# DataONE Architecture

## DataONE Use Cases and APIs

**Change Log**

**20090927:**
- Split usecases into separate document
- Added page breaks for rst2pdf

**20090925:**
- Initial collation of all use cases into single document.
- Recast all diagrams to interaction diagrams using PlantUML automated tools

**20090924:**
- Layout document structure, draft templates

# Table of Contents



# 1 Use Cases Overview

**Date:** 20090924

An important early part in the process of system architecture is identifying the functional requirements that the resulting system should be able to address. Use cases are helpful for this as they describe how actors (typically entities external to the system) interact with system components. Interaction diagrams help define the interfaces and messages transmitted between components.

***Note***

**TODO:** Architecture overview diagram
**TODO:** Component diagram
**TODO:** Broad categories of use cases

## 1.1 Use Case Layout

Each use case document is structured following a common template which includes the following elements.

**Author**
Who generated the use case. This will usually be "VDC TWG".

**Date**
Date the use case was identified and known subsequent modifications.

**Version**
One of "Draft", "Review", "Published"

**Goal**
Brief description of what is intended with the use case.

**Summary**
An overview of the use case which includes the goal and major actors.

**Use case diagram**
A diagram representation of the use case, somewhat following the UML.

**Sequence diagram**
A sequence diagram that shows the operations involved in addressing the use case.

**Actors**
The participants in the use case - includes people and systems.

**Preconditions**
The conditions that must exist for the use case. For example, a particular use case may only be valid if the use has been authenticated.

**Triggers**
Describes the even that causes the use case to be initiated.

**Postconditions**
The state of the system after the use case - what conditions are true after the use case is completed.

**Notes**
Miscellaneous notes and discussion items that can't be placed in the various categories. Comments should be annotated with time and author.

# 2 Use Case Implementation Priority

Prioritizing system implementation to address the use cases involves several factors including:

- *Vision of the project*. The system is being designed with some overall goals described by the vision of the project proposal.
- *Requirements of the community*. The stakeholders that comprise the user and participant community quite likely has some opinion on functionalities of the system that are important to them. If these are not properly addressed, then the resulting system may appear as a failure to them.
- *Requirements of the sponsor*. The sponsor has laid out goals in the RFP that the project is responding to and also in the final agreement for conducting the work.
- *Dependencies between use cases*. Implementation of functionality to address some use cases will require implementation of some components not directly specific in a use case.
- *Resources available for implementation*. Some use cases may be identified as high priority, but would require resources that would prevent implementation of a number of other lower priority features.

**TODO:** Insert an initial prioritization of the use cases.

# 3 Use Cases

## 3.1 Use Case 1 - CRUD Read()

**Author**
VDC TWG

**Date**
- 20091006 Added use case diagram
- 20090924 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
Get object identified by GUID (authenticated or not, notify subscriber of access).

**Summary**
A client has an identifier for some object within the DataONE system and is retrieving the referenced object. The DataONE system must resolve the identifier and return the object bytes after checking that the user has read privileges on the object.
Watchers (subscribers) of the object are notified of the retrieval (what are they notified of?).

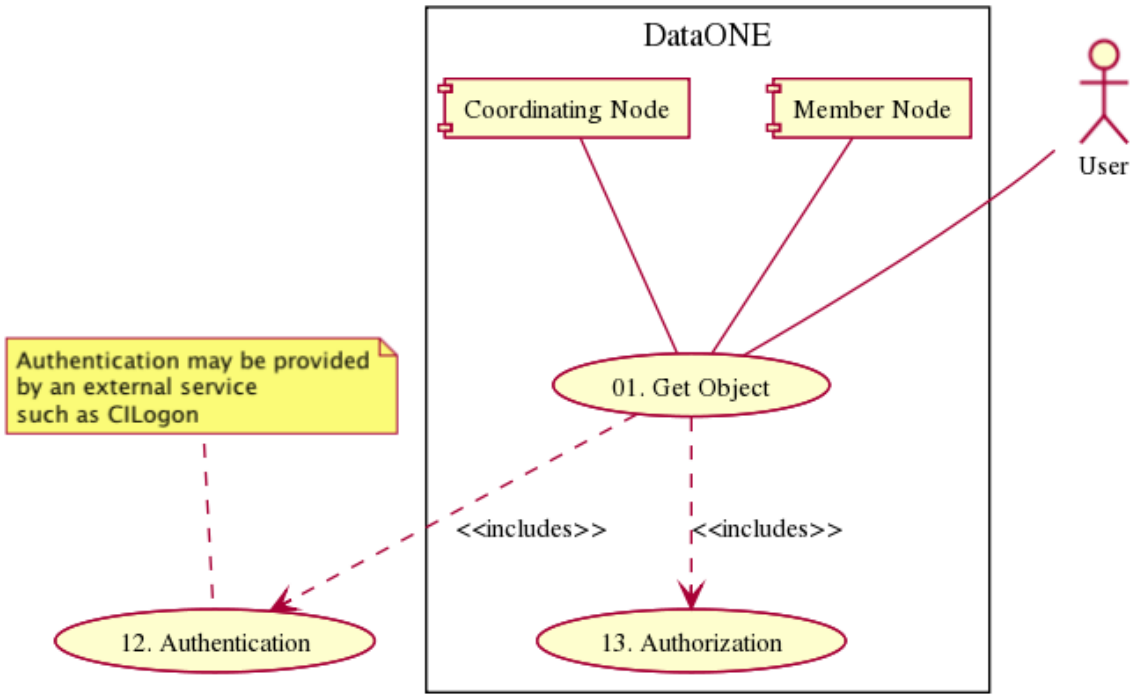
**Actors**
- Client requesting object
- Coordinating Node
- Member Node

**Preconditions**
- Client has authenticated to at the desired level (e.g. client may not have authenticated, so access might be anonymous), and so an `auth_token | None` is available.

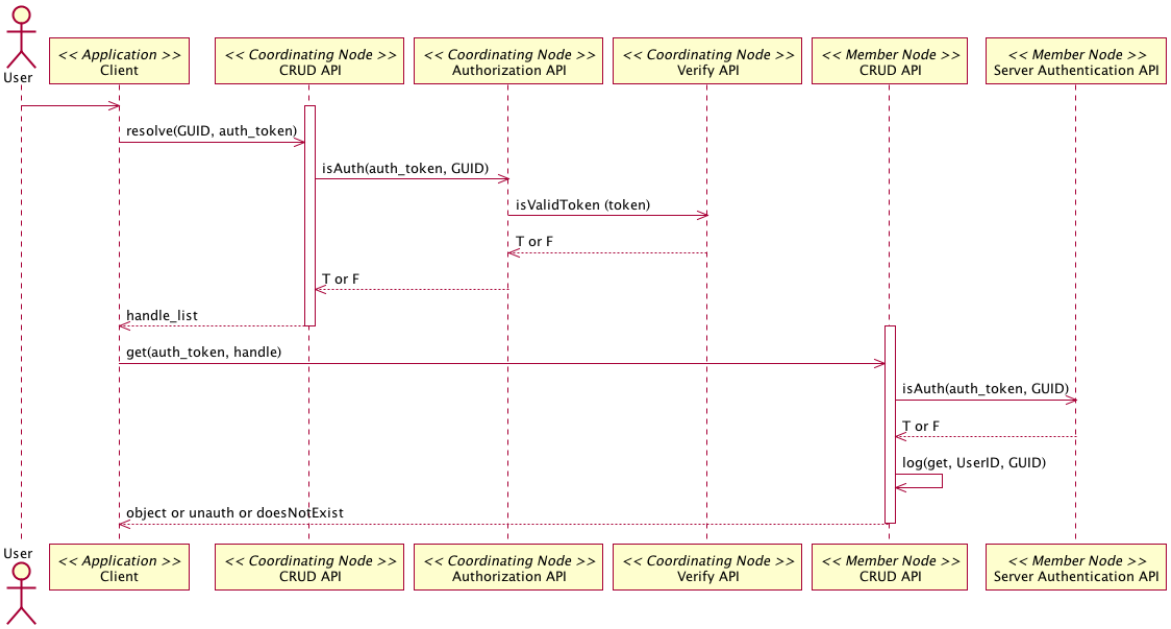
**Triggers**
- An object is requested from the DataONE system.

**Post Conditions**
- The client has a copy of the object bytes (or an error message in the case of failure)
- The log is updated with the results of the operation
- Watchers perform their processing

**Notes**
1. For the GET operation, the coordinating node does not need to check access, because the member node will do this check before delivering the results.
2. For the GET operation, should isAuth() be performed only by CNs? Relying on the MN system metadata requires trusted implementation of the MN system and consistency of system metadata across all MNs (which will be the case, though with uncertain latency). Requiring all isAuth() operations to be performed by CNs will increase trust in the operation (assuming the operation is not spoofed by a MN) though will increase load on CNs.
3. Data sent to watchers might include: timestamp, object identifier, user id, IP of client.




*Use case diagram showing actors and components involved in this action.*




*Sequence diagram for retrieving an object from DataONE.*

## 3.2 Use Case 02 - List GUIDs By Search

**Author**
VDC TWG

**Date**
- 20090927 Provided more detail to use case
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications

**Version**
Draft

**Goal**
Get list of GUIDs from metadata search (anon and authenticated).

**Summary**
A user performs a search against the DataONE system and receives a list of object identifiers (GUIDs) that match the search criteria. The list of GUIDs is filtered such that only objects for which the user has read permission will be returned.
This use case assumes that the search is being performed by submitting a query against a CN.

**Actors**
- Client performing search operation
- Coordinating Node

**Preconditions**
- Client has authenticated to at the desired level (e.g. client may not have authenticated, so access might be anonymous).

**Triggers**
- A search is performed against the DataONE system

**Post Conditions**
- The client has a list of GUIDs for which they have permission to read and match the supplied query, or an error message if the query was invalid.
- The log is updated with the request

**Notes**
- lots of stuff buried in query. What information/interface is needed for resolution/expansion of the query? For example a particular user may be identified with a particular community which uses a particular ontology. Where are things like ontologies maintained and where are the interfaces between these and searches maintained? Searches need to be able to respond to the dynamic nature of domain ontologies. Does the ontology need to have authorization tied to it? A use case would be an ontology which is under development and not ready for general distribution. Another case might be where the ontology contains representations of human subject information, meaning that "cancer survivors" includes this list of people. Or where the ontology provides geospatial information about restricted information (locations of USFS FIA plots). Similar issues apply to T&E species information.
- Group structure representation: A user is a member of zero to many groups, authorization may be carried with that group membership. This is down in the verify API, which needs to take into account group membership and group permission inheritance. Question: What is the resolution for conflicting access permissions?
- Q. Can a query be done at a member node? The diagram implies that the query is only done at a coordinating node. A member node may have a subset (or the full set) of metadata and the user may wish to do the search only at a member node, knowing that the subset is restricted to data relevant to a community of practice.

  A. First iteration will provide query service on CNs, though nothing to prevent MNs implementing query as well.

- Where is the filtering done for the results, in terms of when does the result set get filtered to those which the user can see?

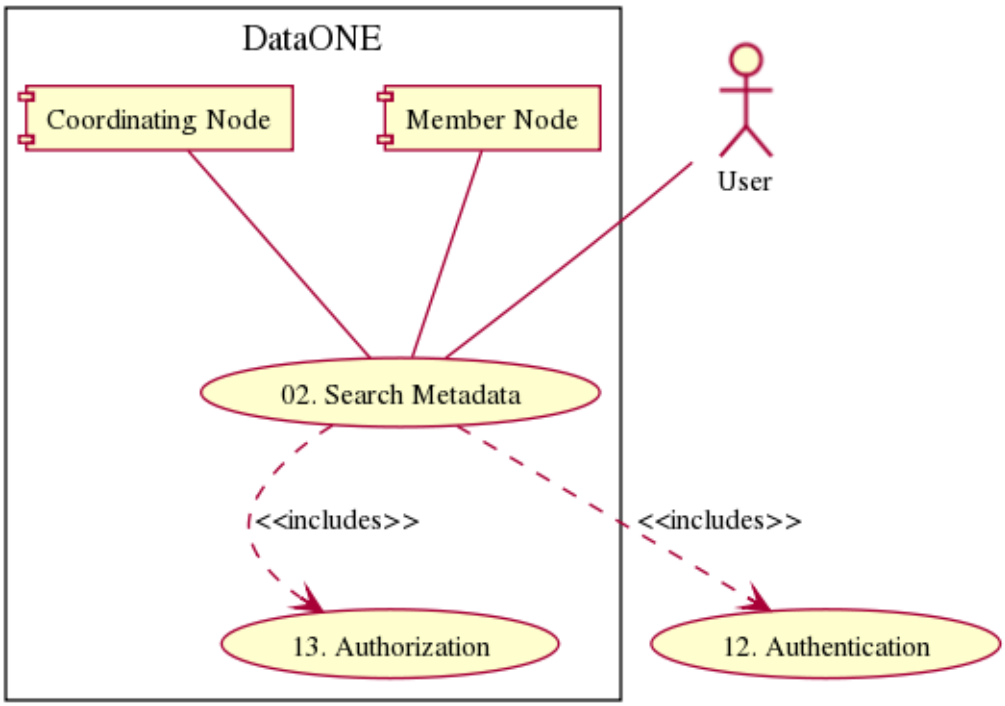
  Q. Does the model allow the query service to do two searches and union the results? The first query might be against the set of records known to be publicly readable, with a follow-on query against the set of records known to have restricted access. Is this envisioning the query service generating a set of results (against the restricted records) and passing that list to the authorization API, or is this envisioning a one by one situation. Or does the authorization API pass back to the query service the list of records that the user is authorized to see, and that gets anded with the result set.

