Cloud Data Management Interface

Version 1.0.1h

Publication of this Working Draft for review and comment has been approved by the CDMI TWG. This draft represents a “best effort” attempt by the CDMI TWG 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://www.snia.org/feedback/.

Working Draft

March 30, 2011

Revision History

Version	Date	Originator	Comments
1.0	4/12/10		Released as a SNIA Technical Position.
1.0a	6/8/10	M. McMinn	Incorporated Trac tickets 79, 82, 87, 92, 93, 95, 96, 116-122, 128, 130, 134, 135, 137, 138, 142, 143, 147, 149, 152, and 153.
1.0b	6/19/10	M. McMinn	Incorporated Trac tickets 158, 161, 162-167, 169-171
1.0c	8/17/10	M. McMinn	Incorporated Trac tickets 146, 154, 155, 159, 174-184, 206-209, 81, 210-221, and 223-227.
1.0d	9/13/10	M. McMinn	Incorporated Trac tickets 39, 124, 144, 145, 218, 219, 231, 235, 237, 238, 240, 245, and 246. Added Krishna Sankar to Acknowledgements section. Changed “SNIA Technical Position” to “Working Draft”.
1.0e	11/03/10	M. McMinn	Incorporated Trac tickets 234, 249, 250, 252-254, and 257. Released as an” Internal Use Draft” for the San Jose Face-to-Face meeting.
1.0.1f	12/6/10	M. McMinn	Updated scope, references, changed “must” to “shall,” and added cross-references. Incorporated Trac tickets 156, 157, 172, 233, 236, 239, 241-244, 247, 248, 251, 252, 255, 259, 260, 263-267, 272, 273, 276, 283, and 285. Performed consistency edits.
1.0.1g	1/21/11	M. McMinn	Incorporated Trac tickets 287, 288, 289, 290, & 306.
1.0.1h	3/29/11	M. McMinn	Incorporated Trac tickets 277, 279, 280, 287, 291-294, 296, 298-300, 305, 307, 310-311, 313, 315, 317-318, 320, 322, 324-329, 331, 337-343, 345, 351, 354-391, and 393-405. Performed consistency edits. Released as Internal Use Draft.
1.0.1h	3/30/11	M. McMinn	Released as a Working Draft.

Contents

Figures

Tables

Foreword

SNIA Web Site

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

SNIA Address

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent to the Storage Networking Industry Association, 425 Market Street, Suite #1020, San Francisco, CA 94105, U.S.A.

Acknowledgements

The SNIA Cloud Storage Technical Working Group, who developed this standard, would like to recognize the significant contributions made by the following members:

Organization Represented	Name of Representative
Cisco Systems	Krishna Sankar
Cisco Systems	Mike Siefer
Hitachi Data Systems	Eric Hibbard
Individual	Rich Ramos
Iron Mountain	Chris Schwarzer
Netapp Inc.	Alan Yoder
NetApp Inc.	David Slik
NetApp Inc.	Lakshmi N. Bairavasundaram
Olocity	Scott Baker
Oracle	Mark Carlson
QLogic	Hue Nguyen

Introduction

Purpose and Audience

This document tells you how to access cloud storage and to manage the data stored there. The intended audience is application developers who are implementing or using cloud storage.

Organization

Table 1 describes the contents of each chapter in this document.

Table 1 - Chapter Contents

Chapter	Contents
1 - Scope	Defines the scope of this document.
2 - References	Lists the documents referenced in 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 - Overview of Cloud Storage	Provides a brief overview of cloud storage and details the philosophy behind the standard as a model for the operations.
6 - Getting Started	Gives an example of the resources that may be accessed and the representations used to modify them.
7 - Interface Specification	Provides a description of HTTP status codes, Cloud Data Management Interface (CDMI) object types, object references, and object manipulations.
8 - Data Object Resource Operations	Provides the normative standard for data object resource operations.
9 - Container Object Resource Operations	Provides the normative standard of container object resource operations.
10 - Domain Object Resource Operations	Provides the normative standard of domain object resource operations.
11 - Queue Object Resource Operations	Provides the normative standard of queue object resource operations.
12 - Capability Object Resource Operations	Provides the normative standard of capability object resource operations.
13 - Exported Protocols	Discusses how virtual machines in the cloud computing environment may use the exported protocols from CDMI containers.
14 - Snapshots	Discusses how snapshots are accessed under CDMI containers.
15 - Serialization/Deserialization	Discusses serialization and deserialization, including import and export of serialized data under CDMI.
16 - Metadata	Provides the normative standard of the metadata used in the interface.

Table 1 - Chapter Contents

Chapter	Contents
17 - CDMI Logging	Describes CDMI functional logging for object functions, security events, data management events, and queues.
18 - Retention and Hold Management	Describes the optional retention management disciplines to be implemented into the system management functions.
19 - Notification Queues	Describes how CDMI clients may efficiently discover what changes have occurred to the system.
20 - Query Queues	Describes how CDMI clients may efficiently discover what content matches a given set of metadata query criteria or full-content search criteria.
Annex A - (normative) Transport Security	Provides normative text for securing the HTTP communications protocol for transferring CDMI messages.
Annex B - (informative) Extending the Interface	Provides informative guidelines for extending the interface for data object versioning.

1 Scope

This International Standard specifies the interface to access cloud storage and to manage the data stored therein. This International Standard is applicable to developers who are implementing or using cloud storage.

