Data Management Use Cases

Version 2.5

July 31, 2019

These use cases describe how researchers manage research data. Use cases DM-01 through DM-04 and DM-11 describe general data management tasks. Use cases DM-05 through DM-10 describe how researchers create and use metadata when managing their research data.

DM-01: Create and share a data collection

A **researcher** needs to create a data collection on a storage resource and share the collection's use and management with a community of researchers who are using many resources. We assume the researcher has registered and has an allocation to use the storage resource. We also assume the researchers with whom the collection is shared have registered.

In most cases, the researcher wants to experience it as follows.

1. First, the researcher reviews documentation provided by the storage resource operators to find out where and how data collections should be stored on the system. (E.g., specific filesystems, paths, data transfer endpoints, storage quotas, allowable storage durations, etc.)
2. Then, the researcher moves (or copies) data to the storage resource from its original location(s).
3. Then, the researcher organizes the structure of the collection (directory hierarchies, file locations).
4. Then, the researcher sets permissions on parts of the collection to allow data to be read and/or modified by anonymous individuals, any authenticated individuals, or a specific set of individuals.

We'll accept any solution as long as the following are true.

1. Authentication is required for Steps 2-4 and it uses the community's standard authentication mechanism.
2. In Step 4, it isn't possible to allow anonymous individuals to modify data.

DM-02: Coordinated computing with a shared data collection

A **group of researchers** needs to conduct a coordinated activity using a shared data collection. (See use case DM-01.) The activity consists of multiple computational, analysis, or visualization tasks, running on multiple compute resources, all acting on the same data collection.

In most cases, each researcher wants to experience it as follows.

1. First, the researcher constructs a job that uses data from the shared data collection and that also specifies a location in the collection where new data should be placed.
2. Then, the researcher submits the job for execution on a compute resource.
3. The job executes, accesses the shared data collection, and produces output that is written to the shared data collection.
4. Steps 1 through 3 are repeated by the same or different researchers.

It will always be like this, except when the researchers use a science gateway or another application instead of HPC/HTC-style job submission. In that case, steps 1-3 are initiated by interacting with the application and the application accesses the data collection.

We'll accept any solution as long as the following are true.

1. When the job in Step 3 accesses the data collection, access control for existing data and attribution of the new data is based on the researcher's identity, which must be securely established.
2. When the application in the alternate scenario accesses the data collection, access control for existing data and attribution of the new data may be based either on the researcher's identity or a "group" identity shared by the research team, depending on how the application is designed.
3. Many instances of Step 3 of this use case (or the alternate scenario) may be happening in parallel.
4. As soon as new data is added to the collection in Step 3 (or via the alternate scenario), it may be used in subsequent iterations of this use case.

DM-03: Automate data ingestion from a set of sensors or instruments

A **researcher** needs to set up automatic and continuous data ingestion into a data collection from a set of sensors or instruments. We assume the researcher has registered and is authorized to add data to a data collection on a storage resource. We also assume the researcher has access to the system(s) where new data from the sensor(s) or instrument(s) is available.

In most cases, the researcher wants to experience it as follows.

1. First, the researcher visits the community website and finds information on the community's data transfer API and related tools.
2. Then, the researcher downloads and installs a tool on the system(s) that have access to new data from the sensor(s) or instrument(s).
3. Then, the researcher configures the system(s) to automatically run the tool when new data is available and specifies a location in the data collection where the new data should be placed.
4. The tool runs automatically when new data is available, and the new data is added to the data collection. This continues until the researcher disables the automation.

It will always be like this except when the researcher uses a workflow service to implement the automation. In that case, Steps 2-4 are replaced as follows. First, the researcher defines an ingest "job" that looks for new data on the source system(s) and adds it to the data collection, then the researcher configures the workflow service to run the job periodically.

We'll accept any solution as long as the following are true.

1. In Steps 3 and 4 (and in the alternate scenario), access control for the data collection and attribution of the new data is based on the researcher's identity, which must be securely established.
2. In the alternate scenario, authentication and authorization of the workflow job is based on the researcher's identity, which must be securely established.
3. The system allows an overall ingestion rate up to 10MB/sec.

DM-04: Migrate data to a new resource

A **researcher** or **educator** needs to migrate data to a new resource when the researcher's allocation ends or when a resource is decommissioned. We assume the researcher has registered, is still authorized to use the resource where the data is stored, and is also authorized to use the new resource. We also assume the researcher is authorized to access the data that will be migrated.

