Cloud Data Management Interface Extension: Jobs

Version 1.0o

"Publication of this Working Draft for review and comment has been approved by the Cloud Storage Technical Working Group. This draft represents a "best effort" attempt by the Cloud Storage Technical Working Group to reach preliminary consensus, and it may be updated, replaced, or made obsolete at any time. This document should not be used as reference material or cited as other than a 'work in progress.' Suggestion for revision should be directed to http:/snia.org/feedback."

Working Draft

Revision History

Date	Version	By	Comments
2011-12-09	1.0a – 1.0c	Hitachi Data Systems: David Hartman, Andrew Glass, Shantanu Sinha, Siamak Roshan, and Kathy Wells	Initial Creation of proposed extension and revisions as discussed at face-to-face meetings during 2011.
2011-12-17	1.0d	David Slik	Updates to include standard SNIA front matter, minor edits to proposed extension as discussed at last face-to-face.
2011-12-22	1.0e	David Hartman	Clarification mechanisms to locate jobs and job containers.
2012-01-04	1.0f	Marie McMinn	Technical edit
2012-01-18	1.0g	David Hartman	Moved the URI for the job from the response body to the HTTP header for the response.
2012-01-19	1.0h	David Hartman	Reworded section for inclusion of the job URI in the HTTP response header.
2012-01-19	1.0i	Marie McMinn	Made a few editorial changes.
2012-01-25	1.0j	David Hartman	Incorporated changes to the capabilities tags as discussed in the 2012-01-24 CDMI TWG meeting. Changes include a new chapter for CDMI job containers.
2012-01-26	1.0k	Marie McMinn	Edited the new text.
2012-01-26	1.0l	David Hartman	Numerous changes to wording and edits to the example for creating a CDMI jobs container. Changes were made to the job container actions.
2012-01-26	1.0m	Marie McMinn	Reviewed last additions. Several small edits, but changed “must” to “shall” twice in last table.
2012-02-06	1.0n	David Slik	Updates to clarify capabilities for job containers.
2012-02-09	1.0o	Marie McMinn	Edited new text.

Jobs Extension

Overview

Some CDMI systems allow jobs (such as deletion, changing metadata, scanning for viruses, etc.) to be performed against CDMI objects. In such a system, multiple jobs may be performed simultaneously against the same object or may be performed against multiple objects. In these systems, the client needs be able to track the status of a job separately from the objects on which the jobs act. Jobs can also be batched, and a method is needed to track the status for the batch job independently of individual jobs that comprise the batch job.

This extension proposes a new type of data object to handle these requirements. The job data object (extended in a similar manner as a query queue object) can be used to define, perform, and track job status independently from the objects on which the job is acting.

Modifications to the CDMI 1.0.1 spec:

1) Insert into Clause "3 Terms"

3.11
job
a data object that manages one or more job actions that can be performed against one or more CDMI objects (job targets)

3.12
job action
a specific change in state performed on a per CDMI object basis as a consequence of a job being run against a CDMI object

Note: Examples include deletion, metadata changes, thumbnail creation, etc.

3.13
job container
a CDMI container object that is capable of storing CDMI job objects

3.14
job state
a value used to control the runtime state of a job

Note: Examples include start, stop, and cancel.

3.15
job target
the CDMI object or objects against which a job performs actions

2) Insert into Clause "12.1.1 Cloud Storage System-Wide Capabilities", Table "Table 102 - System-Wide Capabilities"

Capability Name	Type	Definition
cdmi_jobs	JSON String	If present and "true", this capability indicates that the cloud storage system supports job data objects.
cdmi_jobs_global_container	JSON String	If present, this capability contains the URI for the container for all job data objects in the cloud storage system.

3) Insert into Clause "12.1.3 Data System Metadata Capabilities", Table "Table 104 - Capabilities for Data System Metadata"

Capability Name	Type	Definition
cdmi_job_container_actions	JSON Array of JSON Strings	If present, this capability shall list the job action strings that can be requested for child job data objects created within a given container.

4) Insert into Clause "12.1.4 Data Object Capabilities", Table "Table 105 - Capabilities for Data Objects"

Capability Name	Type	Definition
cdmi_job_states	JSON Array of JSON Strings	If present, this capability shall list the job state strings that can be specified by a client.