2 References

2.1 Normative References

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

[ISO-8601]

International Standards Organization, "Data elements and interchange formats -- Information interchange -- Representation of dates and times", ISO 8601:20044 - http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=40874

[ITU-T509]

International Telecommunications Union Telecommunication Standardization Sector (ITU-T), Recommendation X.509: Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks, May 2000. Specification and technical corrigenda - http://www.itu.int/ITU-T/publications/recs.html

[POSIX ERE]

The Open Group, Base Specifications Issue 6, IEEE Std 1003.1, 2004 Edition - http://www.unix.org/version3/ieee_std.html

[RFC2045]

IETF RFC 2045. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies - http://www.ietf.org/rfc/rfc2045.txt

[RFC2246]

IETF RFC 2246. The TLS Protocol Version 1.0 - http://tools.ietf.org/rfc/rfc2246.txt

[RFC2578]

IETF RFC 2578. Structure of Management Information Version 2 (SMIv2) - http://www.ietf.org/rfc/rfc2578.txt

[RFC2616]

IETF RFC 2616. Hypertext Transfer Protocol -- HTTP/1.1 - http://www.ietf.org/rfc/rfc2616.txt

[RFC3280]

IETF RFC 3280. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile - http://www.ietf.org/rfc/rfc3280.txt

[RFC3530]

IETF RFC 3530. Network File System (NFS) version 4 Protocol - http://www.ietf.org/rfc/rfc3530.txt

[RFC3986]

IETF RFC 3986. Uniform Resource Identifier (URI): Generic Syntax - http://www.rfc-editor.org/rfc/rfc3986.txt

[RFC4346]

IETF RFC 4346. The Transport Layer Security (TLS) Protocol Version 1.1 - http://tools.ietf.org/rfc/rfc4346.txt

[RFC4627]

IETF RFC 4627. The application/json Media Type for JavaScript Object Notation (JSON) - http://www.ietf.org/rfc/rfc4627.txt

[RFC4918]

HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV) - http://www.ietf.org/rfc/rfc4918.txt

[RFC5246]

IETF RFC 5246. The Transport Layer Security (TLS) Protocol Version 1.2 - http://tools.ietf.org/rfc/rfc5246.txt

2.2 Informative References

[CRC]

Williams, Ross, “A Painless Guide to CRC Error Detection Algorithms”, Chapter 16, August 1993, http://www.repairfaq.org/filipg/LINK/F_crc_v3.html

[PKS12]

RSA Laboratories, PKCS #12: Personal Information Exchange Syntax, Version 1.0, June 1999. Specification and Technical Corrigendum - http://www.rsa.com:80/rsalabs/node.asp?id=2138

[REST]

"Representational State Transfer” - http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[RESTful Web]

Richardson, Leonard and Sam Ruby, *RESTful Web Services*, O'Reilly, 2007.

[SIRDM]

Storage Industry Resource Domain Model - http://www.snia.org/education/storage_networking_primer/sirdm/

3 Terms

For the purposes of this document, the following terms and definitions apply.

Note: All of these terms and definitions are taken from the SNIA Dictionary, except for the following: CRC, CRUD, object, object identifier, OCCI, and Uniform Resource Identifier.

3.1
Access Control List
ACL
a persistent list, commonly composed group of Access Control Entries (ACEs), that enumerates the rights of principals (users and groups of users and/or groups) to access resources

3.2
CDMI
Cloud Data Management Interface

3.3
CRC
cyclic redundancy check

3.4
CRUD
create, retrieve, update, delete

3.5
Data storage as a Service
DaaS
cloud storage
delivery over a network of appropriately configured virtual storage and related data services, based on a request for a given service level

delivery of virtualized storage and data services on demand over a network

Typically, DaaS hides limits to scalability, is either self-provisioned or provisionless, and is billed based on consumption.

3.6
domain
shared user authorization database that contains users, groups, and their security policies

Each CDMI object belongs to a single domain, and each domain provides user mapping and accounting information.

3.7
Infrastructure as a Service
IaaS
delivery over a network of an appropriately configured virtual computing environment, based on a request for a given service level

Typically, IaaS is either self-provisioned or provisionless and is billed based on consumption.

3.8
iSCSI
Internet Small Computer Systems Interface

3.9
LUN
Logical Unit Number

3.10
MIME
Multipurpose Internet Mail Extensions

3.11
NFS
Network File System

3.12
object identifier
OID
globally unique object identifier (OID) assigned at creation time for every object stored within a CDMI-compliant system

3.13
object
an entity that has an object identifier (OID), a unique URI, and contains state

Types of CDMI objects include data objects, containers, capabilities, domains, and queues.

3.14
OCCI
Open Cloud Computing Interface

3.15
Platform as a Service
PaaS
delivery over a network of a virtualized programming environment, consisting of an application deployment stack based on a virtual computing environment

Typically, PaaS is based on IaaS, is either self-provisioned or provisionless, and is billed based on consumption.

3.16
private cloud
delivery of SaaS, PaaS, IaaS, and/or DaaS to a restricted set of customers, usually within a single organization

Private clouds are created due to issues of trust.

3.17
public cloud
delivery of SaaS, PaaS, IaaS, and/or DaaS to a relatively unrestricted set of customers

3.18
Representational State Transfer
specific set of principles for defining, addressing, and interacting with resources addressable by URIs

Architectures that follow these principles are said to be RESTful. The principles include abstraction of state into resources and a uniform set of representations and operations (e.g., HTTP verbs like GET and PUT as the only means to manipulate a resource). RESTful interfaces are contrasted with Web Services interfaces such as WBEM, which tend to be RPC-like.

3.19
REST
Representational State Transfer

3.20
Software as a Service
SaaS
delivery over a network, on demand, of the use of an application

3.21
Uniform Resource Identifier
URI
compact sequence of characters that identifies an abstract or physical resource

3.22
XAM
eXtensible Access Method

4 Conventions

4.1 Interface Format

Each interface description has eight sections, as described in Table 2.

