Cloud Data Management Interface Profile: Self-Storage Management

Version 1.0e

"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
11/30/11	1.0a	Mark Carlson	Initial Creation
1/11/12	1.0b	Marie McMinn	Technical Edit
1/20/12	1.0c	Mark Carlson	Reapplied ballot comment changes that Marie missed
1/26/12	1.0d	Mark Carlson Marie McMinn	Changes per discussion in the face-to-face meeting. Minor edits.
2/8/12	1.0e	Mark Carlson	Changes per email feedback

Contents

Tables

Foreword

Abstract

This document defines a profile of the CDMI interface for managing data in a cloud storage environment.

SNIA Website

Current SNIA practice is to make updates and other information available through their website at http://www.snia.org.

SNIA Address

Storage Networking Industry Association, 425 Market Street, Suite #1020, San Francisco, CA 94105, U.S.A.

Introduction

This document is intended for application developers who are implementing or using cloud storage. It documents how to manage the data stored in a unified storage system.

This document is organized as follows:

1 - Scope	Defines the scope of this document
2 – Normative References	Lists the normative references for this document
3 - Terms	Provides terminology used in this document
4 - Conventions	Describes the conventions used in presenting the interfaces and the typographical conventions used in this document
5 - Self-Storage Management	Provides the normative standard of the profile with examples of typical usage

1 Scope

This profile documents how to manage the data stored in a storage system that supports standard block and file protocols. It applies to application developers who are implementing or using cloud storage.

2 Normative References

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

CDMI, Cloud Data Management Interface

3 Terms

See CDMI.

4 Conventions

See CDMI.

5 Self-Storage Management

5.1 Overview

CDMI can be used in situations where the CDMI data path (creation and access of data objects through the HTTP/REST interface) is not supported, and only other interfaces, such as block and file-based interfaces, are used to access the storage.

This profile is intended to allow CDMI to be used as a self-service interface to provision storage and manage data for offerings such as this.

5.2 Capabilities

A CDMI implementation conforming to this profile might implement the system-wide, data system metadata, and container capabilities described in Table 1, Table 2, and Table 3, respectively.

- The optional capabilities make sense for this profile but can be omitted by conformant implementations.
- The mandatory capabilities shall be implemented to conform to this profile.

The system-wide capabilities are listed in Table 1 (see 12.1.1 of CDMI).

Table 1 - System-Wide Capabilities

Capability	Description	Requirement
cdmi_domains	If present and "true", this capability indicates that the cloud storage system supports access control and billing through this interface.	Optional
cdmi_export_cifs	If present and "true", this capability indicates that the cloud storage system supports CIFS as a data path.	Optional
cdmi_dataobjects	If present and "true", this capability indicates that the cloud storage system supports the CDMI data path.	Optional
cdmi_export_iscsi	If present and "true", this capability indicates that the cloud storage system supports iSCSI as a data path.	Optional
cdmi_export_nfs	If present and "true", this capability indicates that the cloud storage system supports NFS as a data path.	Optional
cdmi_export_occi_iscsi	If present and "true", this capability indicates that the cloud storage system supports iSCSI for OCCI as a data path.	Optional
cdmi_export_webdav	If present and "true", this capability indicates that the cloud storage system supports WebDAV as a data path.	Optional

Capability	Description	Requirement
cdmi_metadata_maxitems	If present and "true", this capability indicates that the cloud storage system supports metadata on containers.	Optional
cdmi_metadata_maxsize	If present and "true", this capability indicates that the cloud storage system supports metadata on containers.	Optional
cdmi_metadata_maxtotalsize	If present and "true", this capability indicates that the cloud storage system supports metadata on containers.	Optional
cdmi_notification	If present and "true", this capability indicates that the cloud storage system supports queues and notifications.	Optional
cdmi_logging	If present and "true", this capability indicates that the cloud storage system supports queues and logging.	Optional
cdmi_queues	If present and "true", this capability indicates that the cloud storage system supports queues.	Optional
cdmi_security_access_control	If present and "true", this capability indicates that the cloud storage system supports ACLs.	Optional
cdmi_security_audit	If present and "true", this capability indicates that the cloud storage system supports queues and auditing.	Optional
cdmi_security_encryption	If present and "true", this capability indicates that the cloud storage system supports encrypting data at rest.	Optional
cdmi_serialization_json	If present and "true", this capability indicates that the cloud storage system supports JSON as a serialization format. This is the only supported serialization.	Mandatory
cdmi_snapshots	If present and "true", this capability indicates that the cloud storage system supports container snapshots.	Optional
cdmi_post_queue_by_ID	If present and "true", this capability indicates that the cloud storage system supports queues and objectID access.	Optional

The data system metadata capabilities are listed in Table 2 (see 12.1.3 of CDMI).

