DataONE API - Task #8528

Add MNRead.getVersions, CNRead.getVersions

2018-03-29 22:18 - Peter Slaughter

Status:	New	**Start date:**	2018-03-29
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
Story Points:			

Description

These new services would return the obsolescence chain, as a lists of PIDs, for the specified PID.
Both services (for CN and MN) would have the same argument list, i.e. for MNREAD.getVersions:

MNRead.getVersions(Identifier pid, Integer count, String order)

where the arguments are
- pid: the identifier to retrieve the obs. chain for
- count: specifies how much of the lists to return
- if count is null return the whole list
- if count is a positive number, return count pids in the specified order of, say, ASC or DESC
- order
- if order is DESC, return count from the HEAD, but from the TAIL if ASC.

Should the presence of multiple SIDs assigned in the obsolescence chain determine the lists of PIDs that is returned? For example, should the entire chain be returned even if multiple SIDs are encountered. The document at https://releases.dataone.org/online/api-documentation-v2.0/design/ContentMutability.html#limits-on-the-series indicates that multiple SIDs can be defined in an obs.chain.

Should PIDs be included in the list if the requesting user does not have permission to read the sysmeta for that pid? If no, then how is this gap in the chain denoted?

Related issues:

Related to DataONE API - Task #8529: Add field to Solr index that includes th...	**New**	**2018-03-29**

History

#1 - 2018-03-29 22:52 - Chris Jones

Another couple of things to note:

- The getVersions() API method would return a list of identifiers, but we need to determine the format of that list, and it should be consistent with the DataONE Types. For instance, listObjects() returns an ObjectInfoList which is a collection of ObjectInfo objects. It would be nice if we had a Types.IdentifierList that is just a list of Identifier objects, but we currently don't, and adding that to the Types schema would affect the whole DataONE software stack. So we need to discuss what the return type should be.
- We should discuss the order argument values. DESC and ASC might be better expressed as newest (meaning newest to oldest) and oldest (meaning oldest to newest. Thoughts welcome.
- We would need to define the default order when that argument is null (which I would suggest it being DESC or newest)
- If count is null, return the whole list, but in the case of negative numbers, we may want to throw an exception, so we need to define the exceptions appropriate for this API method.

#2 - 2018-03-29 23:27 - Matthew Jones

Overall this looks good. However, I would argue the count parameter is unnecessary and just complicates things. So I would omit it. The order parameter is also likely unneeded, as its simple for clients to look at the head or tail of the list given that they have to parse it anyways. So the whole thing could be simplified to just a PID parameter.

I don't think SIDS should have any impact on the results. The results are based on the obsolescence chain of the underlying PIDS.

#3 - 2018-03-30 03:50 - Rob Nahf

I think the naive “follow-the-chain” implementation is going to be a bit expensive, with n-1 self-joins on the postgres systemMetadata table. I think it better to define a common chain identifier all members of the obsoletes chain share, so that it’s a two-step lookup: (1) from leaf to root (the chain identifier), then (2) from root to all leaves. It could probably be done as a couple extra columns in the metacat identifier table, or as a dedicated cross-reference table. Especially if this is a call being done automatically from the search UI, it seems like there will be a lot of API calls, so efficiency will matter. It would increase the cost of creates and updates, but would save a lot of cycles in retrieval.

#4 - 2018-03-30 04:09 - Rob Nahf

by the way, use of a chain identifier would be quite analogous to how series IDs work, (composite structure), but chain identifiers would be automatically assigned an likely invisible to users.

While chain identifiers might be distinct from series Ids, in the case of mutable member nodes, obsolete chains may be incomplete and the series Id might help glue the pieces back together, at least on the CN, where there could be discontiguous(?) synchronization.

So in some cases (on the CN), there are multiple series in the obsoletes chain, and in other cases, there will be multiple obsoleted chains in the series, due to discontinuities.

#5 - 2018-03-30 04:11 - Rob Nahf

- Related to Task #8529: Add field to Solr index that includes the obsolescence chain added

#6 - 2018-03-30 19:27 - Chris Jones

To be consistent with other API calls, we should consider calling this listVersions() instead of getVersions(). The latter just popped into my head when I suggested it. Let's discuss.

#7 - 2018-04-03 19:26 - Jing Tao

Here is my thought about the implementation:
I believe the given identifier can be a sid or pid

```
sysMeta = hazelCast.get(id)
if (sysmeta != null ) {
    //it is a pid
    sid = sysmeta.getSID();
    if (sid == null) {
      return the id; //this is a pid without a sid. Just return itself.
     }
} else {
     sid = id; //asume the given id is sid.

}
//then run this query; the order can be modified base on the given value
select guid, obsoleted_by, obsoletes from systemMetadata where series_id = ? order by date_uploaded DESC
```

We can use a data structure to store the information and sort the list. Since the chain should NOT be very long, so the performance will not be an issue.