Table 2 - Interface Format Descriptions

Section	Description
Synopsis	The GET, PUT, and POST semantics.
Capabilities	A description of the supported operations.
Request Headers	The request headers, such as Accept, Authorization, Content-Length, Content-Type, X-Cloud-Client-Specification-Version.
Request Message Body	A description of the message body contents.
Response Headers	The response headers, such as Content-Length, Content-Type.
Response Message Body	A description of the message body contents.
Response Status	A list of codes.
Example	An example of the operation.

4.2 Typographical Conventions

The typographical conventions used in this document are described in Table 3.

Table 3 - Typographical Conventions

Convention	Description
`Fixed-width text`	The names of commands and on-screen computer output.
`Bold, fixed-width text`	What you type, contrasted with on-screen computer output.
Italicized text	Variables, field names, and book titles.
Note:	Additional or useful informative text.
WARNING:	Indicates that you should pay careful attention to the probable action, so that you may avoid system failure or harm.

5 Overview of Cloud Storage

5.1 Introduction

When discussing cloud storage and standards, it is important to distinguish the various resources that are being offered as services. These resources are exposed to clients as functional interfaces (data paths) and are managed by management interfaces (control paths). This standard explores the various types of interfaces that are part of offerings today and shows how they are related. This standard proposes a model for the interfaces that may be mapped to the various offerings and a model that forms the basis for rich cloud storage interfaces into the future.

Another important concept in this standard is that of metadata. When managing large amounts of data with differing requirements, metadata is a convenient mechanism to express those requirements in such a way that underlying data services may differentiate their treatment of the data to meet those requirements.

The appeal of cloud storage is due to some of the same attributes that define other cloud services: pay as you go, the illusion of infinite capacity (elasticity), and the simplicity of use/management. It is therefore important that any interface for cloud storage support these attributes, while allowing for a multitude of business cases and offerings long into the future.

5.2 What is Cloud Storage?

The use of the term cloud in describing these new models arose from architecture drawings that typically used a cloud as the dominant networking icon. The cloud conceptually represents any-to-any connectivity in a network. The cloud also conceptually represents an abstraction of concerns, such that the actual connectivity and the services running in the network that accomplish that connectivity do so with little manual intervention.

This abstraction of complexity and promotion of simplicity is what primarily constitutes a cloud of resources, regardless of type. An important part of the cloud model, in general, is the concept of a pool of resources that is drawn from, on demand, in small increments (smaller than what you would typically purchase by buying equipment). The relatively recent innovation that has made this possible is virtualization.

Thus, cloud storage is simply the delivery of virtualized storage on demand. The formal term that this standard proposes for this is Data storage as a Service (DaaS). DaaS means “delivery over a network of appropriately configured virtual storage and related data services, based on a request for a given service level.”

5.3 Data Storage as a Service

By abstracting data storage behind a set of service interfaces and delivering it on demand, a wide range of actual offerings and implementations are possible. The only type of storage that is excluded from this definition is that which is delivered, not based on demand, but on fixed capacity increments.

An important part of any DaaS offering is the support of legacy clients. Support is accommodated with existing standard protocols such as iSCSI (and others) for block and CIFS/NFS or WebDAV for file network storage, as shown in Figure 1, "Existing Data Storage Interface Standards".

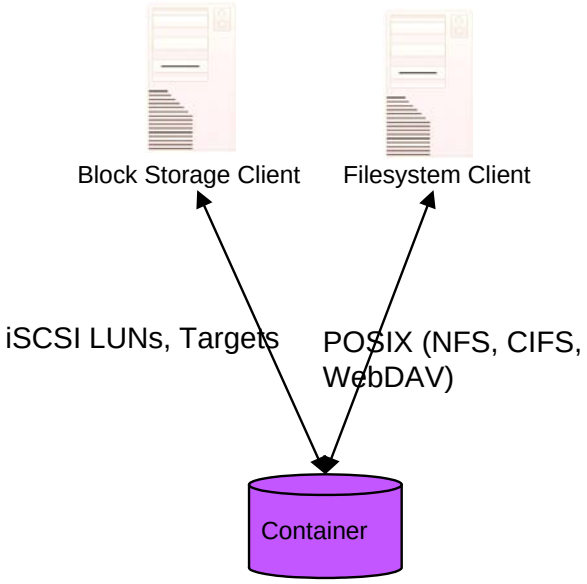


Figure 1 - Existing Data Storage Interface Standards

The difference between the purchase of a dedicated appliance and that of cloud storage is not the functional interface, but merely the fact that the storage is delivered on demand. The customer pays for either what they actually use, or in other cases, what they have allocated for use. In the case of block storage, a Logical Unit Number (LUN), or virtual volume, is the granularity of allocation. For file protocols, a file system is the unit of granularity. In either case, the actual storage space may be thin provisioned and billed for, based on actual usage. Data services, such as compression and deduplication, may be used to further reduce the actual space consumed.

Managing this storage is typically done out of band of these standard data storage interfaces, either through an API, or more commonly, through an administrative browser-based user interface. This interface may be used to invoke other data services as well, such as snapshot and cloning.

In this model, the underlying storage space is abstracted and exposed by these interfaces using the notion of a container. A container is not only a useful abstraction for storage space, but also serves as a grouping of the data stored in it and a point of control for applying data services in the aggregate.

Another type of DaaS offering is one of simple table space storage, allowing for horizontal scaling of database-like operations that certain applications need. Rather than virtualizing relational database instances, table space storage offers a new data storage interface of limited functionality, with the emphasis on scalability rather than features. Scalability allows the tables to be partitioned across multiple nodes based on common key values, affording horizontal scalability at the expense of functions that may typically only be implemented by a vertically-scaled relational database.