5) Insert into Clause "12.1.5 Container Capabilities", Table "Table 106 - Capabilities for Containers"

Capability Name	Type	Definition
cdmi_create_job_container	JSON Array of JSON Strings	If present, this capability indicates that the container allows the creation of job container objects and shall list the job action strings supported for child job containers.
cdmi_create_job_dataobject	JSON String	If present and "true", this capability indicates that the container allows the creation of job data objects.

6) Insert into Clause "16.4 Support for Data System Metadata", Table "Table 118 - Data System Metadata"

Metadata Name	Type	Description	Requirement
cdmi_job_container_actions	JSON Array of JSON Strings	Contains a list of requested job actions to be permitted for job data objects created in the container. The job action strings that can be requested are indicated in the "cdmi_job_container_actions" capability of the parent container. If all supported actions are to be requested, the string "ALL" shall be used.	Optional

7) Insert into Clause "16.5 Support for Provided Data System Metadata", Table "Table 119 - Provided Values of Data Systems Metadata Items"

Metadata Name	Type	Description	Requirement
cdmi_job_container_actions_provided	JSON Array of JSON Strings	Contains a list of job actions that are permitted for job data objects created in the container.	Optional

8) Insert new Clause after 22 - "Query Queues":

23 Job Data Objects

23.1 Job Management

A cloud storage system may optionally implement job management functionality. Job implementation is indicated by the presence of the cloud storage system-wide capabilities for jobs and requires support for CDMI™ data objects.

Jobs allow arbitrary system-defined actions (such as deletion, metadata changes, thumbnail creation, virus scanning, etc.) to be performed against one or more stored CDMI objects. In addition, multiple jobs may perform actions against a single CDMI object. By creating a well-defined "job" object, clients can define jobs, specify which action is to be performed, specify which objects the action is to be performed against, monitor the status, and control the operation of the job in an interoperable and extensible manner.

In addition, multiple jobs may be batched together to apply actions sequentially for each target CDMI object. Such a batch job may affect multiple objects, and each job may progress at a different rate. The client cares about the overall status of its job, not the status on each object that the job affects. Tracking the job completion status in the completionStatus and percentComplete fields of the data object as described in Section 8.2 is not adequate for such systems.

These problems are solved by tracking the job status in a separate CDMI job data object. The job data object provides access to the completion status and percent complete of the job itself, along with other information required to define, monitor, and control the job.

Jobs may be stored in container objects or may exist as standalone data objects with no parent container.

Cloud storage systems should consider implementing support for job data objects when the system supports the following types of client-controlled activities:

- Server-side transformative operations: If the system allows a client to request that an operation be performed against a CDMI data object, the user should initiate and manage the operation through the jobs interface.
- Batch jobs: When running batch jobs that include multiple individual actions, the user needs to track the status for the jobs as the aggregate of the independent tasks.
- Multi-threading: If multiple jobs can be performed on the same object simultaneously, the user needs to track the status of each job independently.
- Long-running jobs: If jobs are run continuously, the user needs to be able to monitor and control the job.

23.2 Job Creation

When a client wishes to create a job data object, it may first check if the system is capable of providing job functionality by checking for the presence of the "cdmi_jobs" capability in the root container capabilities. If this capability is not present, creating a job data object shall be successful, but no job action shall be performed.

Jobs may be created by CDMI clients and from CDMI internal processes.

Examples of jobs created by CDMI clients may include

- deleting data,
- updating metadata, and
- serialization.

CDMI clients may create jobs through a variety of methods. The user may perform an HTTP operation such as a PUT or POST on a specific object in the cloud. The management framework may intercept and process the requests as an asynchronous job. The system shall create a job in a job container and return an HTTP response code of 202 Accepted. The URI for the job shall be included in an HTTP response header field named “X-CDMI-Job”.

A client may directly create a job through a POST or a PUT of a new job data object. The semantics for this are the same as other data objects. The container that accepts the job must have the “cdmi_job_dataobject” capability. The job-specific metadata shall be included in the request and response messages.

Examples of jobs created from internal system processes may include

- data migration,
- virus scans,
- search indexing, and

- periodic backups.

Although a user does not directly initiate these jobs, they can be exposed to the user since these jobs affect data in the system and consume system resources. Jobs that the system creates are created in a job container. To get a list of system-created jobs, clients can query the children of the container.