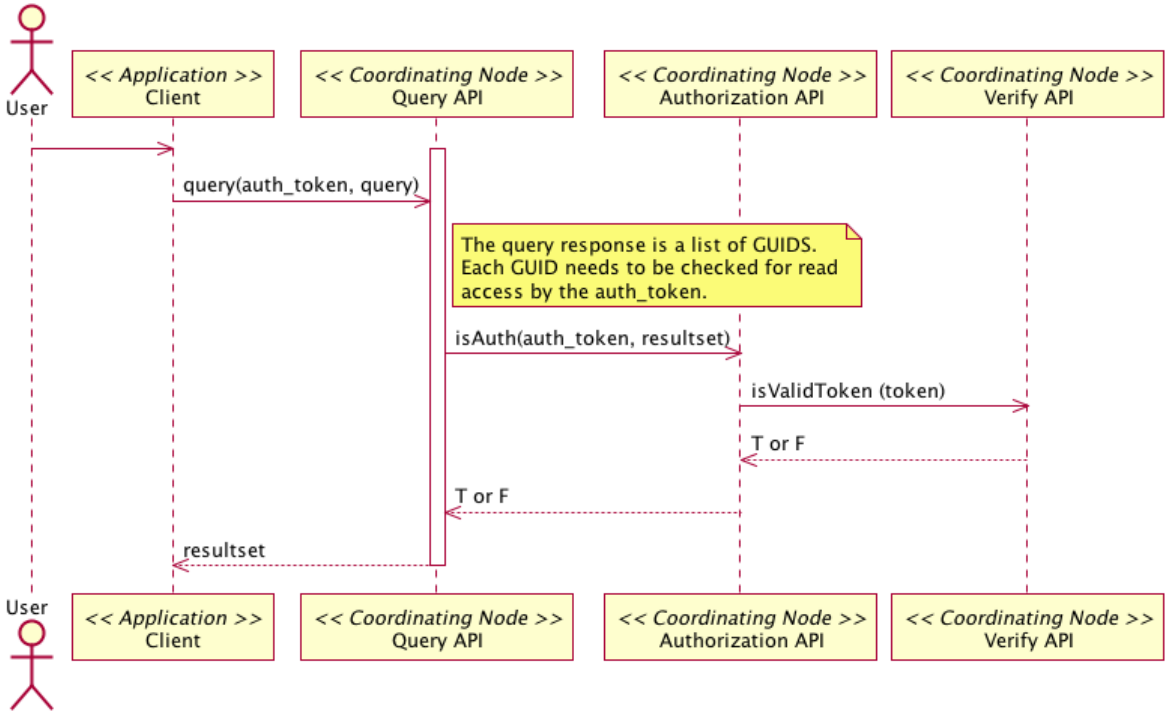
  A. The resultset is a list of GUIDs that needs to be vetted for read access. Hence query results can be merged (union) or joined as the output from such operations is always a list of GUIDs that will be vetted for read access.
- How distributed is the database of users and their authorization records?

  Q. Does the coordinating node have to know all authorization records, with that being replicated? Or is some amount of the authorization maintained by (for example) the "owning" member node and the search system may need to contact multiple authorization authorities for the records in a particular result set? It also implies that if the authorization information is at the coordinating node, that the authorization authority needs a means to validate the replicated authorization information to ensure that the info at the cn is up-to-date.

  A. One scenario is that all authorization information is stored in the system metadata with each object. Then authorization can be performed by MNs or CNs.
- Q. Why does step 4 need to validate the token, given that the token was generated in step 2?

  A. This is normal security validation of all user inputs once those have gone out of the control of the issuing service. Also applies to cases where tokens have a specific lifetime, and the token may be valid, but expired. Authorization token lifetimes are important for both interactive sessions and for cases like workflows, where the session may run for days.
- What logging level is required for the operation? Should the search appear as a hit against each object? Just log the query (and perhaps who did it)?




*Client retrieves a list of GUIDs resulting from a metadata search.*

## 3.3 Use Case 03 - Register MN

**Author**
VDC TWG

**Date**
- 20090927 More detail added to use case description
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
Register a new Member Node.

**Summary**
This use case describes the technical process for addition of a new member node (MN) to to the DataONE infrastructure. It is assumed that the appropriate social contracts have been formed and the MN is operational, ready to be connected.
The MN is identified by a URL which is the service endpoint. A DataONE administrator adds the MN URL to the MN registry. A CN retrieves the registration request from the message queue and queries the MN for capabilities, then verifies those capabilities match what was advertised. If everything is OK, the MN is made a live member of DataONE and replication of content on the MN is scheduled. The new MN also becomes a receiver for content replicated from other MNs.

**Actors**
MN, CN, Administrator

**Preconditions**
- The MN is operational
- There is an agreement between the DataONE operators and the MN operator that the MN is to be added to DataONE
- The CNs are ready for receiving MN registrations

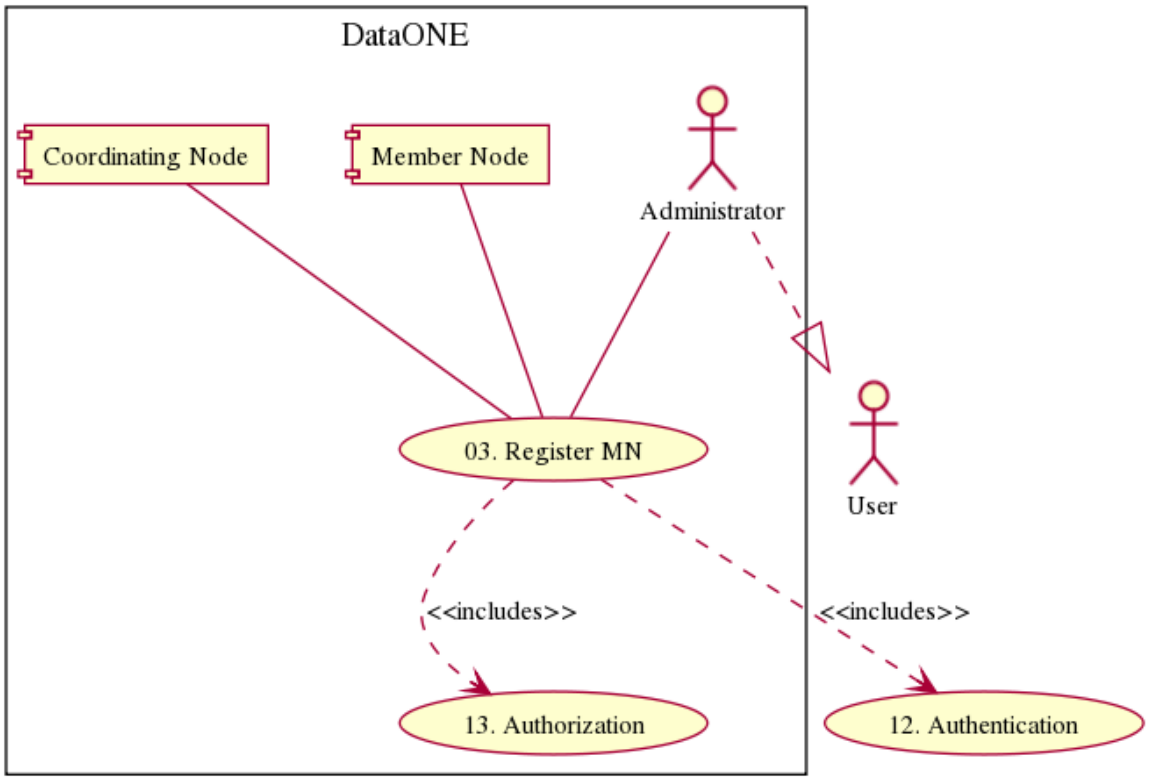
**Triggers**
- A new Member Node is ready to be brought online and a DataONE administrator initiates the process.

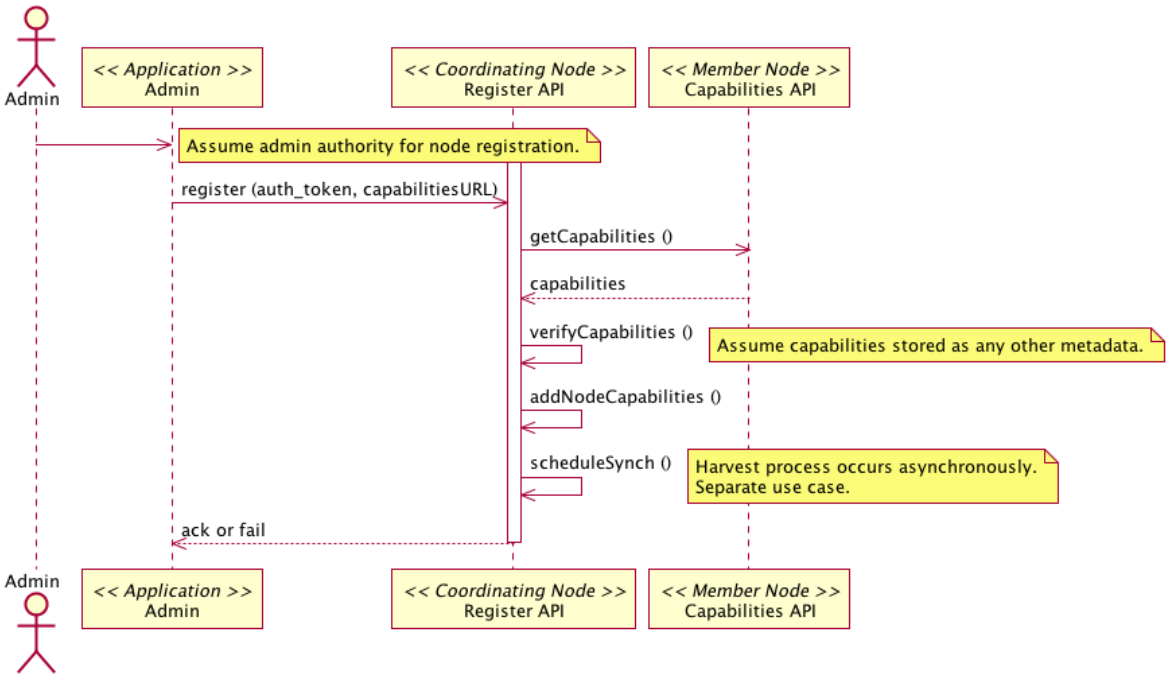
**Post Conditions**
- The new MN operates as part of the DataONE infrastructure
- DataONE infrastructure resources are incremented by the amount available at the new MN
- The operation outcomes are logged
- Synchronization of the MN commences as per scheduled operation
- Synchronize capabilities metadata across CNs
- Notify MN administrators of operation status (email?)
- The new MN is added to CN list of targets for replication

**Notes**
- Specify default replication policies
- Also check for version updates
- Should new nodes be registered with specified trust levels?
- Are there different levels of trust for member nodes?
- Data providers that use acceptable services can still be discovered and accessed without registering in the DataONE registry (Conflicting, as someone has to register it).

  However, a MN that has not registered but does expose DataONE services is not part of the DataONE infrastructure and so does not participate in replication or other DataONE services.
- Allow service providers to register their services, (such as data extraction services) (ala GEOSS), but include mapping to higher semantic model

- For well known services, registration system must be able to describe constraints (e.g., allowable inputs, outputs, algorithms that can be specified)
- Services can be parameterized
- Member node should be able to request the coordinating node to re-validate its capabilities list.


DataONE
Coordinating Node
Member Node
Administrator
User
03. Register MN
<<includes>>
<<includes>>
13. Authorization
12. Authentication


Admin
<< Application >>
Admin
<< Coordinating Node >>
Register API
<< Member Node >>
Capabilities API
Assume admin authority for node registration.
register (auth_token, capabilitiesURL)
getCapabilities ()
capabilities
verifyCapabilities ()
Assume capabilities stored as any other metadata.
addNodeCapabilities ()
scheduleSynch ()
Harvest process occurs asynchronously.
Separate use case.
ack or fail
Admin
<< Application >>
Admin
<< Coordinating Node >>
Register API
<< Member Node >>
Capabilities API

## 3.4 Use Case 04 - CRUD (Create, Update Delete)

**Author**
VDC TWG

**Date**
- 20090929 Filling out details
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
Create, update or delete metadata record on a Member Node.

**Summary**
A user is creating a new metadata record on a Member Node (MN). The mechanism by which the user does this is out of scope for the DataONE system, so this use case continues from the point where a new Data Package is present on the MN.
The metadata is retrieved by the CN using a pull mechanism (CN requests content from the MN). The availability of new content on a MN is indicated by a flag in the response to a ping() message issued by a CN. The CN then retrieves a list of changes from the MN and for each change, retrieves the experiment metadata, stores it and generates a search metadata document by extracting content from the experiment metadata. The changes to the CN (stored experiment metadata, update search metadata) are replicated to the other CNs (Use Case ###). The CN schedules replication of the data package across other MNs (Use Case 06).