A great deal of innovation and change is happening in these interfaces, and each offering has its own unique proprietary interface, as shown in Figure 2, "Storage Interfaces for Database/Table Data".

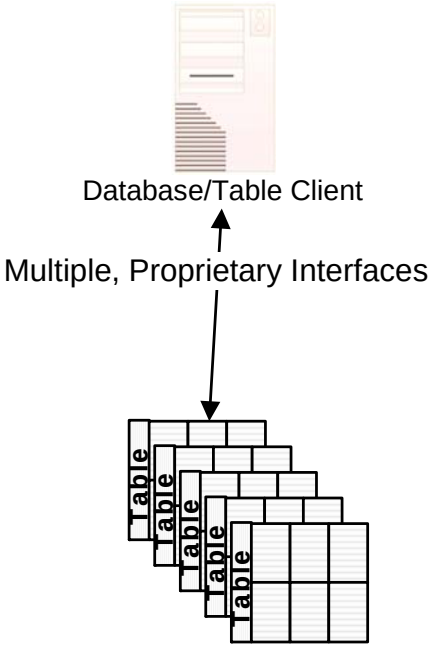


Figure 2 - Storage Interfaces for Database/Table Data

Due to the rapid innovation in this space, it is probably best to wait for further development of this type of cloud storage before trying to standardize a functional interface for this type of storage.

A third category of functional interface for data storage has emerged. This type of interface treats every data object as accessible via a unique URI. It may then be fetched using the standard HTTP protocol, and a browser may be used to invoke the appropriate application to deal with the data.

Each data object is created, retrieved, updated, and deleted (CRUD semantics) as a separate resource. In this type of interface, a container, if used, is a simple grouping of data objects for convenience. Nothing prevents the concept of containers, in this case, from being hierarchical, although any given

implementation might support only a single level of such. This type of container is called a “soft” container, as shown in Figure 3, "Storage Interfaces for Object Storage Client Data".

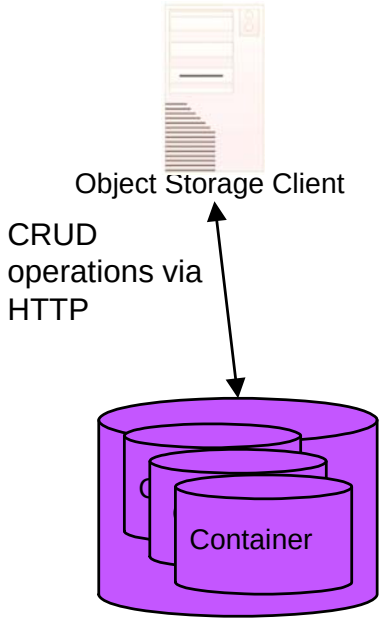


Figure 3 - Storage Interfaces for Object Storage Client Data

While there are several proprietary examples of this type of interface, they all pretty much support the same set of operations. This, then, is an area ripe for standardization.

5.4 Data Management in the Cloud

Many of the initial offerings of cloud storage focused on a kind of “best effort” quality of storage service, with very little offering of additional data services for that data. To address the needs of enterprise applications with cloud storage, however, there is increasing pressure to offer better quality of service and the deployment of additional data services.

The danger, of course, is that cloud storage loses its benefit of simplicity and the abstraction of complexity, as additional data services are applied and the implication that these services need to be managed. One can hardly have cloud storage customers setting up backup schedules through dedicated user interfaces, deploying data services individually for their data elements, and so on.

Fortunately, the SNIA Storage Industry Resource Domain Model (see Figure 4) [SIRDM] gives us a way to minimize this complexity and address the need for cloud storage to remain simple. By using the different types of metadata discussed in that model for a cloud storage interface, an interface may be created that

allows offerings to meet the requirements of the data without adding undue complexity to the management of that data.

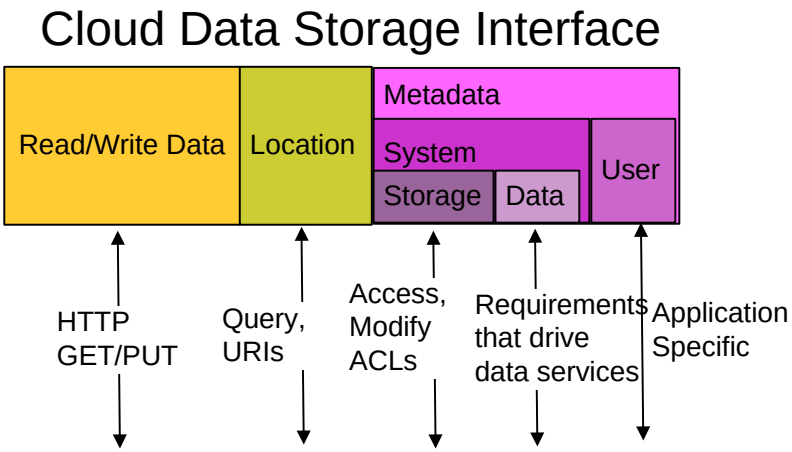


Figure 4 - Using the Resource Domain Model

By supporting metadata in a cloud storage interface standard and prescribing how the storage system and data system metadata is interpreted to meet the requirements of the data, the simplicity required by the cloud storage paradigm may be maintained and may still address the requirements of enterprise applications and their data.

User metadata is retained by the cloud and may be used to find the data objects and containers by doing a query for specific metadata values. The schema for this metadata may be determined by each application, domain, or the user. For more information on support for user metadata, see Section 16.2.