23.2 Required Metadata for a Job Data Object

When a client creates a job data object, the metadata described in Table 127 shall be provided. Attempts to change metadata in this table shall result in an HTTP status code of 403 Forbidden. After a job data object has been created, the metadata items in this table cannot be changed, except for cdmi_job_state and cdmi_job_action". The metadata item, cdmi_job_action, can only be removed, indicating to the system that the job data object shall no longer manage jobs; instead, it shall be treated as a regular CDMI data object.

Table 127 - Required Metadata for a Job Data Object

Metadata Name	Type	Description	Requirement
cdmi_job_action	JSON String	A system-defined identifier that indicates what action should be performed against each CDMI object that the job targets. Job actions defined as part of the CDMI specification (see FIXME) begin with the prefix "cdmi_job_action_". Job actions defined by vendors should begin with a reverse DNS notation such as "org.snia.” to prevent namespace conflicts. Only job actions specified in the data system metadata items listed in cdmi_job_container_actions_provided of the parent container of the job data object are considered supported actions.	Mandatory
cdmi_job_action_params	JSON Object or JSON Array	Contains job action-specific parameters that control how a job action behaves. For example, a thumbnail action may take parameters that indicate the height and width and/or desired size, output format, etc.	Optional

Metadata Name	Type	Description	Requirement
cdmi_job_state	JSON String	Controls the desired runtime state of the job. Defined values are one of the following: • "Start" indicates that the job should be performed. • "Pause" indicates that the job should be temporarily stopped. • "Cancel" indicates that the job should be permanently stopped. If this field is not present, a job state of "Running" shall be used. Only values specified in the "cdmi_job_states" capability of the job data object are considered valid states.	Optional
cdmi_job_target	JSON Array or JSON String	Indicates against which CDMI objects the job action is performed. Contains either an array of URIs to CDMI objects against which the job action shall be performed or a single URI to a CDMI queue. Each value enqueued in the queue is a URI to a CDMI object against which the job action shall be performed. For details on how queues are used with jobs, see FIXME.	Optional
cdmi_job_results	JSON String	Contains the URI to a CDMI queue that is used to indicate the results of performing a job. If present, the job shall enqueue a job-defined result value of performing the action against each job target.	Optional
cdmi_job_autodelete	JSON String	Contains the length of time until the job data object should be deleted, measured in seconds, since the time the job status transitions to "Complete", "Canceled", or "Error". If this field is not present, the job shall not be automatically deleted.	Optional
cdmi_job_scheduleTime	JSON String	The earliest time that the job can run, specified in ISO-8601 format (see 5.14). The job is scheduled to run as soon as possible if this field is omitted or if the time specified is earlier than the current system time.	Optional

EXAMPLE 1 An example of the metadata associated with a job data object is as follows:

```
{
     "metadata" : {
          "cdmi_job_action" : "cdmi_job_action_delete",
          "cdmi_job_target" : [
               "/cdmi_objectid/00007ED900100DA32EC94351F8970400",
               "/cdmi_objectid/00007ED90010F077F4EB1C99C87524CC",
               "/cdmi_objectid/00007ED90010512EB55A9304EAC5D4AA"
          ],
          "cdmi_job_autodelete" : "0"
     }
}
```

Jobs can be used in combination with query and notification queues to perform an action against each query result or notification result.

EXAMPLE 2 Combining jobs with query and notification queues is specified as follows:

```
{
     "metadata" : {
          "cdmi_job_action" : "cdmi_job_action_delete",
          "cdmi_job_target" : "/container/jobs/mp3_create_queue"
     }
}
```

23.3 Job Status Metadata

Table 128 describes the system-created metadata that provides details on the status of the job.