**Actors**
Member Node, additional Member Nodes, Coordinating Nodes

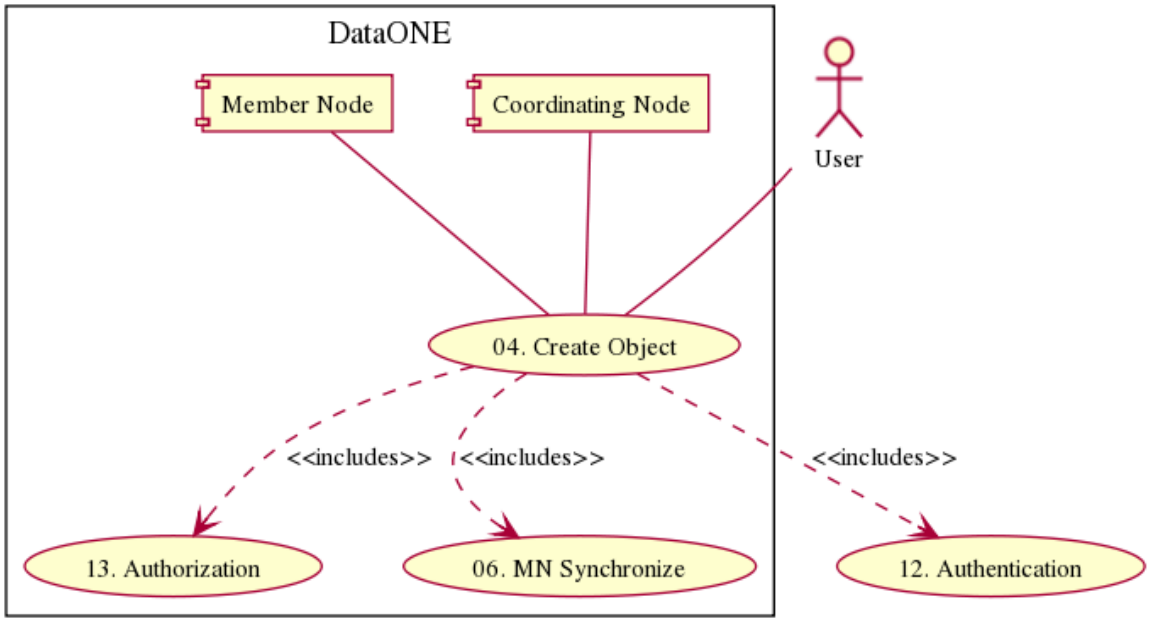
Preconditions
- Use is authenticated and so client application has an auth_token for the process.
- DataONE system operational
- MN is registered and is monitored

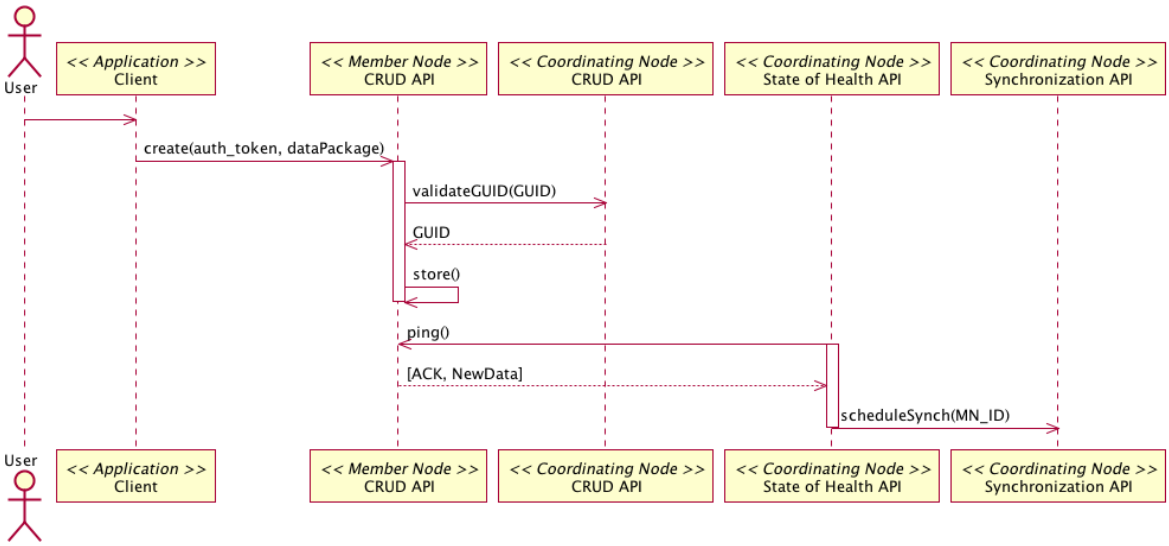
Triggers
- New data package is present on Member Node
- An existing data package is modified on a Member Node
- A data package is deleted from a Member Node

Post Conditions
- New data package is replicated across MNs
- Search metadata is updated
- Operation recorded in logs
- CN update of search metadata
- CN replication of metadata
- Watchers notified of change

### *3.4.1 Notes*

- Are IDs are pegged to versions? If so, update might differ more radically from create.
- If delete is a metadata operation only, then will be very similar to update. But if it is a physical operation, it becomes more complex. Will there be an archival copy? Will member nodes that hold replicas be "forced" to destroy the replicas?
- Is submit from a member node a push or pull?
- Should queuing of a submission within a member node be required for success? (We assume yes, to guarantee replication of submissions).
- Can data packages be deleted or modified?




*Create, update, delete, search metadata or data object in Member Node.*

## 3.5 Use Case 05 - CRUD (Create, Update Delete)

**Author**
VDC TWG

**Date**
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
Create/Update/Delete data object in Member Node.

**Summary**
This is a dupe of UC04 for now. May split out the update and delete portions to different use cases at some point in the future.

**Actors**
xxx

**Preconditions**
xxx

**Triggers**
xxx

**Post Conditions**
xxx

### *3.5.1 Notes*

- See all questions above (Use Case #4) for metadata operations.
- client should be able ask CN for a list of acceptable MNs to upload data with specific characteristics (e.g., size, institution, theme)

## 3.6 Use Case 06 - MN Synchronize

**Author**
VDC TWG

**Date**

- 20090930 Filled out content, changed mode to pull from MN
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
Replicate / synchronize metadata record between Member Node and Coordinating Node.

Summary

As data packages, or portions there of are created or modified, the metadata associated with those needs to be replicated to the coordinating nodes.

The presence of new or changed information on a Member Node (MN) is made known to a Coordinating Node (CN) through the status information in a `ping()` response. If so indicated, the CN schedules a synchronization operation with the MN, a list of changed object GUIDS is retrieved by the CN, and the CN proceeds to retrieve and process each object identified.

If new data packages are present on the MN, then a MN-MN synchronization process is scheduled.

**Actors**
Member Node, Coordinating Node

**Preconditions**

- Member Node has new or changed data packages

**Triggers**

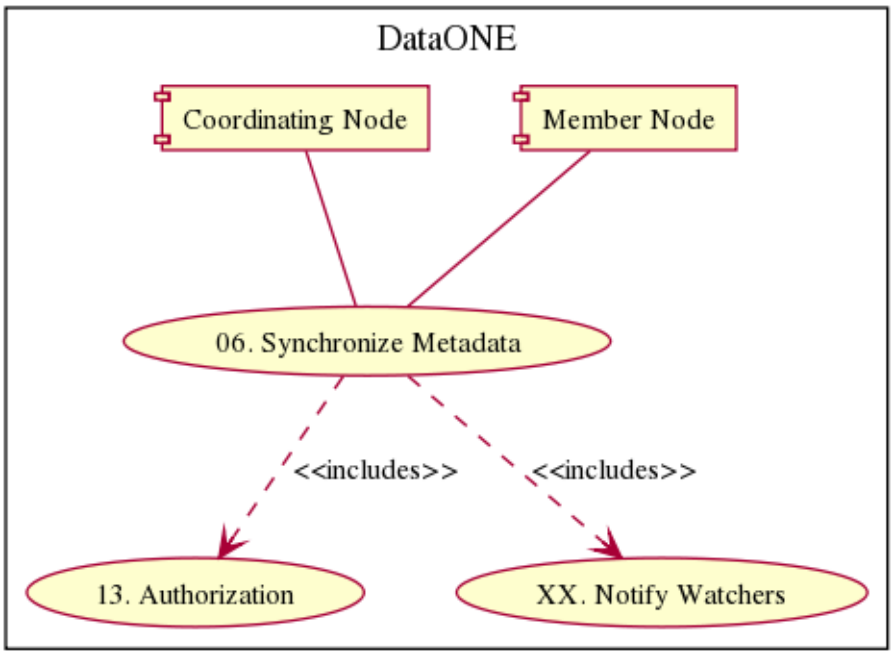
- New or changed content is present on a Member Node

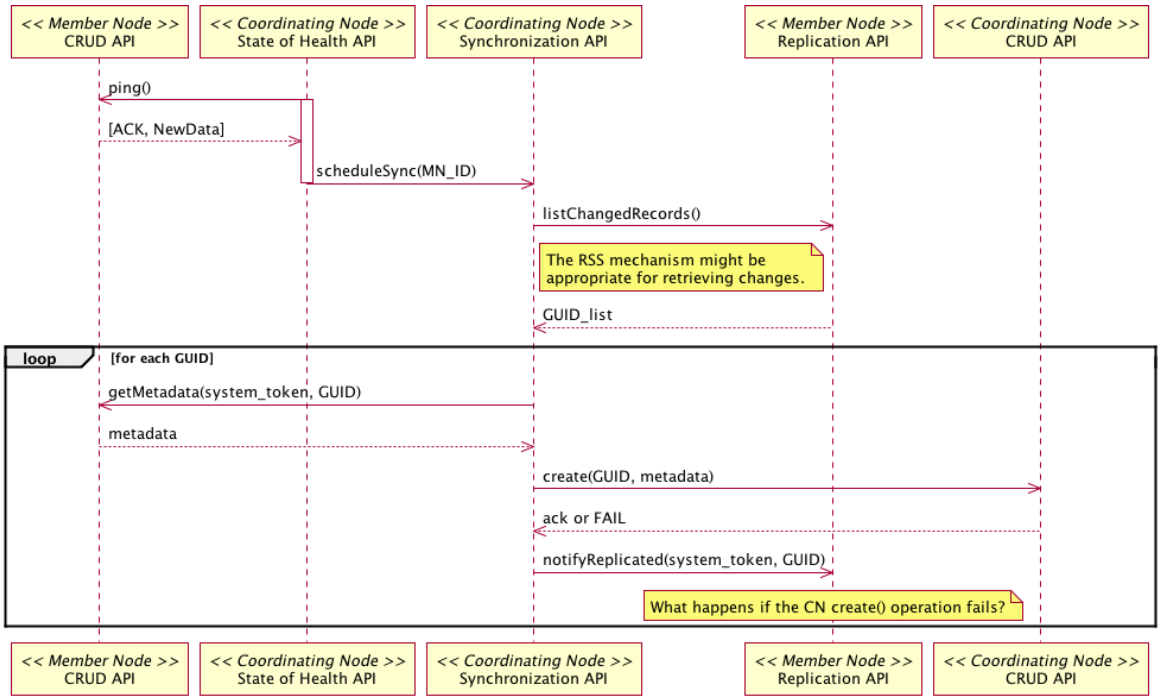
**Post Conditions**

- MN is in sync with CN
- Log
- Watchers for changes to MN, CN, or DataONE system

**Notes**

- Overloading ping() for notification of change seems a little bit of a hack, though changing the name from ping() to status() makes it seem OK.




*Synchronize Metadata between Member and Coordinating Node*

## 3.7 Use Case 07 - CN Batch Upload

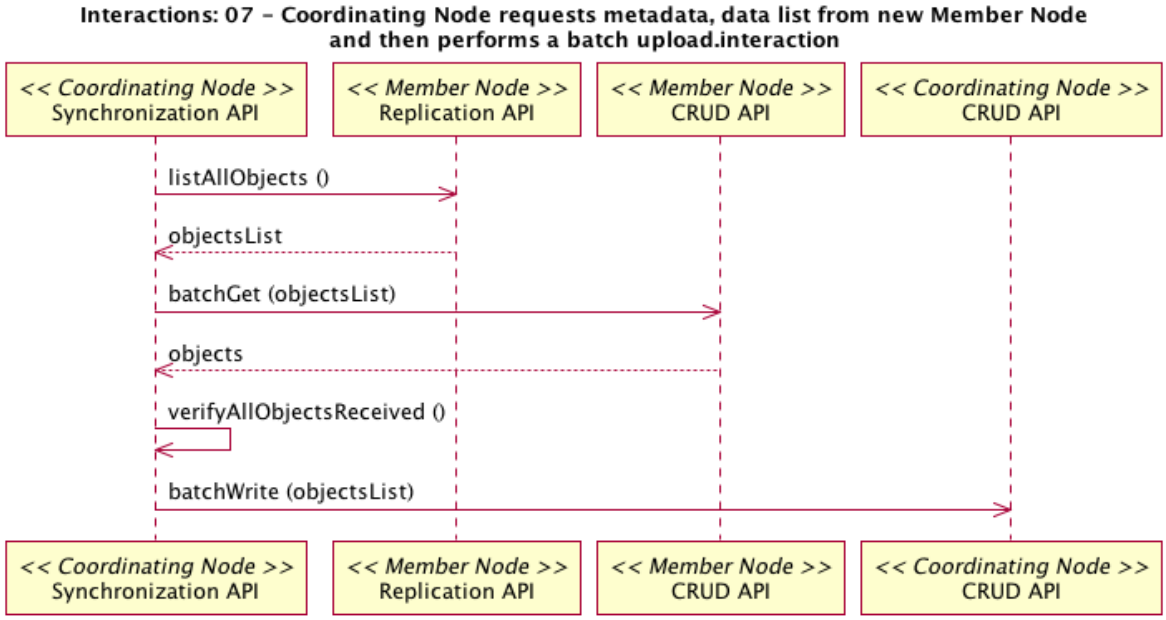
**Author**
VDC TWG

**Date**
- 20090930 Flagged as dupe
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
Batch Operations - Coordinating Node requests metadata /data list from new Member Node and then batch upload (disable indexing for example to improve insert performance).