In most cases, the researcher wants to experience it as follows.

1. The researcher identifies folders of importance to their ongoing efforts.
2. The researcher initiates a command or script to perform data transfer operations on a collection of files and/or folders.
3. Data is transferred to the destination resource.
4. When data transfer is complete for all identified data, the researcher is notified of the completion of the transfer process.

It will always be like this except when the storage service provider offers an automatic method for Step 1, in which case the researcher doesn't have to personally identify the data to be moved.

We'll accept any solution as long as the following are true.

1. In Step 3, when the data is stored on the new resource, it is automatically verified to ensure the data is complete and hasn't been altered in transit.
2. 100% of data must be successfully transferred and verified.
3. Data that fails to be transferred or verified is automatically re-transferred.
4. The researcher mustn't need to run through Steps 1 and 2 more than once per storage system.
5. After Step 2, the researcher no longer needs to be connected to the system.

DM-05: Manually create metadata for a data object

A **researcher** needs to manually create metadata for a data object. The metadata is a set of attributes that describes the object and that can be used to index and/or search for the object when using metadata functions.

In most cases, the **researcher** wants to experience it as follows.

1. First, the researcher accesses the metadata function and selects a specific metadata collection to work with.
2. Then, the researcher provides an identifier for the data object being described.
3. Then, the researcher uses the interface to add or remove attributes to the object's metadata and to change the values of the object's metadata attributes.
4. Finally, the researcher is able to review the new metadata and either store it or save a local copy for later use.

It will always be like this except when the researcher chooses to store the metadata immediately, in which case the experience described in DM-08 should happen in parallel with this. (The steps may interweave.)

We'll accept any solution as long as in Step 4, the resulting metadata can be used with other metadata functions.

DM-06: Run a researcher-supplied tool to generate metadata for data objects

A **researcher** needs to generate metadata for one or more data objects using a tool or application selected by (and possibly provided by) the researcher. The metadata is a set of attributes that describes the objects and that can be used to index and/or search for the objects when using metadata functions. The researcher is responsible for installing the tool or application.

In most cases, the **researcher** wants to experience it as follows.

1. First, the researcher logs into the system where the tool or application is available.
2. Then, the researcher invokes the tool and specifies the data objects to be processed. The tool generates new metadata for the data objects.
3. Finally, the researcher is able to review the new metadata and either store it or save a local copy for later use.

It will always be like this except when the researcher chooses to store the metadata immediately, in which case the experience described in DM-08 should happen in parallel with this. (The steps may interweave.)

We'll accept any solution as long as in Step 3, the resulting metadata can be used with other metadata functions.

DM-07: Automatically extract metadata from data objects

A **researcher** needs to automatically extract metadata for one or more data objects. The metadata is a set of attributes that describes the objects and that can be used to index and/or search for the objects when using metadata functions.

In most cases, the **researcher** wants to experience it as follows.

1. First, the researcher logs into a system from which the data objects are accessible.
2. Then, the researcher invokes the metadata function and specifies the data objects to be processed. New metadata is automatically generated for the data objects.
3. Finally, the researcher is able to review the new metadata and either store it or save a local copy for later use.

It will always be like this except when the researcher chooses to store the metadata immediately, in which case the experience described in DM-08 should happen in parallel with this. (The steps may interweave.)

We'll accept any solution as long as the following are true.

1. In Step 2, the automatically generated metadata should include "ordinary" attributes like the following: filesystem metadata including filename, path, timestamps; database metadata including database and table names, record indices, query or view; the researcher's identity; the hostname and basic attributed about the host; a timestamp for the metadata extraction.
2. In Step 3, the resulting metadata can be used with other metadata functions.

DM-08: Store metadata for later use

A **researcher** needs to add metadata to a metadata collection so that it can be used later in search operations. We assume that the metadata has already been generated and is currently accessible to the researcher.

In most cases, the **researcher** wants to experience it as follows.

1. First, the researcher invokes the metadata function and either specifies a metadata collection to work with or creates a new collection.
2. Then, the researcher requests to add new metadata to the collection and references the new metadata. The metadata function performs any necessary validation of the new metadata and then it either returns an error (if there's something wrong with the metadata or the request) or it adds the metadata to the collection.
3. Finally, the researcher should be able to use the interface to confirm that the new metadata has been added.

It will always be like this except when the researcher needs to generate the metadata and add it to the collection at the same time. In that case, the steps described in either DM-05, DM-06, or DM-07 will be interwoven with these.