Storage system metadata is produced and interpreted by the cloud offering or basic storage functions, such as modification and access statistics, and for governing access control. For more information on support for storage system metadata, see Chapter 16, "Metadata".

Data system metadata is interpreted by the cloud offering as data requirements that drive the operation of underlying data services for that data. It may apply to an aggregation of data objects in a container or even to individual data objects, if the offering supports this level of granularity. For more information on support for data system metadata, see Section 16.4.

5.5 Data and Container Management

There is no reason that managing data and managing containers should involve different paradigms. Therefore, this standard proposes that the use of metadata be extended from applying to individual data elements to applying to containers of data as well. Thus, any data placed into a container essentially inherits the data system metadata of the container into which it was placed. When creating a new container within an existing container, the new container would similarly inherit the metadata settings of its parent's data system. Of course, the data system metadata may be overridden at the container or individual data element level, as desired.

Even if the functional interface that the offering provides does not support setting this type of metadata on individual data elements, it may still be applied to the containers, even though it may not be able to be overridden on the basis of individual data elements through that interface. For file-based interfaces that support extended attributes (i.e., CIFS, NFSv4), these extended attributes may be used to specify the data system metadata to override that specified for the container through these existing standard interfaces.

The mapping of extended attribute names and values to individual file data requirements as supported by cloud storage will be done as a follow-on effort.

5.6 Reference Model for Cloud Storage Interfaces

By putting all of these elements together, a reference model is created, as shown in Figure 5, "Cloud Storage Reference Model":

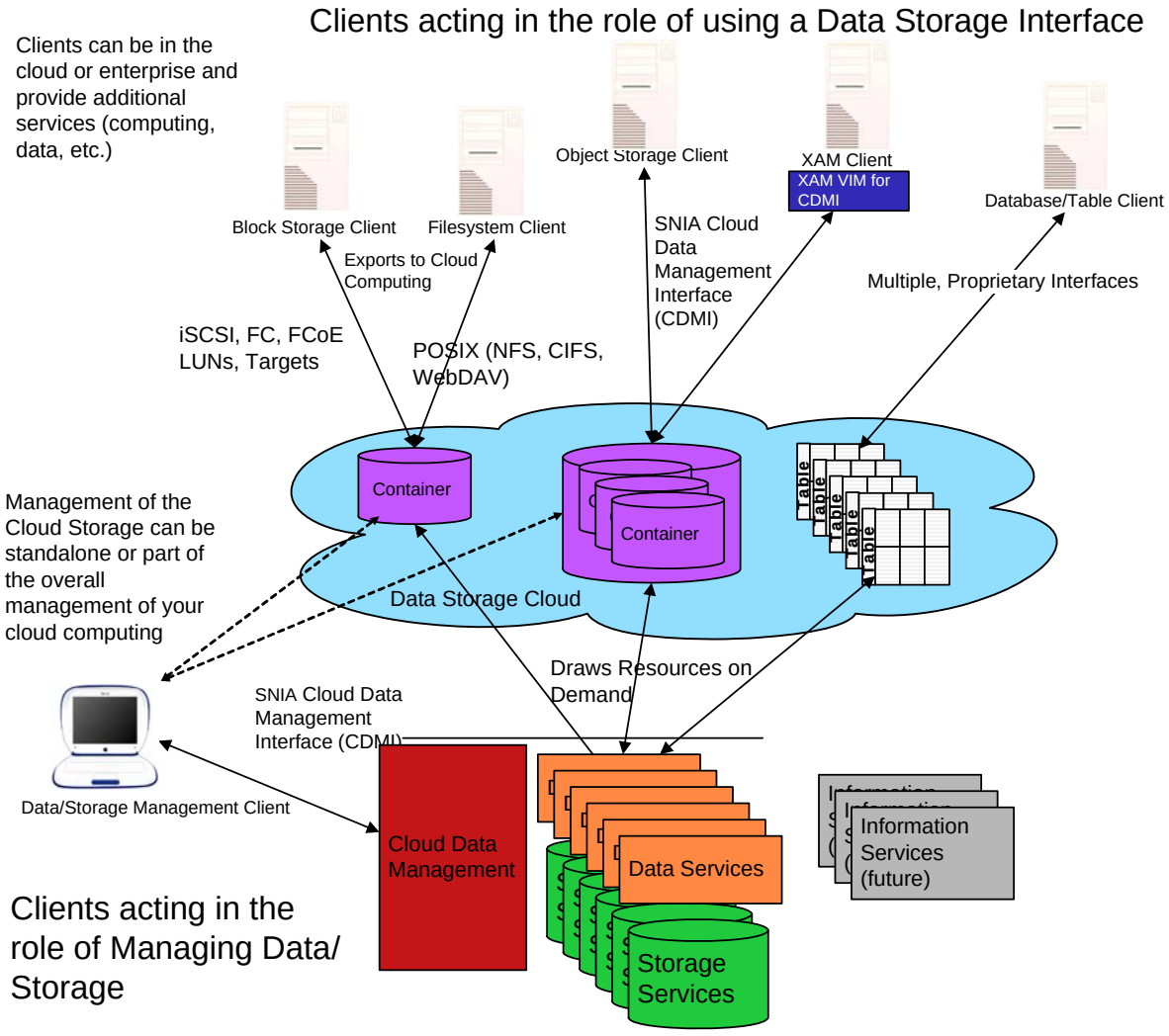


Figure 5 - Cloud Storage Reference Model

This model shows multiple types of cloud data storage interfaces that are able to support both legacy and new applications. All of the interfaces allow storage to be provided on demand, drawn from a pool of resources. The capacity is drawn from a pool of storage capacity provided by storage services. The data services are applied to individual data elements, as determined by the data system metadata. Metadata specifies the data requirements on the basis of individual data elements or on groups of data elements (containers).