**Summary**
This is a dupe of UC06 after refactoring that use case use a pull mechanism to retrieve changed records.




**Actors**
xxx

**Preconditions**
xxx

**Triggers**
xxx

**Post Conditions**
xxx

### *3.7.1 Notes*

xxx

## 3.8 Use Case 08 - Replication Policy Communication

**Author**
VDC TWG

**Date**
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
Communication of replication policy metadata between Member Nodes and Coordinating Nodes.

Summary

The replication policy of Member Nodes (MN) indicates factors such as the amount of storage space available, bandwidth constraints, the types of data and metadata that can be managed, and perhaps access control restrictions. This information is used by Coordinating Nodes (CN) to balance the distribution of data packages throughout the DataONE system to achieve the goals of data package persistence and accessibility.

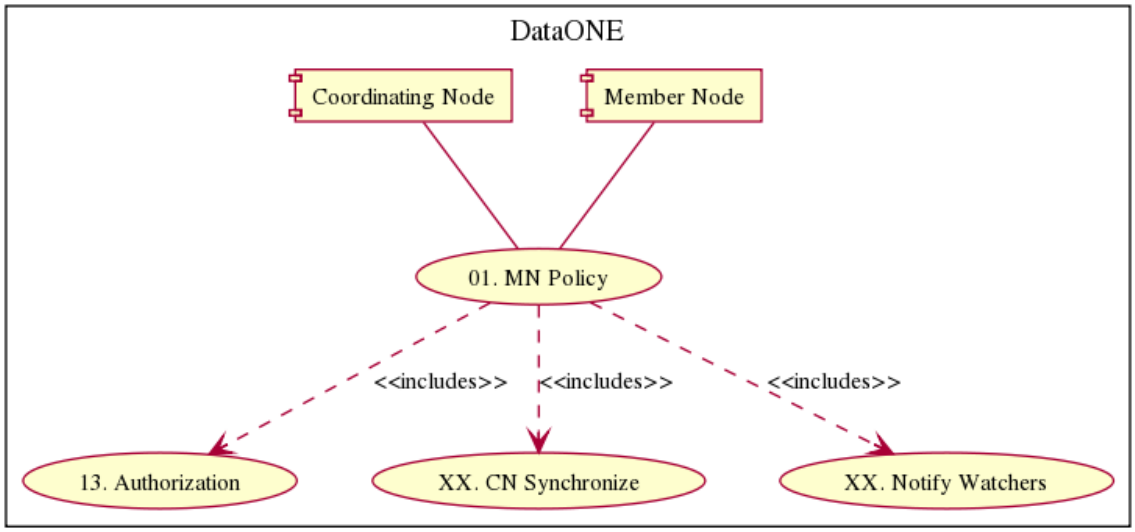
**Actors**
Member Node, Coordinating Node

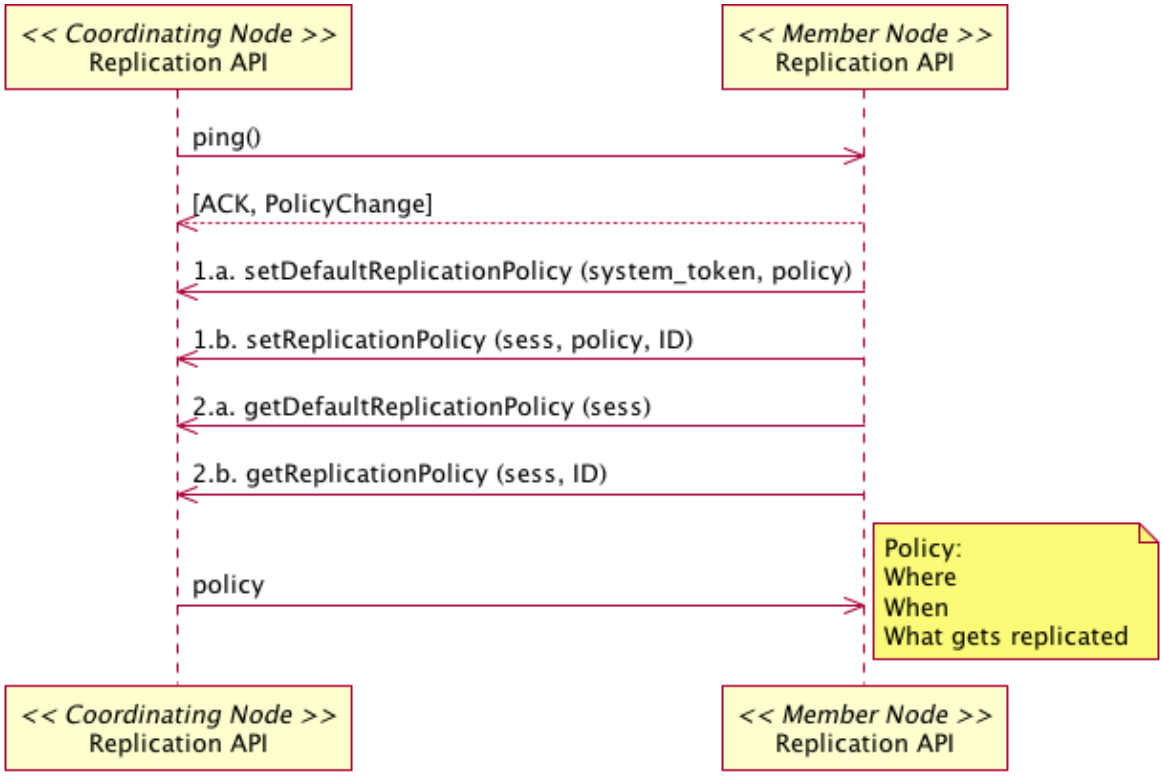
**Preconditions**
- Member Node is registered with a Coordinating Node
- Member Node is operational

**Triggers**
- A Member Node changes available capacity, bandwidth or some other operating characteristic
- A Member Node changes available services
- A Member Node software stack is updated (e.g. new version)

**Post Conditions**
- The DataONE system is updated with current state of available resources
- Change in replication policy may trigger a re-synchronization (e.g. if storage capacity shrinks or is enlarged)
- Operation is logged
- Watchers are notified in change in DataONE system property

**Notes**
- The goal as stated originally for this use case was "Communication of replication policy metadata among Member Nodes and Coordinating Nodes."
- There should be some restriction on how often replication policy can be changed to avoid thrashing that might occur for example, if a MN flips between significant differences in storage capacities (low to high, then low again).




*Communication of Replication Policy Metadata*

## 3.9 Use Case 09 - Replicate MN to MN

**Author**
VDC TWG

**Date**
- 20091001 Filled out description a bit more
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

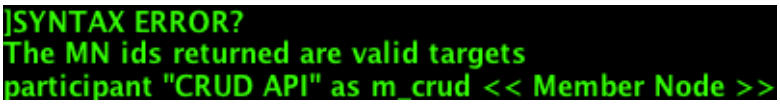
**Goal**
Replicate data from Member Node to Member Node.

Summary

Replication of content between Member Nodes (MN) is done to improve persistence of information (avoid data loss with loss of MN) and to improve accessibility (more choices for content retrieval can lower bandwidth requirements for any particular MN). The process of replication may be facilitated by a Coordinating Node (CN), or perhaps only between MNs. The interaction diagram presented here follows the former notion, where replication is facilitated by a CN.

A full copy of a data package is made during the replication process, so the original experiment metadata and data is copied to the recipient MN. An important question is whether that content is stored in it's original form (for data this must be the case) or translated into a form that is supported by the MN (likely to be the case for experiment metadata).

If experiment metadata is translated into a local dialect by a recipient MN, then it is important that the original metadata is preserved on the CNs, as it is always possible that the original MN where the content was published may go offline or be removed from the DataONE system.




**Actors**
Member Node x2, Coordinating Node

**Preconditions**
- Content is present on a Member Node

**Triggers**
- New data is available on a Member Node
- Information on a Member Node is altered
- Capabilities of a Member Node changes (accepting more or less content)
- Replication policy of DataONE or a Member Node changes

**Post Conditions**
- Content is present on one more Member Node
- System metadata is updated to reflect the change
- Watchers are notified of the change
- Member Node and Coordinating Node logs are updated

### *3.9.1 Notes*

Null.

# 3.10 Use Case 10 - MN Status Reports

**Author**

VDC TWG

**Date**

- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**

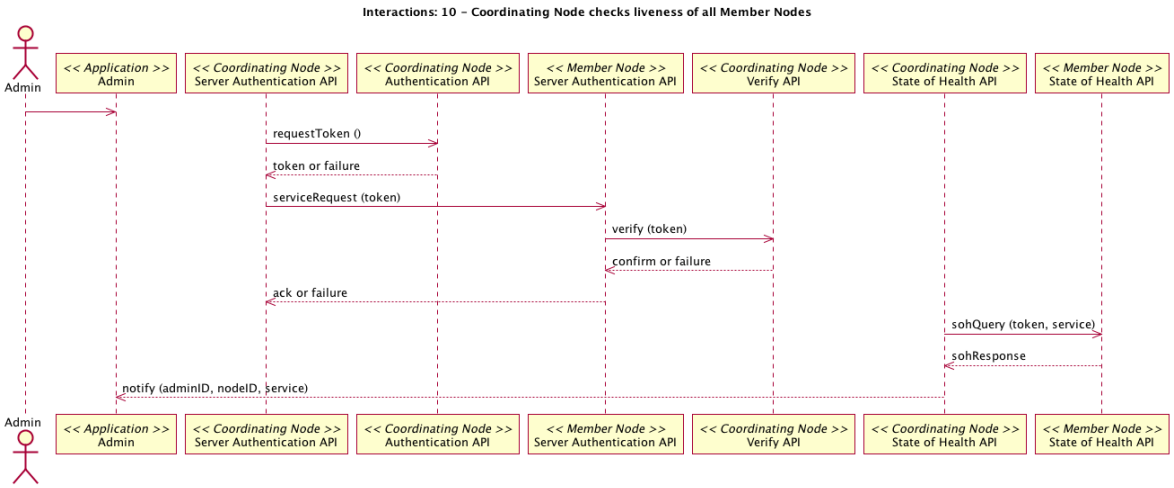
Draft

**Goal**

Coordinating Node checks "liveness" of all Member Nodes - checks ping, service x, load, space, bandwidth, transaction rate, ...

Summary

xxx




**Actors**

xxx

**Preconditions**

xxx

**Triggers**

xxx

**Post Conditions**

xxx

## *3.10.1 Notes*

xxx

# 3.11 Use Case 11 - CRUD Workflow Objects

**Author**
VDC TWG

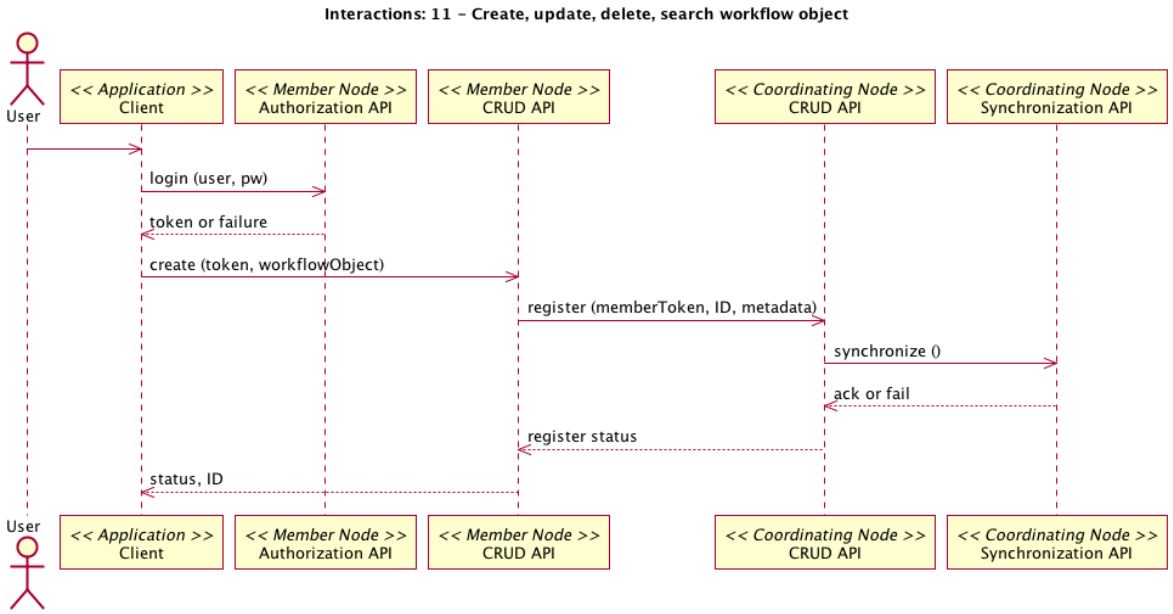
**Date**
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
Create / update / delete / search workflow objects.

