana:next </path/to/xyz/fcr:versions/12346> ;   # to memento
    iana:prev </path/to/xyz/fcr:versions/12344> ;   # to memento

   ... triples from original resource at the time of versioning...

 or, if one decided to put the mememento one more layer down to keep it
totally separate, it might look like this (as in this LDPRm really just
wraps the actual resource):

(I'm not clear on how a binary and it's metadata would be represented)
    ldp:contains </path/to/xyz/fcr:versions/12345/version> ,
        </path/to/xyz/fcr:versions/12345/version/fcr:metadata> ;
```

Use Cases

Versioning/Authorization Use-Cases

Issues / Concerns:

URIs in LDPRm, and the "multiple parent" problem

It seems to be difficult to determine the identity of the "parent" of a resource via ldp:contains with versioning.

1. **LDPR contains LDPR** (without versioning). This one is simple. A ldp:contains A/B. A/B has exactly one parent: A.
2. **LDPRv contains LDPRv**. This one is also pretty simple. A ldp:contains A/B, A/B has exactly one parent: A.
3. **LDPRm contains LDPRv**. This one is a little less straightforward. There could be multiple resources that assert they contain B: What does a client need to know the identity of the resource that contains B?
    1. A_m1 (an LDPRm for A): <A_m1> <ldp:contains> <B>
    2. A_m2 (a different LDPRm for A): <A_m2> <ldp:contaiins> <B>
    3. A (an LDPRv): <A> <ldp:contains> <B>
4. **LDPRm contains LDPRm**. This one causes problems due to the fact that an LDPRm MUST be contained by an LDPCv as well. For memento B_m1 of resource B What does a client need to know in order to determine which one is B_m1's parent via containment?:
    1. A_m1 (an LDPRm for A): <A_m1> <ldp:contains> <B_m1>
    2. A_m2 (a different LDPRm for A): <A_m2> <ldp:contaiins> <B_m1>
    3. B_cv (The LDPCv for B): <B_cv> <ldp:contains> <B_m1>

Snapshot Versioning (problem ?)

The current fedora implementation creates (and links to) new LDPRs when a non-empty LDPRv is versioned. These resources are neither an LDPRv, nor LDPRm

For example, consider a container A and A/B where A ldp:contains B.

Creating a version v1 of <A> creates a resource <A/fcr:versions/v1>. This is essentially an LDPRm, and contains triple <A/fcr:versions/v1> ldp:contains <A/fcr:versions/v1/B>.

Issues are:

1. <A/fcr:versions/v1/B> is a new LDPR, bit its relationship to versioning is not defined in any spec. It is not an LDPRv, nor an LDPRm. It does not the have equivalent of an LDPCv associated with it.
2. There is no explicit or implicit relationship between <A/fcr:versions/v1/B> and <B>

All of this may be fine, but it lies outside of any specification. Essentially, "when you create a new version of a resource, that resource versions now points new and different things that it didn't previously point to, and have nothing to do with versioning"

Implementation thoughts

1. LDPRv – has child LDPCv – has children LDPRm(s)
   as long as we do not generate ldp:contains triples on the LDPRv for the LDPCv using this Predicate. We can ensure that deleting a LDPRv will remove the LDPCv and deleting the LDPCv with delete all the LDPRm(s).
2. How do we do the above filtering?
    1. Do we add new RDF types (fcrepo-specification issue 233)?
    2. Do we just add a property to the JCR node?
3. A GET or HEAD request to a timemap (LDPCv) should return all LDRm(s) and their start-end ranges.
    1. Do we want to generate this list on the fly or store it on the LDPCv and update it on any create/update or delete?
    2. The Memento-Datetime provided when creating a LDPRm (or the current date) indicates the start datetime of that version.
        1. would the end datetime be the next update to the LDPRv?
        2. would it be the lastModified date of the LDPRv?
4. Datetime negotiation of the LDPRv (URI-R) meaning a request to the LDPRv with an Accept-Datetime header can return either
    1. a 200 OK status and a Content-Location header with the address of the LDPRm (URI-M), if this is a GET request the LDPRm body should be returned too.