5.7 SNIA Cloud Data Management Interface

As shown in Figure 5, "Cloud Storage Reference Model", the SNIA Cloud Data Management Interface (CDMI) is the functional interface that applications may use to create, retrieve, update, and delete data elements from the cloud. As part of this interface, the client will be able to discover the capabilities of the cloud storage offering and to use this interface to manage containers and the data that is placed in them. In addition, data system metadata may be set on containers and their contained data elements through this interface.

The majority of existing cloud storage offerings today will likely be able to implement the interface. They may implement the interface with an adapter to their existing proprietary interface, or they may implement the interface directly, side by side with their existing interfaces. In addition, existing client libraries, such as XAM, may be adapted to this interface, as shown in Figure 5.

This interface may also be used by administrative and management applications to manage containers, domains, security access, and monitoring/billing information, even for storage that is functionally accessible by legacy or proprietary protocols. The capabilities of the underlying storage and data services are exposed so that clients may understand the offering.

Conformant cloud offerings may offer a subset of the CDMI interface, as long as they expose the limitations in the capabilities part of the interface.

The CDMI standard uses RESTful principles in the interface design where possible [REST]. Typically, cloud implementations will offer a subset of CDMI and describe what is implemented via the capabilities. For more information on the REST principles, please see [RESTful Web].

5.8 Object Model for CDMI

The model behind the Cloud Data Management Interface is shown in Figure 6, "CDMI Interface Model".

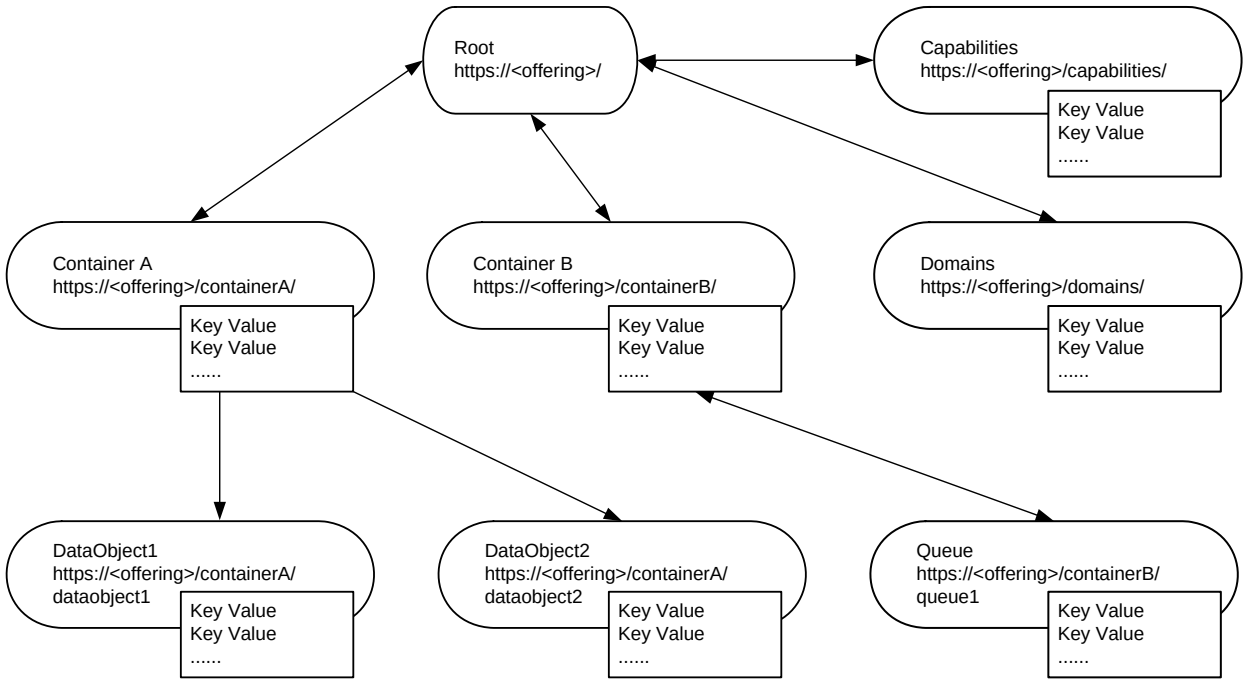


Figure 6 - CDMI Interface Model

For data storage operations, the client of the interface only needs to know about container objects and data objects. All data path implementations are required to support at least one level of containers, a sort of grouping of data objects. As shown in Figure 6, the client may do a PUT to the container URI and create a new container with the specified name. The KEY/VALUE metadata is optional. Once a container is created, a client may do a PUT to create a data object URI. A subsequent GET will fetch the actual data object and its value. The only metadata KEY/VALUE required on the data object PUT is content type (MIME). Other KEY/VALUE pairs may be used to specify the data requirements at the object level. This metadata is defined in the CDMI standard.

CDMI also defines an object, called a queue, which has special properties for in-order, first in, first-out creation and fetching of queue objects, similar to a container of data objects. More information on queues may be found in Chapter 11, "Queue Object Resource Operations".

The CDMI does not need to be used as the data path, and it applies to cloud storage that is exposed as either standard or proprietary interfaces. In this case, the client might stop at creating the container. The metadata is also used to configure the data requirements of the storage under the exported protocol that the container exposes, such as a block protocol or a file protocol. While many implementations may use an underlying file to store data for a block protocol (such as iSCSI), in the CDMI interface, the container is used as the abstraction for applying the data system metadata for this data and for attaching the structures that govern the exported protocols. This maps to the concept of a block storage location (Target, LUN) being a container of data objects, each a block in size and located by their Logical Block Address.