Summary

xxx




**Actors**
xxx

**Preconditions**
xxx

**Triggers**
xxx

**Post Conditions**
xxx

## *3.11.1 Notes*

xxx

## 3.12 Use Case 12 - User Authentication

**Author**

VDC TWG

**Date**

- 20091001 Filled out more details
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**

Draft

**Goal**

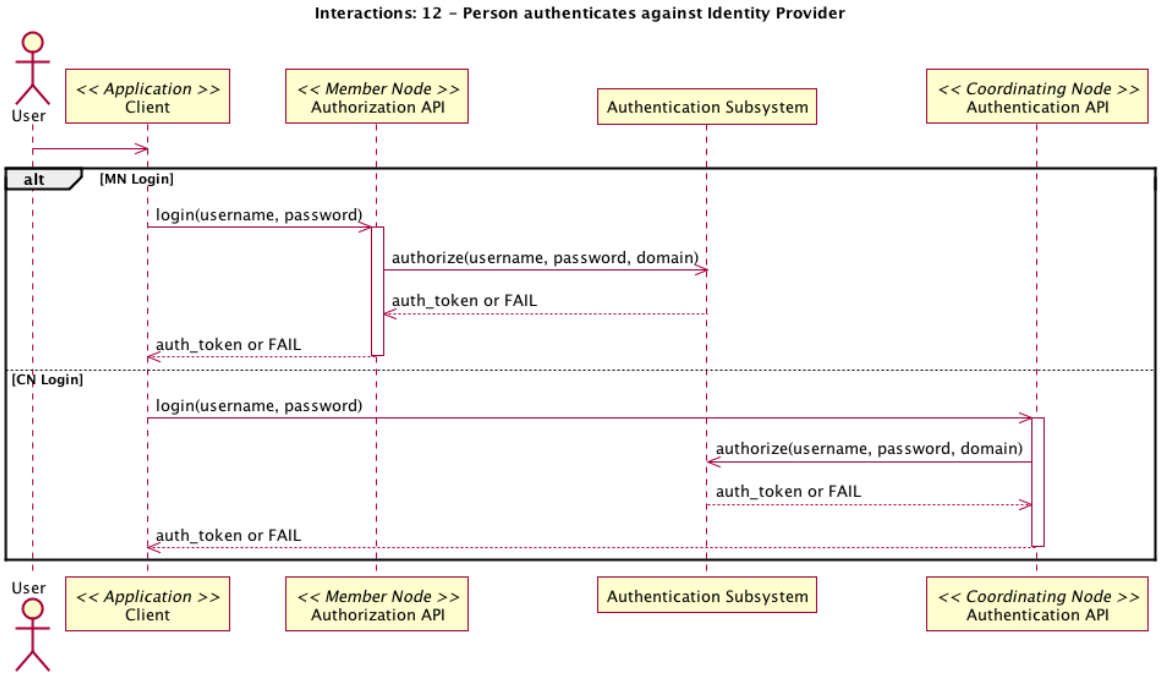
User Authentication - Person via client software authenticates against Identify Provider to establish session token.

**Summary**

Many operations in the DataONE system require user authentication to ensure that the user's identity is known to the system, and that appropriate access controls can be executed based on the identity.

The actual identity and authentication framework may exist outside of DataONE (one possible example might be the emerging CILogon service).

The authentication process might occur using either a Coordinating Node or a Member Node (if it implements the DataONE authentication API).




**Actors**

User, Member Node, Coordinating Node, Authentication System

**Preconditions**

- User is not authenticated in the system

**Triggers**

- A user logs on to the DataONE system.
- A user tries to perform an operation in the DataONE system that requires authentication.

**Post Conditions**

- An `auth_token` is returned to the user (application) that can be used for future interactions with the DataONE system.
- In the event of authentication failure, the `auth_token` will provide identity equivalent to an anonymous user.
- The authentication operation is recorded in MN or CN logs (depending on where the authentication attempt was made)

### *3.12.1 Notes*

- To be a bit more abstract, it might be good to use something like login(user, credentials) or login(subject) for the method signature. Here "subject" would be similar to the javax.security.auth.Subject concept. (PEA)
- Can logins also happening against a MN, which simply proxies the request up to the CN? (PEA)
- Need to specify a time to live for auth_tokens. Is this part of the authentication sub-system? If so, can the TTL be altered by the DataONE system?

## 3.13 Use Case 13 - User Authorization

**Author**
VDC TWG

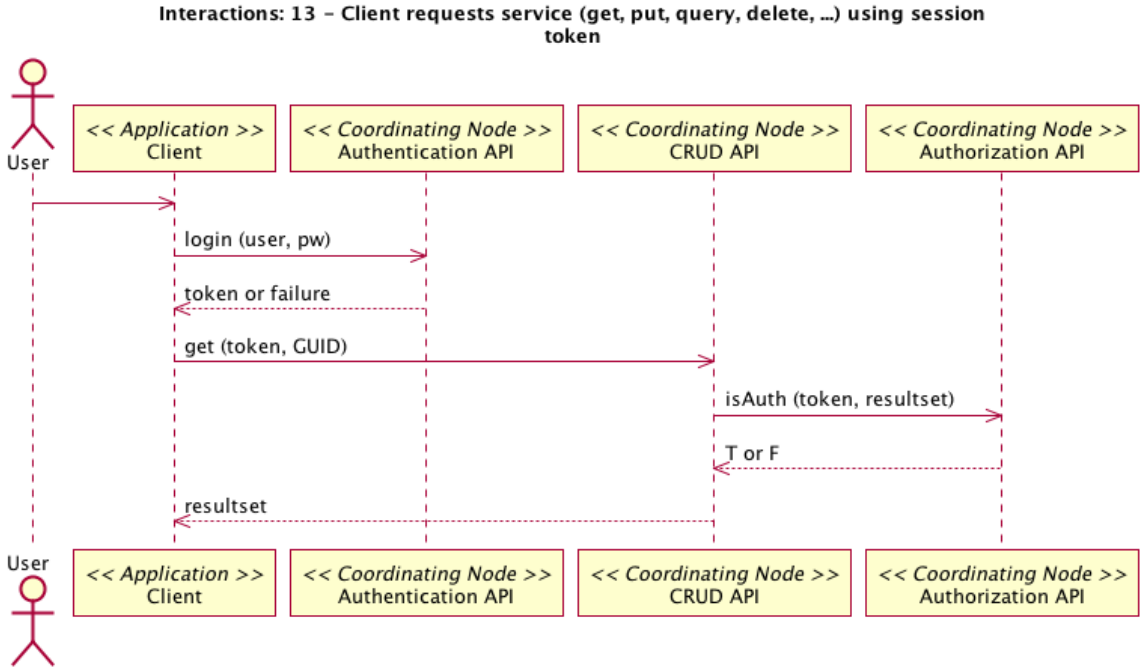
**Date**
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
User Authorization - Client requests service (get, put, query, delete, ...) using session token.

Summary

xxx




**Actors**
xxx

**Preconditions**
xxx

**Triggers**
xxx

**Post Conditions**
xxx

### *3.13.1 Notes*

- It would be great to factor out the authentication steps (#1, 2) from the diagram and replace it with some sort of precondition like "Client has authentication token." (PEA)

# 3.14 Use Case 14 - System Authentication and Authorization

**Author**
VDC TWG

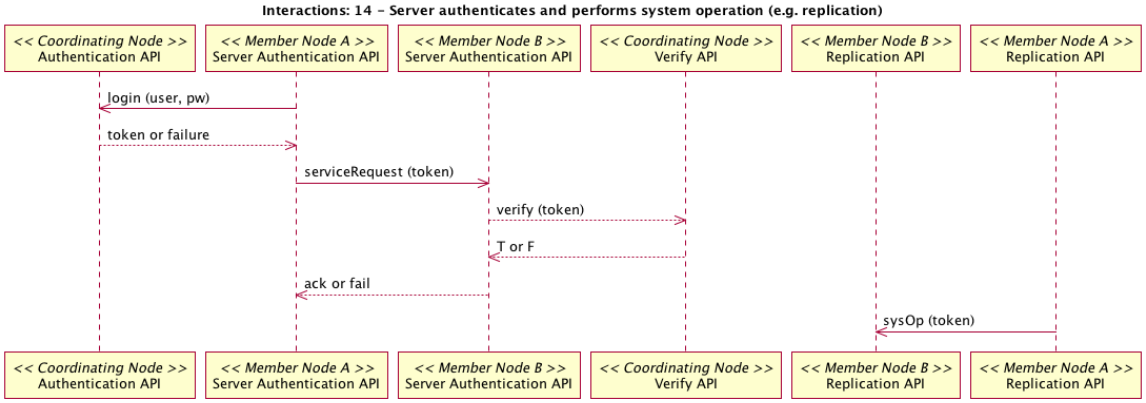
**Date**
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
System Authentication/Authorization - Server authenticates and performs system operations (e.g. replication).

Summary

xxx




**Actors**
xxx

**Preconditions**
xxx

**Triggers**
xxx

**Post Conditions**
xxx

## *3.14.1 Notes*

- I don't understand Step #7. To me the serviceRequest in step #3 is the actual request for MN-B to do a "system operation", which kicks off the verification steps #4 and #5. If this example service request is replication, then step #3 should point to the replication API, and internally MN-B decides that it must perform the verify step #4. Then step #6 is the response to the system operation, and step #7 is not needed. (I think that this diagram must be showing a kind of pre-approval process for the operation in step #7, but it doesn't make sense to me because in step #7, MN-B will just have to re-verify the token again.) (PEA)

## 3.15 Use Case 15 - Account Management

**Author**
VDC TWG

**Date**
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

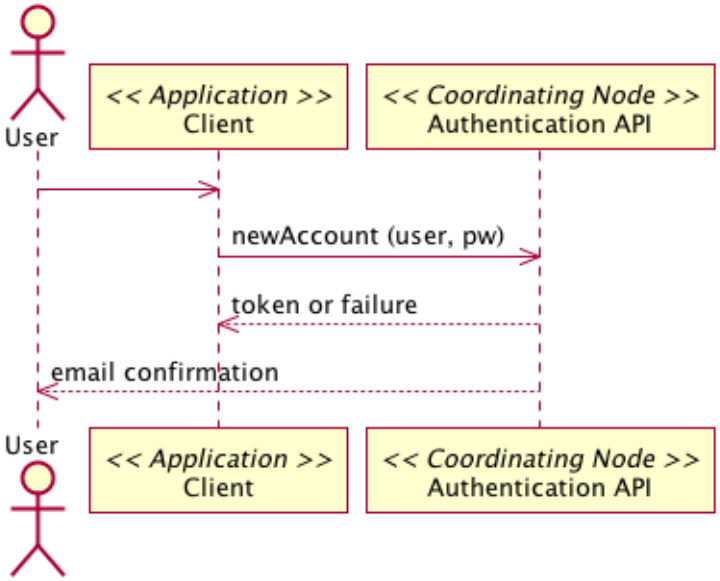
**Version**
Draft

**Goal**
User Account Management - Create new user account on Identity Provider (also edit, delete, ...).

Summary

xxx

**Interactions: 15 – Create new user account on Identity Provider (also edit, delete)**




**Actors**
xxx

**Preconditions**
xxx

**Triggers**
xxx

**Post Conditions**
xxx

### *3.15.1 Notes*

- This diagram should include a step for email confirmation.
- By default, accounts have no real privileges. To get higher privileges, users may have to jump through more hoops (such as verifying their association with a project/institution)
- Presumably, if we are using external identity providers this user account management functionality isn't provided by the CN. Right? (PEA)

## 3.16 Use Case 16 - Log CRUD Operations

**Author**
VDC TWG

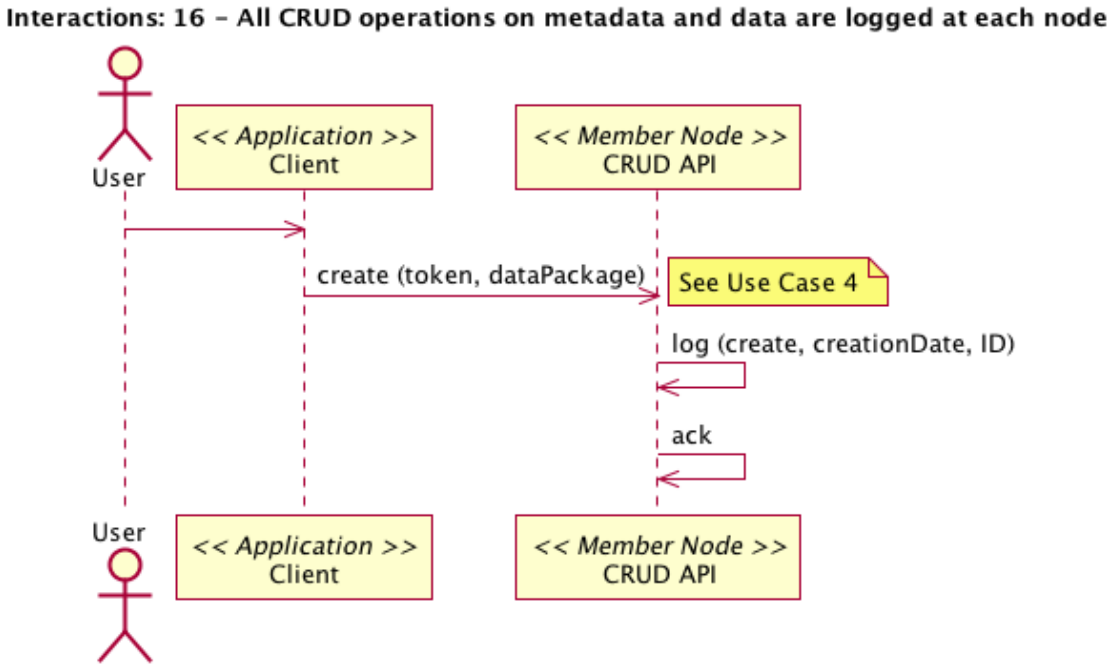
**Date**
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
All CRUD operations on metadata and data are logged at each node.