We'll accept any solution as long as the following are true.

1. The metadata management function should allow for metadata collections to have configurable requirements on the metadata that can be added to them. (E.g, formal schema or simpler attribute rules.)
2. If the metadata management function supports shared metadata collections (collections accessible by multiple people), it must also support and enforce configurable access controls for collections.
3. The metadata management function should also provide a mechanism for researchers to delete collections and manage the existing metadata in collections.

DM-09: Search metadata for specific objects of interest

A **researcher** needs to search a metadata collection to identify data objects of interest. We assume that the collection has already been populated with metadata about the relevant data objects.

In most cases, the researcher wants to experience it as follows.

1. First, the researcher invokes the metadata function and specifies a metadata collection to work with.
2. Then, the researcher enters a search query. The metadata function returns a set of data object identifiers whose metadata matches the search query.
3. Finally, the researcher is able to save the list of identifiers for subsequent use.

We'll accept any solution as long as in Step 2, the metadata function clearly indicates whether no metadata exists or no metadata was found that matched the parameters given.

DM-10: Add metadata search features to an application

A **science gateway developer** or **application developer** (hereafter referred to as the **developer**) needs to add metadata search features to an application he or she is developing for use by researchers. We assume that the collection(s) to be searched have already been populated with metadata about the relevant data objects. The developer is, of course, responsible for creating the application.

In most cases, the **developer** wants to experience it as follows.

1. First, the developer visits the community website, navigates to the developer documentation section, and scans or searches for information about the community's metadata functions. He or she should find documentation for a metadata search API.
2. Then, the developer uses the API documentation to add metadata search features to the application.
3. Finally, the developer makes the application available to researchers and the metadata features in the application work as described.

We'll accept any solution as long as the following are true.

1. In Step 1, the API should support the most common development environments used for science gateways and research applications, like: Python, Java, C/C++. (Ideally, the API would offer a generic API specification allowing developers to generate bindings for different languages.)
2. In Step 1, the API should be reasonably easy for research developers to understand and use.
3. In Step 2, if the developer encounters issues or needs help, he or she can submit a support request and receive support consistent with community expectations.

4. In Step 3, if researchers encounter issues with the metadata features and report them to the developer and the developer needs help resolving them, the developer can submit a support request and receive support consistent with community expectations.

DM-11: Post-allocation data access

A **researcher** needs to access his/her data on a community resource after his/her allocation on the resource ends. The researcher needs to continue using the same mechanisms and credentials that were used during the allocation, as opposed to having to learn a new set of mechanisms and/or credentials. We assume the resource is still integrated with the community and that the service provider (SP) for the resource offers a grace period for post-allocation data access.

In most cases, the researcher wants to experience it as follows.

1. First, during the researcher's allocation period, he or she uses community mechanisms to create data on a resource. (We assume these mechanisms use community identities.)
2. Then, when the allocation period is over and within a grace period determined by each SP, the researcher needs to use the same mechanisms (and the same community identity) to pull data off of the resource.
3. Finally, when the SP's grace period expires, the researcher will no longer be able to use the researcher's community identity or data access mechanisms with the resource.

We'll take any solution, as long as...

1. In Step 2, the user experience should be as close as possible to what it was during the allocation period. The researcher shouldn't have to switch to a new access mechanism or username/password.
2. In Step 3, each SP has the ability to determine and implement its own grace period, including the option of not having a grace period at all.

References

[1] XSEDE Data Management Use Cases v1.6. A Hossain, C Jordan, March 20, 2014. (http://hdl.handle.net/2142/48909)

[2] XSEDE Data Analytics Use Cases v0.3. Strande, et al, June 14, 2013. (http://hdl.handle.net/2142/45702)

History

	Version	Date	Changes	Author
Preceding document (See [2].)	0.3	6/14/2013	Final version of the original XSEDE-1 DA-02 use case: "Data Preparation"	Strande, et al (see doc)
Preceding document (See [1].)	1.6	3/20/2014	Final version of the original XSEDE-1 DM-05 use case: "Metadata Storage and Access"	Altaf Hossain, Chris Jordan
Entire document	2.0	2/27/2018	Divided original DM-05 into multiple use cases, each for a specific role/actor or variation	Liming
Entire document	2.5	7/31/2019	Reformatted DM-01 through DM-04 to the XSEDE-2 use case format; added DM-11; removed unnecessary XSEDE terminology	Liming