A cloud offering may also support domains, which allow administrative ownership to be associated with stored objects. Domains allow the standard to determine how user credentials are mapped to principles used in an Access Control List (ACL), allow granting of special cloud-related privileges, and allow delegation to external user authorization systems, such as LDAP or Active Directory. Domains may also be hierarchical, allowing for corporate domains with multiple children domains for departments or individuals. The domain concept is also used to aggregate usage data that is used to bill, meter, and monitor cloud use.

Finally, a capabilities resource and associated URI allows a client to discover the capabilities of the offering and its implementation of CDMI. The interface requires this resource, but static pages that list exactly what is implemented may satisfy this requirement.

5.9 CDMI Metadata

CDMI uses many different types of metadata, including HTTP metadata, data system metadata, user metadata, and storage system metadata.

HTTP metadata is metadata that is related to the use of the HTTP protocol, such as content-size, content-type, etc. This type of metadata is not specifically related to the CDMI standard, but needs to be discussed to explain how CDMI uses the HTTP standard.

Data system metadata is metadata that is specified by a CDMI client and attached to a container or data object, abstractly specifying the data requirements that are then supplied by data services that are deployed in the cloud storage system. The data system metadata settings are treated as goals. In some cases, actual measurements toward these goals are specified.

User metadata is arbitrarily-defined JSON strings that are specified by the CDMI client and attached to objects. The namespace used for user metadata is self-administered (such as using the reverse domain name) and restricted to not beginning with the prefix “cdmi_”.

Storage system metadata is read-only metadata that is generated by the storage services in the system to provide useful information to a CDMI client. Examples include ACLs, creation time, etc.

Table 4 - Creation/Consumption of Storage System Metadata

	Created by User	Created By System
Consumed by User	User metadata.	Storage system metadata.
Consumed by System	Data system metadata.	N/A.

5.10 Object ID

Every object stored within a CDMI-compliant system will have a globally unique object identifier (OID) assigned at creation time. The CDMI object ID is a string but has rules for how it is generated and how it obtains its uniqueness. Each offering that implements CDMI is able to produce these identifiers without conflicting with other offerings.

Every CDMI system shall allow access to stored objects, by object ID, by allowing the object ID to be appended to the root container URI. For example, for the following root container URI, the second URI would provide access to objects by object ID:

```
http://cloud.example.com/root
http://cloud.example.com/root/cdmi_objectid/<objectID>
```

5.11 CDMI Object ID Format

The offering shall create the Object ID, which identifies an object. The Object ID shall be globally unique and shall conform to the format defined in Figure 7, "Object ID Format". The native format of an Object ID is a variable-length byte sequence and shall be a maximum length of 40 bytes. An application should treat Object IDs as opaque byte strings. However, the Object ID format is defined such that its integrity may be validated, and independent offerings may assign unique Object ID values independently.

0	1	2	3	4	5	6	7	8	9	10	...	38	39
Reserved (zero)	Enterprise Number			Reserved (zero)	Length	CRC		Opaque Data					

Figure 7 - Object ID Format

As shown in Figure 7,

- The reserved bytes shall be set to zero.
- The *Enterprise Number* field shall be the SNMP enterprise number of the offering organization that created the Object ID, in network byte order. See [RFC2578] and http://www.iana.org/assignments/enterprise-numbers. 0 is a reserved value.
- The 5th byte shall contain the full length of the Object ID, in bytes.
- The *CRC* field shall contain a 2-byte (16-bit) CRC in network byte order. The *CRC* field enables the Object ID to be validated for integrity. The *CRC* field shall be generated by running the algorithm

[CRC] across all bytes of the Object ID, as defined by the *Length* field, with the *CRC* field set to zero. The CRC function shall have the following parameters:

— Name : "CRC-16"

— Width : 16

— Poly : 0x8005

— Init : 0x0000

— RefIn : True

— RefOut : True

— XorOut : 0x0000

— Check : 0xBB3D

This function defines a 16-bit CRC with polynomial 0x8005, reflected input, and reflected output. This CRC-16 is specified in [CRC].

- Opaque data in each Object ID shall be unique for a given Enterprise Number.
- The native format for an Object ID is binary. When necessary, such as when included in JSON strings, the Object ID textual representation shall be hex-encoded and case insensitive.

5.12 Security

Security, in the context of CDMI, refers to the protective measures employed in managing and accessing data and storage. The specific objectives to be addressed by security include:

- Provide a mechanism that assures that the communications between a CDMI client and server may not be read or modified by a third party
- Provide a mechanism that allows CDMI clients and servers to provide an assurance of their identity
- Provide a mechanism that allows control of the actions a CDMI client is permitted to perform on a CDMI server
- Provide a mechanism for records to be generated for actions performed by a CDMI client on a CDMI server
- Provide mechanisms to protect data at rest
- Provide a mechanism to eliminate data in a controlled manner
- Provide mechanisms to discover the security capabilities of a particular implementation

Security measures within CDMI may be summarized as transport security, user and entity authentication, authorization and access controls, data integrity, data and media sanitization, data retention, protections against malware, data at-rest encryption, and security capability queries. With the exception of both the transport security and the security capability queries, which are mandatory to implement, the security measures may vary significantly from implementation to implementation.