    2. a 302 Found status and a Location header with the address of the LDPRm (URI-M), the client must make a direct GET request to the provided Location URI.

3. which of the above two is better. A provides the body in one request (a GET to the LDPRv), but B does not use Content-Location that we already use for external content.

Design Implementation

If some of this is crazy or overkill, please don't do it. I'm finding it harder to create interface and classes than I expected without actually writing any code.

- In fcrepo-kernel-api and fcrepo-kernel-modeshape
  - Repurpose isVersioned(), enableVersioning() and disableVersioning() in FedoraResource interface, add getTimeMap() function this is our LDPRv.
  - Create TimeMap interace defining getVersions(), getVersion(final Date/Instant d), createVersion(), createVersion(final Date /Instant d), getOriginal() and Modeshape implementing class (LDPCv)
    - VersionService interface and VersionServiceImpl functions/code (if useful) can be moved to TimeMap class. createVersion() for instance.
  - Repurpose FedoraVersion interface and FedoraVersionImpl, perhaps need getOriginal() and getTimemap() for them (LDPRm)
- In fcrepo-http-api
  - Repurpose FedoraVersioning paths for use with TimeMap
  - Repurpose FedoraVersions to allow direct access to a specific Memento via its URI. (/fcrepo/rest/foo/fcr:versions/1234-5678)
  - Add new handing of headers and setting of response headers in FedoraLdp (**FCREPO-2612** - Getting issue details... STATUS)
  - Handle DateTime negotiation on LDPRvs in FedoraLdp (**FCREPO-2613** - Getting issue details... STATUS)

REST Interaction questions

(POST w/o body) LDPCv (copy) LDPRv LDPRm

1. POST to LDPCv without body and with or without Memento-Datetime header
2. (If WebAC enabled)
   1. Locate ACL on LDPCv, and up tree.
   2. Verify Read/Write permissions.
   3. Throw Exception if not permitted
3. Get LDPRv and create new child of LDPCv with this body.
4. Add rdf:type or identifier of Memento type (Identify as a Memento)
   1. If we use a rdf:type to mark a Memento then should we display that when the Memento is requested?
   2. Should we just call Mementos anything inside of a TimeMap?
5. Add snapshot datetime as property
   1. Where do we store the snapshot datetime? Property on the node?

(POST w body) LDPCv (compare LDP subtype) LDPRv LDPRm

1. POST to LDPCv with a new body and with or without Memento-Datetime header
   1. Can you provide a body and NOT provide a date?
2. (If WebAC enabled)
   1. Locate ACL on LDPCv, and up tree.
   2. Verify Read/Write permissions.
   3. Throw Exception if not permitted
3. Compare the rdf:type of resource because we can't provide a LDP-NR as a Memento of a LDP-RS or vice versa
4. Set the entity body as the new resource.
5. Identify as a Memento?
6. Add snapshot datetime as property.

(GET/HEAD w Accept: application/link-format) LDPCv

1. GET/HEAD to LDPCv
2. (If WebAC enabled)
   1. Locate ACL on LDPCv, and up tree.
   2. Verify Read permissions.
   3. Throw Exception if not permitted
   4. Locate ACL on ??? for LDPRm(s)
      1. where do we put the ACL for all the Mementos?
   5. Verify Read permissions.
   6. Throw Exception if not permitted.
3. Get all children Mementos, generate list of Mementos and return

(GET/HEAD w Accept-Datetime: header) LDPRv

1. GET/HEAD to LDPRv
2. (If WebAC enabled)
    1. Locate ACL on LDPRv and up tree
    2. Verify Read permissions.
    3. Throw Exception if not permitted.
    4. Locate ACL on LDPCv and up tree
    5. Verify Read permissions.
    6. Throw Exception if not permitted.
    7. Locate ACL on LDPRm(s)
    8. Verify Read permissions.
    9. Throw Exception if not permitted.
3. Get list of all LDPRm(s) and snapshot datetimes.
4. Find closest LDPRm(s)
5. Return 302 Found and Location: <path to chosen LDPRm>

General interaction questions.

1. How do you set the ACL that covers all Mementos?
    1. Can't POST/PUT to LDPRm(s) as Mementos are immutable
2.