Summary

xxx




**Actors**
xxx

**Preconditions**
xxx

**Triggers**
xxx

**Post Conditions**
xxx

### *3.16.1 Notes*

xxx

# 3.17 Use Case 17 - CRUD Logs Aggregated at CNs

**Author**
VDC TWG

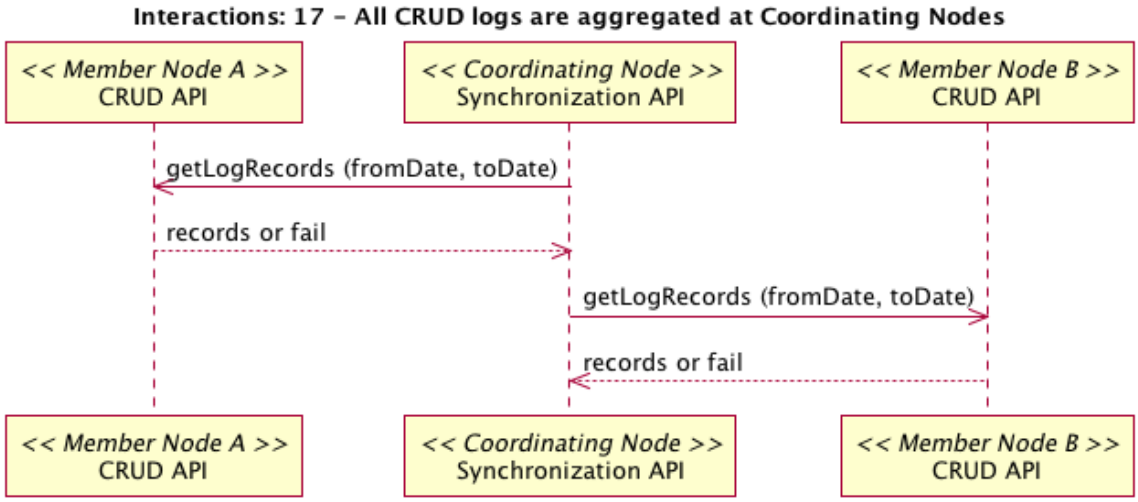
**Date**
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
All CRUD logs are aggregated at coordinating nodes.

Summary

xxx




**Actors**
xxx

**Preconditions**
xxx

**Triggers**
xxx

**Post Conditions**
xxx

## *3.17.1 Notes*

xxx

## 3.18 Use Case 18 - MN Retrieve Aggregated Logs

**Author**
VDC TWG

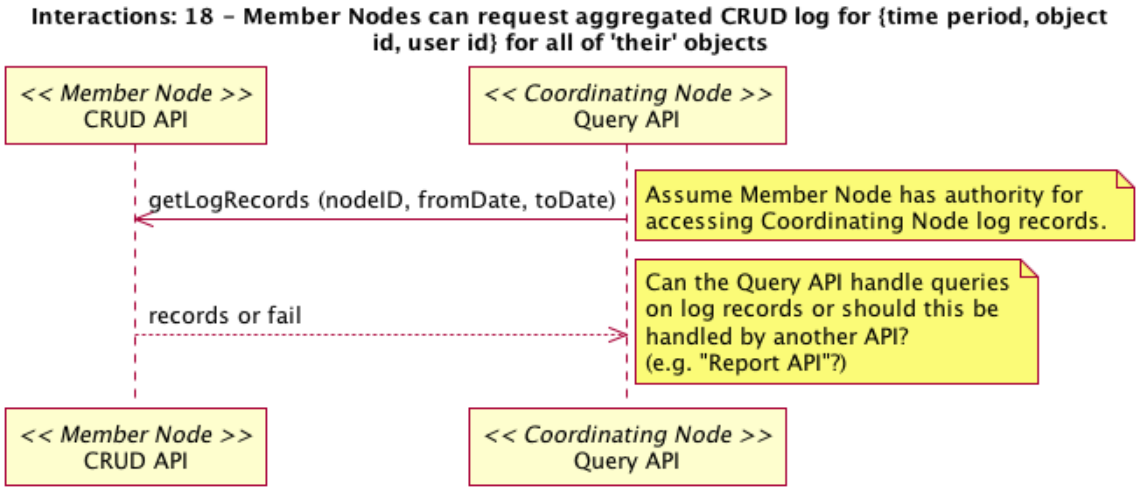
**Date**
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
Member nodes can request aggregated CRUD log for {time period/object id/userid} for all of 'their' objects.

Summary

xxx




**Actors**
xxx

**Preconditions**
xxx

**Triggers**
xxx

**Post Conditions**
xxx

### *3.18.1 Notes*

xxx

# 3.19 Use Case 19 - Retrieve Object Download Summary

**Author**
VDC TWG

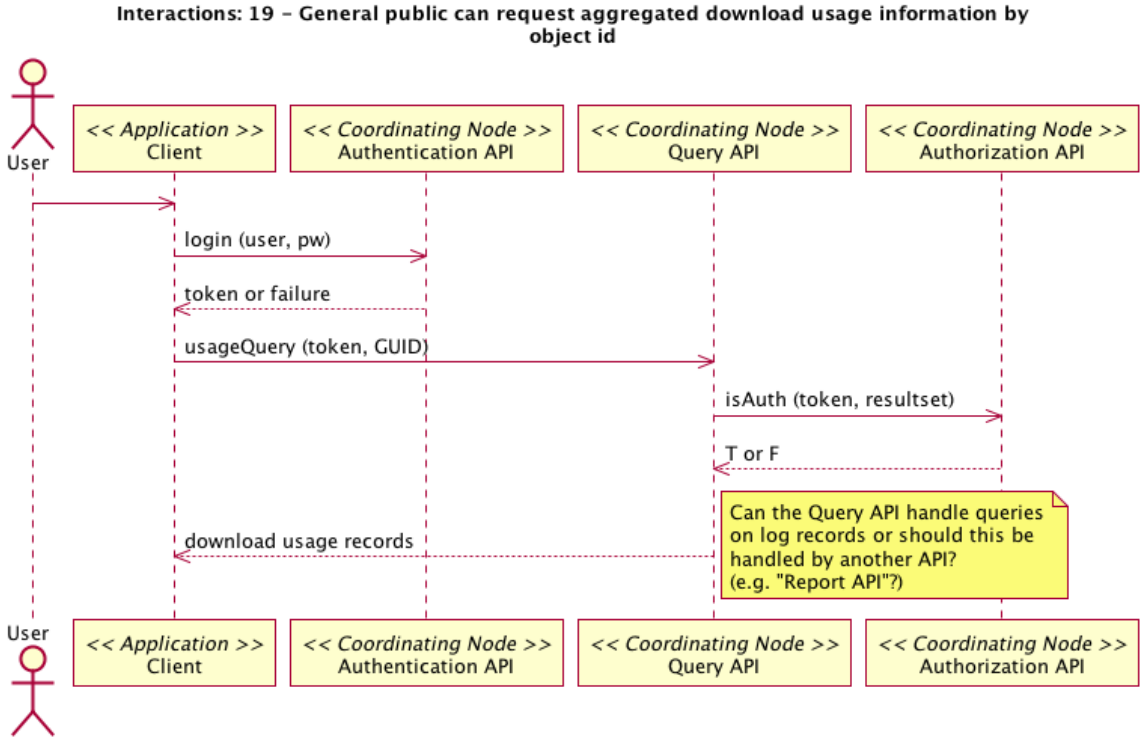
**Date**
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
General public can request aggregated download usage information by object id.

Summary

xxx




**Actors**
xxx

**Preconditions**
xxx

**Triggers**
xxx

**Post Conditions**
xxx

## *3.19.1 Notes*

xxx

# 3.20 Use Case 20 - Owner Retrieve Aggregate Logs

**Author**

VDC TWG

**Date**

- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**

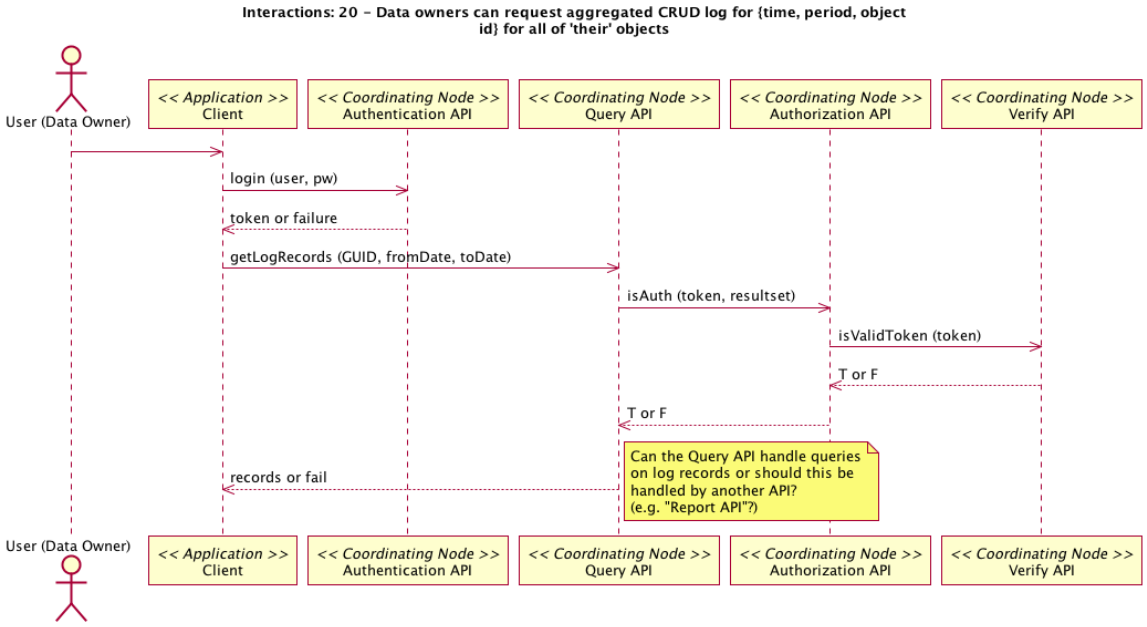
Draft

**Goal**

Data owners can request aggregated CRUD log for {time period/object id} for all of 'their' objects.

Summary

XXX




**Actors**

XXX

**Preconditions**

XXX

**Triggers**

XXX

**Post Conditions**

XXX

## *3.20.1 Notes*

XXX

## 3.21 Use Case 21 - Owner Subscribe to CRUD Operations

**Author**
VDC TWG

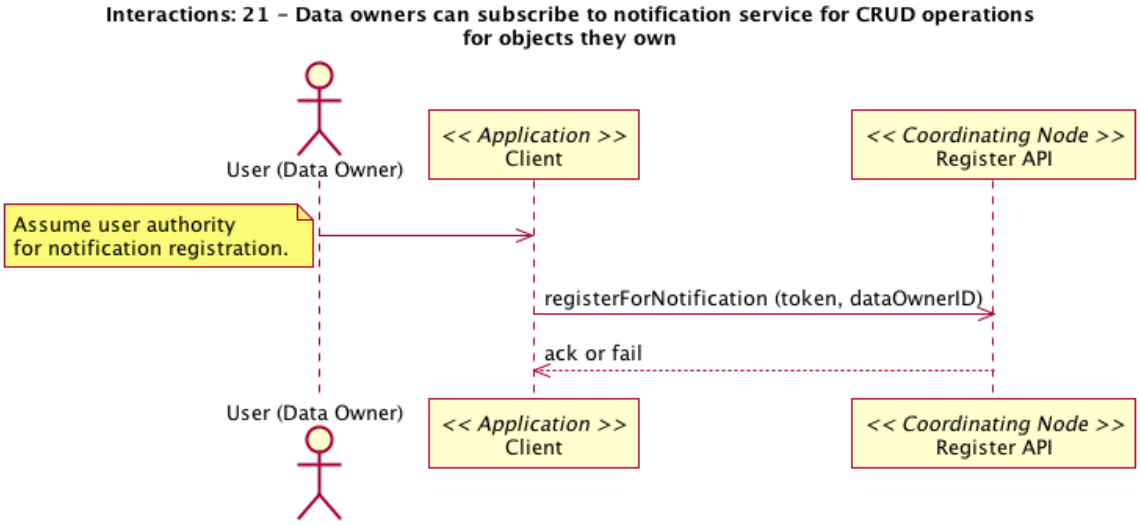
**Date**
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
Data owners can subscribe to notification service for CRUD operations for objects they own.