Table 128 - Job Status Metadata

Metadata Name	Type	Description	Requirement
cdmi_job_status	JSON String	A string that indicates the status of the job using one of the following values. • "Pending" indicates that the job object has been created but has not yet started running. • "Processing" indicates that the job is acting against the specified targets. • "Idle" indicates that the job has completed acting against the specified targets and will resume if additional targets are specified. • "Complete" indicates that the job has completed acting against the specified targets and will not resume. • "Canceled" indicates that the job was canceled before it acted against all of the specified targets. • A string that begins with "Error" indicates that an error prevented the job from acting against one or more of the specified targets.	Mandatory

Metadata Name	Type	Description	Requirement
cdmi_job_detailedStatus	JSON String	A message indicating what the job is currently doing or indicating the details about the error if it failed.	Optional
cdmi_job_percentComplete	JSON String	The value shall be an integer numeric value from 0 through 100.	Optional
cdmi_job_startTime	JSON String	When present, this metadata item indicates the time when the job started in ISO-8601 format (see 5.14).	Optional
cdmi_job_endTime	JSON String	When present, this metadata item indicates the time when the job completed, was halted, or went into an error status in ISO-8601 format (see 5.14).	Optional

EXAMPLE An example of the job status metadata associated with a completed job data object is as follows:

```
{
     "metadata" : {
          "cdmi_job_status" : "Complete",
          "cdmi_job_percentComplete" : "100",
          "cdmi_job_startTime" : "2011-12-17T22:14:24.323452Z",
          "cdmi_job_endTime" : "2011-12-17T22:14:28.834753Z"
     }
}
```

23.4 Job Lifecycle

Figure 1 shows the lifecycle of a job data object:

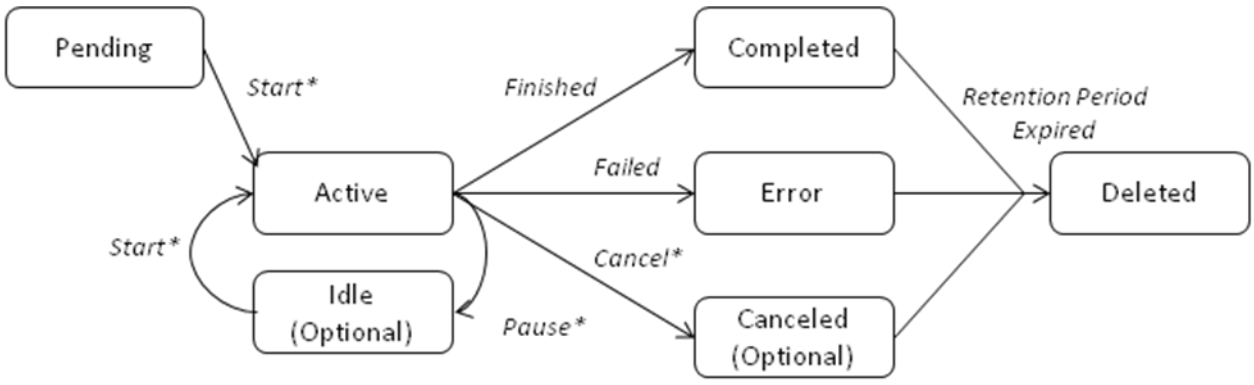

Figure 1 - Lifecycle of a Job Data Object

The following status values will be reflected in the cdmi_job_status field of the job data object: Pending, Active, Idle, Completed, Error, and Canceled.

The job is created in the Pending state. If it is started, it moves to the Active state. The job may optionally move between the Active and Idle states; however, all systems may not support the Idle state. The job moves to Completed, Error, or Canceled once it is finished. The Canceled state is optional, as it may not make sense in some systems. After completion, the job is retained until the client deletes the job or until the cdmi_job_autodelete period elapses.

The system can permit the client to start, pause, restart, or cancel a job using the cdmi_job_state metadata item. This functionality is optional, as the ability to directly control a job depends on the system.

23.5 Job Actions

A client can use the "cdmi_jobs_actions" system-wide capability to discover which job actions are supported.

Table 129 describes the job actions defined in this international standard.

Table 129 – Job Actions

Job Action	Description
cdmi_job_action_sequential_batch	Sequential batch jobs perform two or more jobs one after another against each targeted CDMI object in a specified order. • Sequential batch jobs have the job action identifier of "cdmi_job_action_batch_sequential". • The action parameters are an ordered JSON array of URIs to other job data objects that define the individual operations to be performed. Each of these component jobs shall not have a "job_target" or "job_state", as the "job_target" and "job_state" of the sequential batch job shall be used instead.
cdmi_job_action_parallel_batch	Parallel batch jobs perform two or more individual jobs in any order or at the same time against each targeted CDMI object. Parallel batch jobs should only perform job actions that do not alter the target data objects, or unspecified results may occur. • Parallel batch jobs have the job action identifier of "cdmi_job_action_batch_parallel". • The action parameters are a JSON array of URIs to other job data objects that define the individual operations to be performed. Each of these component jobs shall not have a "job_target" or "job_state", as the "job_target" and "job_state" of the parallel batch job shall be used instead.
cdmi_job_action_delete	Deletion jobs delete the target CDMI objects. • Delete jobs have the job action identifier of "cdmi_job_action_delete". • No job action parameters are required.