Table 2 - Data System Metadata Capabilities

Capability	Description	Requirement
cdmi_assignedsize	If present and "true", this capability indicates that the cloud storage system supports exported protocols that require size limits.	Mandatory

Capability	Description	Requirement
cdmi_data_redundancy	If present and "true", this capability indicates that the cloud storage system supports data mirroring.	Optional
cdmi_data_retention	If present and "true", this capability indicates that the system implements retention controls.	Optional
cdmi_data_autodelete	If present and "true", this capability indicates that the cloud storage system implements auto deletion after the retention period expires.	Optional
cdmi_data_holds	If present and "true", this capability indicates that the cloud storage system implements legal holds on the deletion of data.	Optional
cdmi_immediate_redundancy	If present and "true", this capability indicates that the cloud storage system implements synchronous mirroring.	Optional
cdmi_infrastructure_redundancy	If present and "true", this capability indicates that the cloud storage system implements no single point of failure.	Optional
cdmi_latency	If present and "true", this capability indicates that the cloud storage system implements tiering based on latency hints.	Optional
cdmi_RPO	If present and "true", this capability indicates that the cloud storage system implements backup/recovery.	Optional
cdmi_RTO	If present and "true", this capability indicates that the cloud storage system implements backup/recovery.	Optional
cdmi_throughput	If present and "true", this capability indicates that the cloud storage system implements placement based on throughput hints.	Optional

The container capabilities are listed in Table 3 (see Section 12.1.5 of CDMI).

Table 3 - Container Capabilities

Capability	Description	Requirement
cdmi_read_metadata	If present and "true", this capability indicates that the cloud storage system supports the ability to read metadata on the container.	Mandatory
cdmi_modify_metadata	If present and "true", this capability indicates that the cloud storage system supports the ability to modify metadata on the container.	Mandatory
cdmi_snapshot	If present and "true", this capability indicates that the cloud storage system implements container shapshots.	Optional

Capability	Description	Requirement
cdmi_create_container	If present and "true", this capability indicates that the cloud storage system supports creating containers in this container.	Optional
cdmi_cifs_export	If present and "true", this capability indicates that the container supports exporting it via CIFS.	Optional
cdmi_nfs_export	If present and "true", this capability indicates that the container supports exporting it via NFS.	Optional
cdmi_iscsi_export	If present and "true", this capability indicates that the container supports exporting it via iSCSI.	Optional
cdmi_occi_export	If present and "true", this capability indicates that the container supports exporting it via OCCI.	Optional
cdmi_webdav_export	If present and "true", this capability indicates that the container supports exporting it via WebDAV.	Optional
cdmi_delete_container	If present and "true", this capability indicates that the container can be deleted.	Mandatory
cdmi_move_container	If present and "true", this capability indicates that a container can be moved into this container.	Optional

5.3 Profile Operations

Because this profile is intended for storage systems where the CDMI data path is not used, the primary operations revolve around discovering available types of containers, provisioning new containers matching those types, and maintaining the container's lifecycle, including the data services that are deployed against the data contained within.

5.3.1 Discovering Available Types of Containers

Some storage systems may allow for the dynamic configuration of supported data services against any container that can be provisioned; however, the typical case is more likely to be pools of previously configured storage. The data services are likely to be already configured for each of these pools as well. This configuration, then, is represented by specific types of container capabilities that can be discovered in the capabilities tree under the cdmi_capabilities/container path. If pools of preconfigured storage exist, the implementation shall create children of the container capability resource to conform to this profile. Otherwise, an implementation can conform to this profile by allowing containers to be dynamically configured with any data service that the container capabilities support.

The following example shows fetching a container capabilities object with three children, each of which represents a type of container.

EXAMPLE 1 Perform a GET to the container capabilities URI:

```
GET /cdmi_capabilities/container/ HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-capability
X-CDMI-Specification-Version: 1.0.1
```

The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-capability
X-CDMI-Specification-Version: 1.0.1

{
   "objectType" : "application/cdmi-capability",
   "objectID" : "00007E7F0010CEC234AD9E3EBFE9531D",
   "objectName" : "container/",
   "parentURI" : "/cdmi_capabilities/",
    "parentID" : "00007E7F0010DCECC805FB6D195DDBCB",
   "capabilities" : {
       "cdmi_read_metadata" : "true",
       "cdmi_modify_metadata" : "true",
       "cdmi_nfs _export " : "true",
       "cdmi_webdav _export " : "true",
       "cdmi_iscsi _export " : "true",
       "cdmi_create_container" : "true",
        "cdmi_delete_container" : "true"
   },
   "childrenrange" : "0-2",
   "children" : [
       "gold/",
       "silver/",
        "bronze/"
   ]
}
```

The next step would be to discover the capabilities of each of these types of containers. First, we will fetch the capabilities of the bronze storage.

EXAMPLE 2 Perform a GET to the bronze container capabilities URI:

```
GET /cdmi_capabilities/container/bronze/ HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-capability
X-CDMI-Specification-Version: 1.0.1
```

The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-capability
X-CDMI-Specification-Version: 1.0.1

{
   "objectType" : "application/cdmi-capability",
   "objectID" : "00007E7F0010CEC234AD9E3EBFE9531D",
   "objectName" : "bronze/",
   "parentURI" : "/cdmi_capabilities/container/",
    "parentID" : "00007E7F0010DCECC805FB6D195DDBCB",
   "capabilities" : {
       "cdmi_read_metadata" : "true",
       "cdmi_modify_metadata" : "true",
       "cdmi_nfs _export " : "true",
       "cdmi_webdav _export " : "true",
       "cdmi_iscsi _export " : "true",
       "cdmi_create_container" : "true",
       "cdmi_delete_container" : "true",
       "cdmi_assignedsize" : "true"
```

```
    }
}
```

From this example, we can see that the bronze type of container is pretty basic, but it does support the export of NFS, iSCSI, and WebDAV. Next, we will fetch the capabilities of the silver storage.

EXAMPLE 3 Perform a GET to the silver container capabilities URI:

```
GET /cdmi_capabilities/container/silver/ HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-capability
X-CDMI-Specification-Version: 1.0.1
```

The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-capability
X-CDMI-Specification-Version: 1.0.1

{
   "objectType" : "application/cdmi-capability",
   "objectID" : "00007E7F0010CEC234AD9E3EBFE9531D",
   "objectName" : "silver/",
   "parentURI" : "/cdmi_capabilities/container/",
   "parentID" : "00007E7F0010DCECC805FB6D195DDBCB",
   "capabilities" : {
       "cdmi_read_metadata" : "true",
       "cdmi_modify_metadata" : "true",
       "cdmi_nfs _export " : "true",
       "cdmi_webdav _export " : "true",
       "cdmi_iscsi _export " : "true",
       "cdmi_create_container" : "true",
        "cdmi_delete_container" : "true",
        "cdmi_assignedsize" : "true",
        "cdmi_data_redundancy" : "2",
        "cdmi_immediate_redundancy" : "true",
        "cdmi_RPO" : "true",
        "cdmi_RTO" : "true"
   }
}
```

From this example, we can see that the silver type of container adds synchronous mirroring and backup and restore capabilities to the basic capabilities of the bronze container type.

Finally, we will fetch the capabilities of the gold storage.

EXAMPLE 4 Perform a GET to the gold container capabilities URI:

```
GET /cdmi_capabilities/container/gold/ HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-capability
X-CDMI-Specification-Version: 1.0.1
```

The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-capability
X-CDMI-Specification-Version: 1.0.1
```

```
{
   "objectType" : "application/cdmi-capability",
   "objectID" : "00007E7F0010CEC234AD9E3EBFE9531D",
   "objectName" : "gold/",
   "parentURI" : "/cdmi_capabilities/container/",
    "parentID" : "00007E7F0010DCECC805FB6D195DDBCB",
   "capabilities" : {
       "cdmi_read_metadata" : "true",
       "cdmi_modify_metadata" : "true",
       "cdmi_nfs _export " : "true",
       "cdmi_webdav _export " : "true",
       "cdmi_iscsi _export " : "true",
       "cdmi_create_container" : "true",
        "cdmi_delete_container" : "true",
        "cdmi_assignedsize" : "true",
        "cdmi_data_redundancy" : "true",
        "cdmi_immediate_redundancy" : "true",
        "cdmi_RPO" : "true",
        "cdmi_RTO" : "true",
        "cdmi_latency" : "true",
       "cdmi_throughput" : "true"
   }
}
```

From this example, we can see that the gold type of container adds tiering based on latency hints and placement based on throughput hints to the more enhanced capabilities of the silver container type.

5.3.2 Container Operations

An implementation that conforms to this profile shall support the following container operations:

- 9.2 Create a Container Object using CDMI Content Type
- 9.4 Read a Container Object using CDMI Content Type
- 9.6 Update a Container Object using CDMI Content Type
- 9.7 Delete a Container Object using CDMI Content Type

In addition, an implementation that conforms to this profile shall support at least one container export protocol or shall allow for use of the CDMI data path on that container.

5.3.2.1 Create a Container Object

Once the container capabilities and possibly specific pre-configured container types have been discovered, the discovered values can be used to create container objects that use those capabilities.

First, we will create a container object with capabilities according to the silver type that was discovered.