Summary

xxx




**Actors**
xxx

**Preconditions**
xxx

**Triggers**
xxx

**Post Conditions**
xxx

### *3.21.1 Notes*

xxx

# 3.22 Use Case 22 - Link/Citation Report for Owner

**Author**

VDC TWG

**Date**

- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**

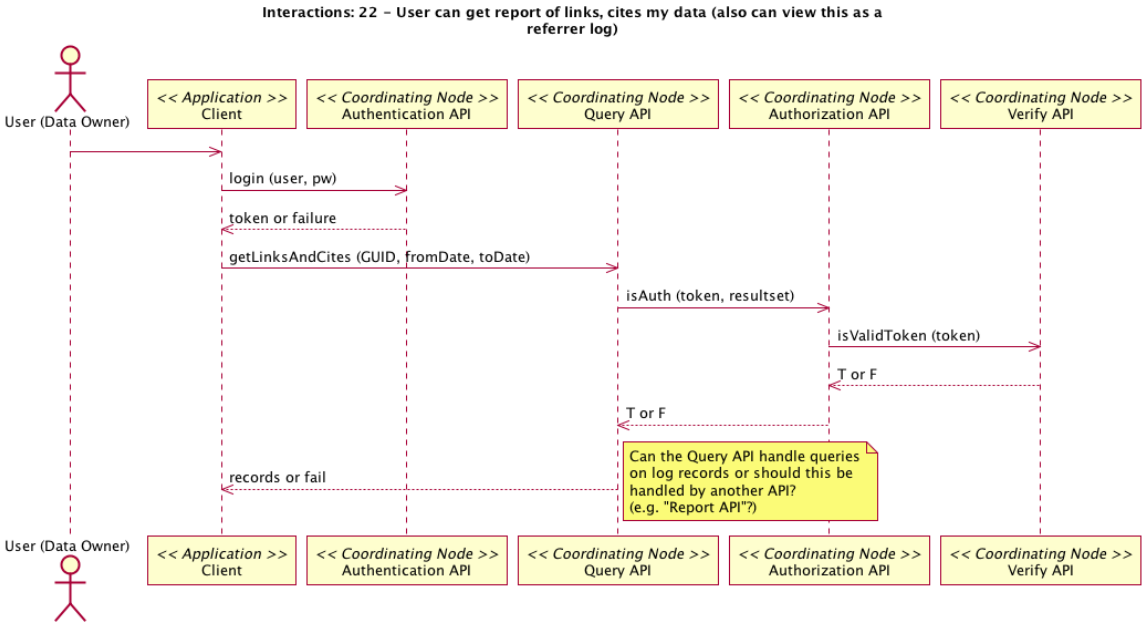
Draft

**Goal**

User can get report of links/cites my data (also can view this as a referrer log).

Summary

XXX




**Actors**

XXX

**Preconditions**

XXX

**Triggers**

XXX

**Post Conditions**

XXX

## *3.22.1 Notes*

XXX

# 3.23 Use Case 23 - Owner Expunge Content

**Author**

VDC TWG

**Date**

- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**

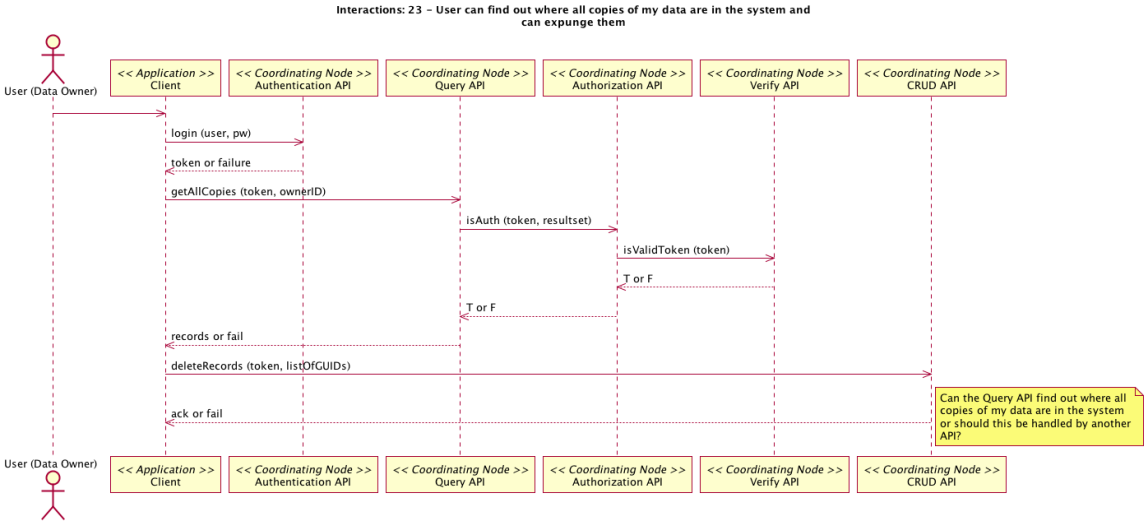
Draft

**Goal**

User can find out where all copies of my data are in the system and can expunge them.

Summary

XXX




**Actors**

XXX

**Preconditions**

XXX

**Triggers**

XXX

**Post Conditions**

XXX

## *3.23.1 Notes*

- Dealing with inappropriate usage of (ie, copyrighted objects that are stored)

# 3.24 Use Case 24 - MNs and CNs Support Transactions

**Author**
VDC TWG

**Date**
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
Transactions - CNs and MNs should support transaction sets where operations all complete successfully or get rolled back (e.g., upload both data and metadata records).

Summary

xxx

**Interactions: 24 – CNs and MNs should support transaction sets**

**Actors**
xxx

**Preconditions**
xxx

**Triggers**
xxx

**Post Conditions**
xxx

## *3.24.1 Notes*

- Does this include just one service, or sets of services that can be included in a transaction?
- Can transactions span multiple MNs? MNs and CNs?

## 3.25 Use Case 25 - Detect Damaged Content

**Author**
VDC TWG

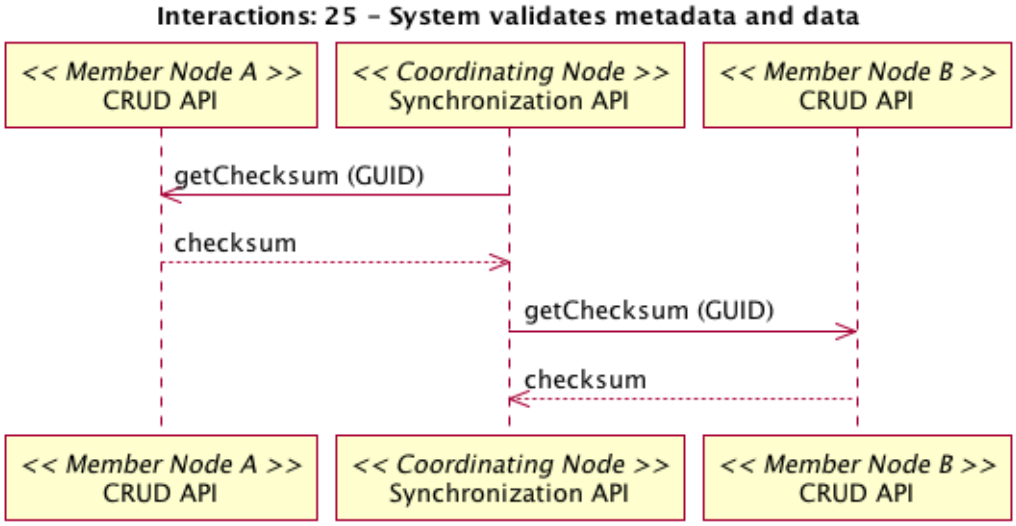
**Date**
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
System should scans for damaged/defaced data and metadata using some validation process.

Summary

xxx




**Actors**
xxx

**Preconditions**
xxx

**Triggers**
xxx

**Post Conditions**
xxx

### *3.25.1 Notes*

xxx

# 3.26 Use Case 26 - Data Quality Checks

**Author**
VDC TWG

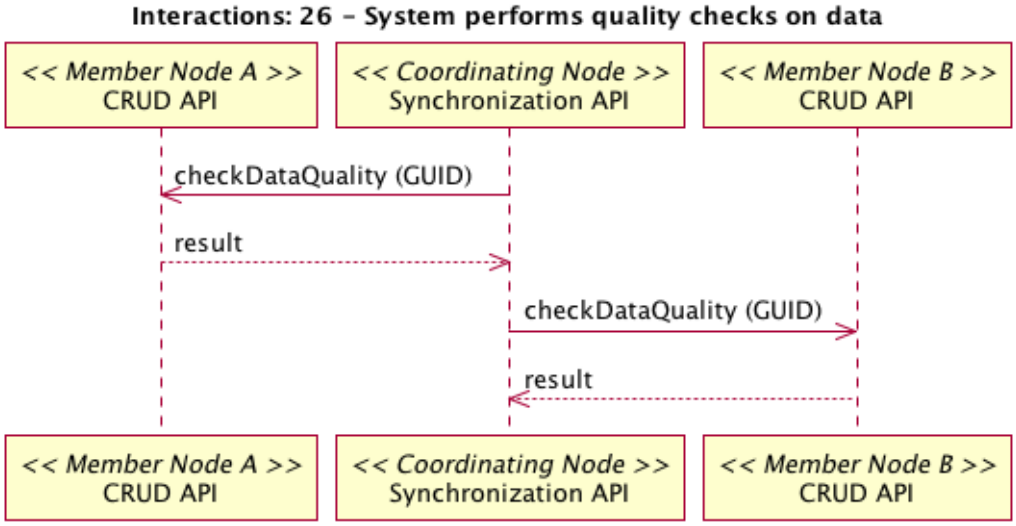
**Date**
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
System performs data quality checks on data.

Summary

xxx




**Actors**
xxx

**Preconditions**
xxx

**Triggers**
xxx

**Post Conditions**
xxx

## *3.26.1 Notes*

- Is this more for verifying the quality of metadata? Otherwise how does it differ from #25?

## 3.27 Use Case 27 - Metadata Version Migration

**Author**
VDC TWG

**Date**
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
CN should support forward migration of metadata documents from one version to another within a standard and to other standards.

Summary

xxx

**Interactions: 27 – Migration -- CN should support forward migration of metadata documents from one version to another within a standard and to other standards**

**Actors**
xxx

**Preconditions**
xxx

**Triggers**
xxx

**Post Conditions**
xxx

### *3.27.1 Notes*

- Should this apply to data as well?

# 3.28 Use Case 28 - Derived Product Original Change Notification

**Author**
VDC TWG

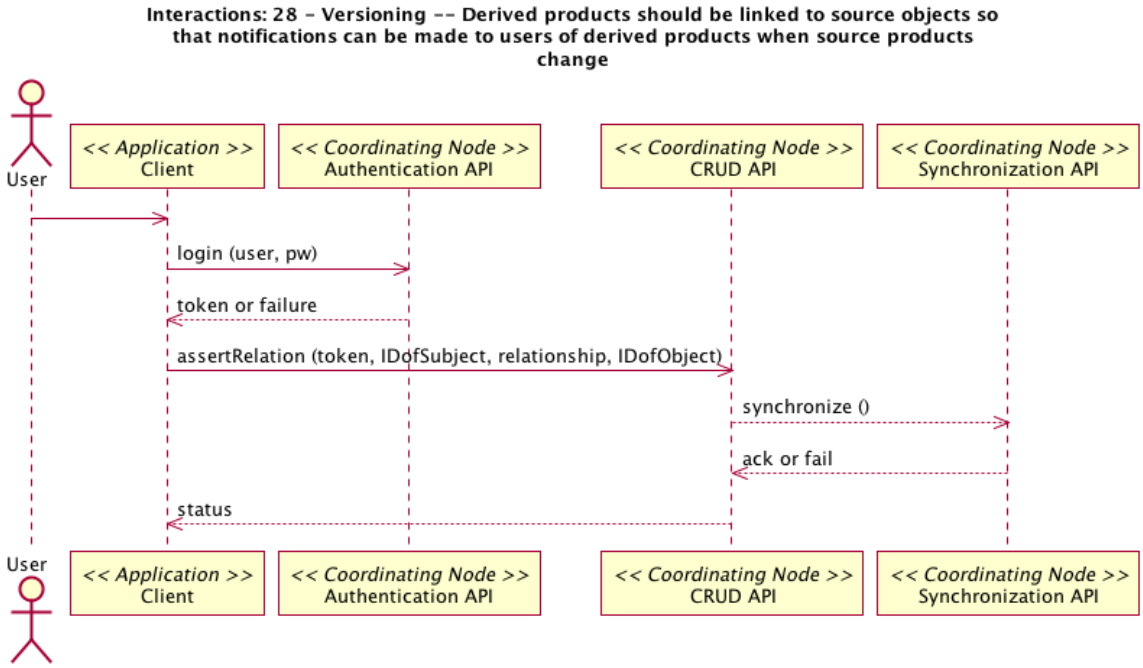
**Date**
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
Relationships/Versioning - Derived products should be linked to source objects so that notifications can be made to users of derived products when source products change.

Summary

xxx




**Actors**
xxx

**Preconditions**
xxx

**Triggers**
xxx

**Post Conditions**
xxx

## *3.28.1 Notes*

- Who can assert these relationships? The relevant object owners? Anybody?
- How are relationships removed?

# 3.29 Use Case 29 - CN Load Balancing

**Author**
VDC TWG

**Date**
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
Load Balancing - Requests to coordinating nodes are load balanced.