Job Action	Description
cdmi_job_action_update_metadata	Update metadata jobs manipulate the metadata of target CDMI objects. • Update metadata jobs have the job action identifier of "cdmi_job_action_update_metadata". • The action parameters of a JSON object contain three JSON containers: — "update_add" contains metadata items to be added to the data object if they don't already exist; — "update_modify" contains metadata items to be overwritten if they already exist; and — "update_delete" contains metadata items to be removed from the data object.

9) Insert new Clause after 23 - "Jobs":

24 Job Containers

24.1 Containers for Job Objects

CDMI job container objects can store job data objects. Job data objects are described in detail in Clause 23. Use of job containers is optional in CDMI systems but is mandatory if clients are permitted to create job data objects.

Job containers may be dedicated to storing only job data objects, or they may store other containers and data objects, including job data objects. CDMI systems may automatically create job containers, and in such systems, CDMI clients may not have the ability to create or delete job containers. Other systems may allow CDMI clients to create or delete job containers that support storing job data objects that the system or CDMI clients create.

A CDMI system may create and implement a single, global jobs container that CDMI clients cannot change. If present, clients can locate this global jobs container by the URI specified by the “cdmi_jobs_global_container” capability described in 12.1.1.

Systems may allow multiple job containers. Jobs may be grouped in containers along with non-job data objects. One use of multiple containers is to group jobs by type. Systems may allow CDMI clients to create their own job containers.

When job containers are supported, a CDMI client shall identify job containers using the “cdmi_job_container_actions” data system metadata capability described in 12.1.3.

The ability of a CDMI client to create a job container object within a container is indicated by the “cdmi_create_job_container” container capability described in 12.1.5. This capability also indicates any restrictions on job actions for a created child job container.

Once a job container has been created, the data system metadata of the cdmi_job_container_actions_provided contains an array of JSON strings that indicate the allowable actions that can be requested for job data objects that are created within that job container (see 16.5). The system generates this list depending on which actions are supported and which actions are requested in the data system metadata of the cdmi_job_container_actions described in 16.4.

A system may allow jobs to be created or deleted within a job container. This function is indicated by the capabilities associated with the job container.

- The ability of a CDMI client to create a job data object within a job container is indicated by the “cdmi_create_job_dataobject” container capability described in 12.1.5.
- The ability of a CDMI client to delete a job data object within a job container is indicated by the “cdmi_delete_dataobject” data object capability described in 12.1.3.

EXAMPLE Using capabilities and data system metadata, the client follows these steps to create a new job container that allows jobs for deleting CDMI objects:

1. Examine the presence and value of the “cdmi_create_job_container” capability of the parent container to see if child job containers can be created and if the cdmi_job_action_delete action is supported.
2. If job container creation is supported and the cdmi_job_action_delete action is supported, create a new child container with the cdmi_job_container_actions data system metadata set to "ALL" (or include the value "cdmi_job_action_delete") to indicate to the server that job data objects with delete job actions will be created in this newly created container.
3. Examine the cdmi_job_container_actions_provided data system metadata of the newly created container to ensure that "cdmi_job_action_delete" is included in the list.
4. Examine the “cdmi_create_job_dataobject” capability of the newly created container to ensure that job creation is supported.
5. If job data object creation is supported and the desired action is supported, create a new child data object with cdmi_jobs_action metadata supporting the cdmi_job_action_delete job action.

EXAMPLE An example of the job metadata associated with a job container that indicates that only delete action jobs can be created is as follows:

```
{
     "metadata" : {
          "cdmi_job_container_actions" : [
               "ALL"
          ],
          "cdmi_job_container_actions_provided" : [
               "cdmi_job_action_delete"
          ]
     }
}
```