EXAMPLE Perform a PUT to the new container object URI:

```
PUT /MyContainer/ HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-container
```

```
Content-Type: application/cdmi-container
X-CDMI-Specification-Version: 1.0.1

{
    "metadata": {
        "cdmi_data_redundancy": "2",
        "cdmi_immediate_redundancy": "true",
        "cdmi_RPO": "86400",
        "cdmi_RTO": "360"
    },
    "export": {
        "Network/NFSv4": {
            "identifier": "/users",
            "permissions": "domain"
        }
    }
}
```

The response looks like:

```
HTTP/1.1 201 Created
Content-Type: application/cdmi-container
X-CDMI-Specification-Version: 1.0.1

{
    "objectID": "AABwbQAQ7EacyeWGVRGqCA==",
    "parentURI": "/",
    "domainURI": "/cdmi_domains/MyDomain/",
    "capabilitiesURI": "/cdmi_capabilities/Container/",
    "completionStatus": "Complete",
    "metadata": {
        "cdmi_size": "20000000",
        "cdmi_data_redundancy": "2",
        "cdmi_immediate_redundancy": "true",
        "cdmi_RPO": "86400",
        "cdmi_RTO": "360"
        "cdmi_size_provided": "20000000",
        "cdmi_data_redundancy_provided": "2",
        "cdmi_immediate_redundancy_provided ": "true",
        "cdmi_RPO_provided": "86400",
        "cdmi_RTO_provided": "360"
    },
    "export": {
        "Network/NFSv4": {
            "identifier": "/users",
            "permissions": "domain"
        }
    },
    "childrenrange": "",
    "children": []
}
```

If the create request does not include metadata items for some of the values in this type of Container, the server shall do a “best match” to one of the preconfigured types and supply the corresponding values. Note that the “_provided” values have been set to the actual configuration.

5.3.2.2 Read a Container Object

Next, we will read the container object we just created.

EXAMPLE Perform a GET to the container object URI:

```
GET /MyContainer/ HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-container
X-CDMI-Specification-Version: 1.0.1
```

The response looks like:

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-container
X-CDMI-Specification-Version: 1.0.1

{
    "objectID": "AABwbQAQ7EacyeWGVRGqCA==",
    "parentURI": "/",
    "domainURI": "/cdmi_domains/MyDomain/",
    "capabilitiesURI": "/cdmi_capabilities/Container/silver",
    "completionStatus": "Complete",
    "metadata": {
        "cdmi_size": "20000000",
        "cdmi_data_redundancy": "2",
        "cdmi_immediate_redundancy": "true",
        "cdmi_RPO": "86400",
        "cdmi_RTO": "360"
        "cdmi_size_provided": "20000000",
        "cdmi_data_redundancy_provided": "2",
        "cdmi_immediate_redundancy_provided ": "true",
        "cdmi_RPO_provided": "86400",
        "cdmi_RTO_provided": "360"
    },
    "export": {
        "Network/NFSv4": {
            "identifier": "/users",
            "permissions": "domain"
        }
    },
    "childrenrange": "",
    "children": []
}
```

5.3.2.3 Update a Container Object

Next, we will update the container object we just read to add an additional export.

EXAMPLE Perform a PUT to the new container object URI:

```
PUT /MyContainer/ HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-container
Content-Type: application/cdmi-container
X-CDMI-Specification-Version: 1.0.1

{
```

```
    "metadata": {
        "cdmi_data_redundancy": "2",
        "cdmi_immediate_redundancy": "true",
        "cdmi_RPO": "86400",
        "cdmi_RTO": "60"
        "cdmi_size_provided": "20000000",
        "cdmi_data_redundancy_provided": "2",
        "cdmi_immediate_redundancy_provided ": "true",
        "cdmi_RPO_provided": "86400",
        "cdmi_RTO_provided": "360"
    },
    "export": {
        "Network/NFSv4": {
            "identifier": "/users",
            "permissions": "domain"
        },
        "OCCI/NFSv4": {
            "identifier":
"AAAAFAAo7EFMb3JlbSBpcHN1bSBkb2xvciBzaXQgYW1ldCBhbWV0Lg==",
            "permissions": "f63aaa26-30b7-4a30-91ca-1d03c1e52214"
        }
    }
}
```

The response looks like:

```
HTTP/1.1 204 No Content
```

5.3.2.4 Delete a Container Object

Finally, we will delete the container object and return the storage to the pool.

EXAMPLE Perform a DELETE to the container object URI:

```
DELETE /MyContainer/ HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-container
Content-Type: application/cdmi-container
X-CDMI-Specification-Version: 1.0.1
```

The response looks like:

```
HTTP/1.1 204 No Content
```