Summary

xxx

**Interactions: 29 – Load balancing between Coordinating Nodes**

**Actors**
xxx

**Preconditions**
xxx

**Triggers**
xxx

**Post Conditions**
xxx

## *3.29.1 Notes*

xxx

# 3.30 Use Case 30 - MN Outage Notification

**Author**
VDC TWG

**Date**
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

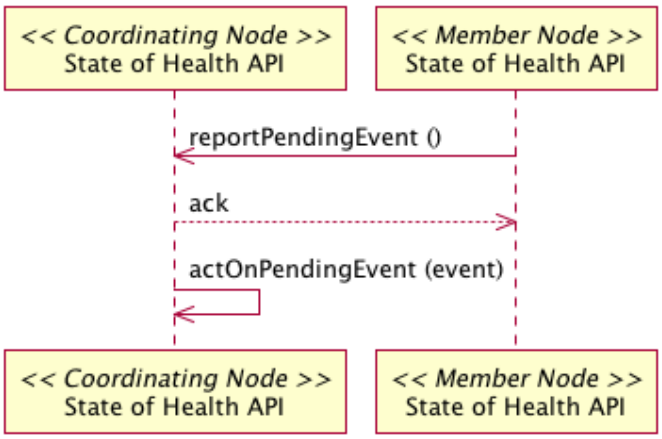
**Version**
Draft

**Goal**
MN can notify CN about pending outages, severity, and duration, and CNs may want to act on that knowledge to maintain seamless operation.

Summary

xxx

**Interactions: 30 – MN can notify CN about pending outages, severity and duration, and CNs may want to act on that knowledge to maintain seamless operation**




**Actors**
xxx

**Preconditions**
xxx

**Triggers**
xxx

**Post Conditions**
xxx

## *3.30.1 Notes*

xxx

## 3.31 Use Case 31 - Manage Access Policies

**Author**
VDC TWG

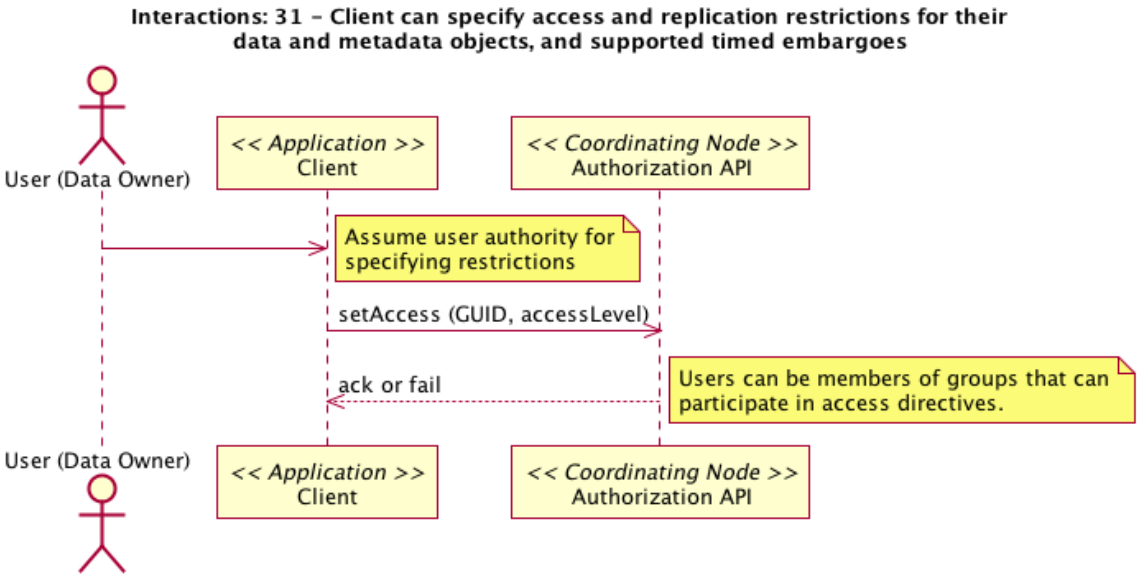
**Date**
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
Manage Access Policies - Client can specify access restrictions for their data and metadata objects. Also supports release time embargoes.

Summary

xxx




**Actors**
xxx

**Preconditions**
xxx

**Triggers**
xxx

**Post Conditions**
xxx

### *3.31.1 Notes*

- Users can be members of groups that can participate in access directives.
- I have removed the phrase "and replication" from the use case statement because use case #8 deals with setting replication policies. (PEA)
- Step #1, should have a signature of setAccess(token, GUID, accessPolicy). Even though the diagram says "Assume user authority for specifying restrictions", practically speaking we will need to verify that authority and the user's identify with a token. Also "accessLevel" sounds very limited, and access policy implies a possibly more sophisticated access policy delineation, including embargoes.

## 3.32 Use Case 32 - Transfer Object Ownership

**Author**
VDC TWG

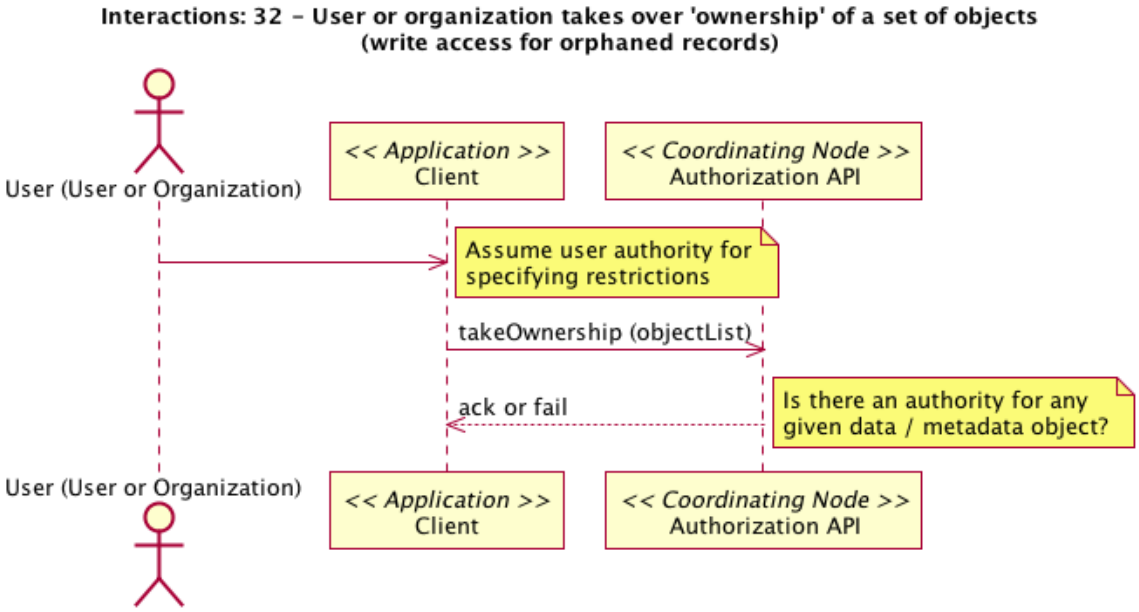
**Date**
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
User or organization takes over 'ownership' of a set of objects (write access for orphaned records)

Summary

xxx




**Actors**
xxx

**Preconditions**
xxx

**Triggers**
xxx

**Post Conditions**
xxx

### *3.32.1 Notes*

- Is there an authority for any given data/metadata object?

## 3.33 Use Case 33 - Search for Data

**Author**
VDC TWG

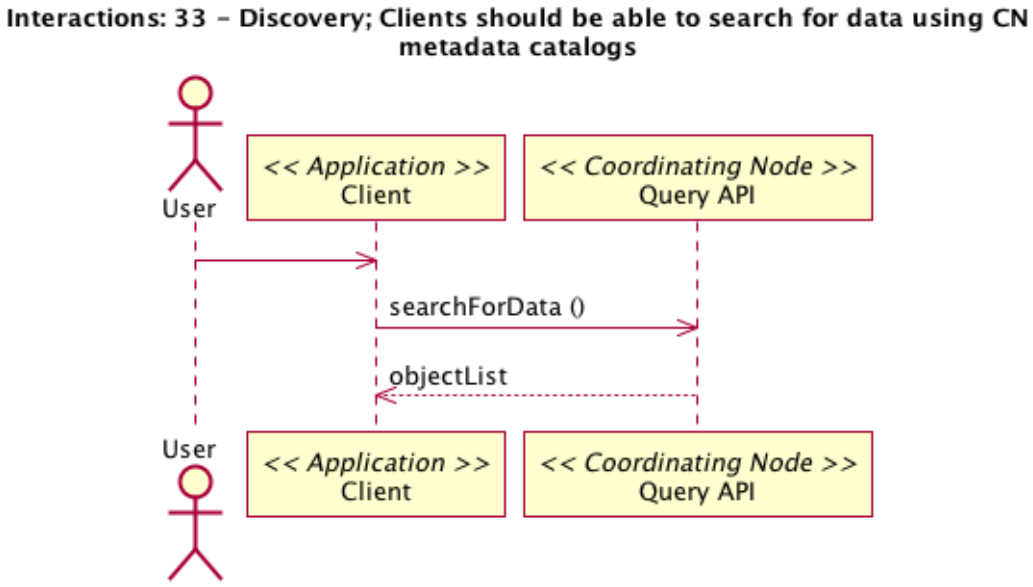
**Date**
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
Clients should be able to search for data using CN metadata catalogs.

Summary

xxx




**Actors**
xxx

**Preconditions**
xxx

**Triggers**
xxx

**Post Conditions**
xxx

### *3.33.1 Notes*

xxx

## 3.34 Use Case 34 - CNs Support Other Discovery Mechanisms (e.g. Google)

**Author**
VDC TWG

**Date**
- 20090925 Adding to use case description and documentation
- 20090114 (TWG meeting). Subsequent various modifications.

**Version**
Draft

**Goal**
Coordinating Nodes publish metadata in formats for other discovery services like Google/Libraries/GCMD/etc.

Summary

xxx

**Interactions: 34 – Discovery; Coordinating Nodes publish metadata in formats for other discovery services like Google, Libraries, GCMD, etc.**

**Actors**
xxx

**Preconditions**
xxx

**Triggers**
xxx

**Post Conditions**
xxx

### *3.34.1 Notes*

xxx

# 4 DataONE Application Programming Interfaces

isAuthorized (sess, transID) read (ID, sess, transID)

## 4.1 Member Node APIs

These APIs are implemented by Member Nodes.

### *4.1.1 Authorization API*

Provides a mechanism for a User to authenticate themselves with the DataONE system and verify that an operation is permitted for an authenticated User.

```
login (user, pw)
isAuth (token, GUID)
```

***Methods***

| Path | Parameters | Method | Returns | Description |
|---|---|---|---|---|
| /login/ | POST | user, password | auth_token or FAIL | Authenticates a user. The returned value is a composite of the user id and authentication token |
| /<GUID>/isAuth/ | GET | auth_token, GUID | permissions | The token is passed in the HTTP authorization header. Permissions are returned as a set of flags (can read, can write)? |

***Note***

Should add a mechanism for logging out, and perhaps convenience methods for "canRead()" and "canWrite()" to verify that a user is able to read / write an object

### *4.1.2 CRUD API*

```
batchGet (objectsList)
checkDataQuality (GUID)
create (token, dataPackage)
create (token, workflowObject)
get (token, handle)
getChecksum (GUID)
getLogRecords (fromDate, toDate)
log (create, creationDate, ID)
register (memberToken, ID, metadata)
```

### 4.1.3 Replication API

```
cancelReplication (sess, transID)
listAllObjects ()
listRecords ()
replicationSucceeds (transID)
startReplica (ID, sessionID, transID, srcMN)
sysOp (token)
```

### 4.1.4 Server Authentication API

### 4.1.5 Capabilities API

```
serviceRequest (token)
verify (token)
getCapabilities ()
State of Health API
sohQuery (token, service)
```

## 4.2 Coordinating Node APIs

### 4.2.1 Authentication API

```
login (user, pw)
requestToken ()
newAccount (user, pw)
```

### 4.2.2 Verify API

```
isValidToken (token)
Authorization API
isAuth (token, GUID)
isAuth (token, resultset)
isValidToken (token)
setAccess (GUID, accessLevel)
takeOwnership (objectList)
```

### 4.2.3 Query API

```
getAllCopies (token, ownerID)
getLinksAndCites (GUID, fromDate, toDate)
getLogRecords (GUID, fromDate, toDate)
getLogRecords (nodeID, fromDate, toDate)
query (token, query)
searchForData ()
usageQuery (token, GUID)
```

### 4.2.4 CRUD API

```
assertRelation (token, IDofSubject, relationship, IDofObject)
batchWrite (objectsList)
deleteRecords (token, listOfGUIDs)
get (token, GUID)
register (memberToken, ID, metadata)
resolve (GUID, token)
storeRecords ()
write ()
```

### 4.2.5 Replication API

```
getDefaultReplicationPolicy (sess)
getMNReplicateList ()
getReplicationPolicy (sess, ID)
isReplicationNeeded ()
replicateMetadata ()
setDefaultReplicationPolicy (sess, policy)
setReplicationPolicy (sess, policy, ID)
```

### 4.2.6 Synchronization API

```
pollForNewIDs ()
synchronize ()
verifyAllRecordsReceived ()
```

### 4.2.7 Register API

```
addNodeCapabilities ()
register (token, capabilitiesURL)
registerForNotification (token, dataOwnerID)
scheduleSynch ()
verifyCapabilities ()
```

### 4.2.8 Server Authentication API

### *4.2.9 State of Health API*

```
actOnPendingEvent (event)
reportPendingEvent (event)
Admin Application
notify (adminID, nodeID, service)
```