When security is a concern, the CDMI client should begin with a series of security capability queries to determine the exact nature of the security features that are available. (See Section 12.1.1, "Cloud Storage

System-Wide Capabilities".) Based on responses from the CDMI implementation, a risk-based decision should be made as to whether the CDMI resource should be used. This is particularly true when the data to be stored in the cloud storage is sensitive or regulated, such that it shall be protected (for example, encrypted) or handled in a particular manner (for example, full accountability and traceability of management and access).

HTTP is the mandatory transport mechanism for this version of CDMI, and HTTP over TLS (HTTPS) is the mechanism used to secure the communications between CDMI entities. To ensure both security and interoperability, all CDMI implementations are required to implement the Transport Layer Security (TLS) protocol as described in Annex A, but its use by CDMI entities is optional (see Annex A, "(normative) Transport Security").

Additional details associated with the security capabilities are found in other sections of this document.

5.13 Required HTTP Support

A conformant implementation of CDMI shall also be a conformant implementation of [RFC2616] (HTTP 1.1). This standard lists the sections of [RFC2616] that shall be supported.

5.13.1 Content-Type Negotiation

A client may optionally supply an Accept header. If the content type of the requested resource is not present in the header, the server shall return a `406 Not Acceptable` status code. (See Section 12 of [RFC2616].)

5.13.2 Range Support

The server shall support Range headers and partial content responses (see Section 14.16 of [RFC2616]).

5.14 Time Representations

All date/time values, unless otherwise specified, are in the [ISO-8601] extended representation (YYYY-MM-DDThh:mm:ss.ssssssZ). The full precision must be specified, the sub-second separator must be a ".", the Z UTC zone indicator must be included, and all timestamps must be in UTC time zone. The YYYY-MM-DDT24:00:00.000000Z hour must not be used and represented as YYYY-MM-DDT00:00:00.000000Z.

All time durations, unless otherwise specified, are in the [ISO-8601] start date/end date representation (YYYY-MM-DDThh:mm:ss.ssssssZ/YYYY-MM-DDThh:mm:ss.ssssssZ). The end-date must be equal to or later than the start-date. The full precision must be specified, the sub-second separator must be a ".", the Z UTC zone indicator must be included, and all timestamps must be in UTC time zone. The YYYY-MM-DDT24:00:00.000000Z hour must not be used and represented as YYYY-MM-DDT00:00:00.000000Z.

6 Common Operations

The example transactions in this chapter illustrate some of the more common CDMI operations.

6.1 Discover the Capabilities of a Cloud Storage Provider

Perform a GET to the capabilities URI:

```
GET /cdmi_capabilities/ HTTP/1.1
Host: cloud.example.com
X-CDMI-Specification-Version: 1.0
```

The response looks like:

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-capability
X-CDMI-Specification-Version: 1.0

{
    "objectURI" : "/cdmi_capabilities/",
    "objectID" : "0000706D00100C435125A61B4C289455",
    "objectName" : "cdmi_capabilities/",
    "parentURI" : "/",
    "capabilities" : {
        "cdmi_domains" : "true",
        "cdmi_export_nfs" : "true",
        "cdmi_export_webdav" : "true",
        "cdmi_export_iscsi" : "true",
        "cdmi_queues" : "true",
        "cdmi_notification" : "true",
        "cdmi_query" : "true",
        "cdmi_metadata_maxsize" : "4096",
        "cdmi_metadata_maxitems" : "1024",
        "cdmi_size" : "true",
        "cdmi_list_children" : "true",
        "cdmi_read_metadata" : "true",
        "cdmi_modify_metadata" : "true",
        "cdmi_create_container" : "true",
        "cdmi_delete_container" : "true"
    },
    "childrenrange" : "0-3",
    "children" : [
        "domain/",
        "container/",
        "dataobject/",
        "queue/"
    ]
}
```

6.2 Create a New Container

Perform a PUT to the new container 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

{
    "metadata" : {

    }
}
```

The response looks like:

```
HTTP/1.1 201 Created
Content-Type: application/cdmi-container
X-CDMI-Specification-Version: 1.0

{
    "objectURI" : "/MyContainer/",
    "objectID" : "0000706D0010D538DEEE8E38399E2815",
    "objectName" : "MyContainer/",
    "parentURI" : "/",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/Container/",
    "completionStatus" : "Complete",
    "metadata" : {
        "cdmi_size" : "0"
    },
    "childrenrange" : "",
    "children" : [

    ]
}
```

6.3 Create a Data Object in a Container

Perform a PUT to the new data object URI:

```
PUT /MyContainer/MyDataObject.txt HTTP/1.1/
Host: cloud.example.com
Accept: application/cdmi-object
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0

{
    "mimetype" : "text/plain",
    "metadata" : {

    },
    "value" : "Hello CDMI World!"
}
```

The response looks like:

```
HTTP/1.1 201 Created
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0

{
    "objectURI" : "/MyContainer/MyDataObject.txt",
    "objectID" : "0000706D0010734CE0BAEB29DD542B51",
    "objectName" : "MyDataObject.txt",
    "parentURI" : "/MyContainer/",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/DataObject/",
    "completionStatus" : "Complete",
    "mimetype" : "text/plain",
    "metadata" : {
        "cdmi_size" : "17"
    }
}
```

6.4 List the Contents of a Container

Perform a GET to the container URI:

```
GET /MyContainer/ HTTP/1.1
Host: cloud.example.com
Accept: */*
X-CDMI-Specification-Version: 1.0
```

The response looks like:

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-container
X-CDMI-Specification-Version: 1.0

{
    "objectURI" : "/MyContainer/",
    "objectID" : "0000706D0010D538DEEE8E38399E2815",
    "objectName" : "MyContainer/",
    "parentURI" : "/",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/Container/",
    "completionStatus" : "Complete",
    "metadata" : {
        "cdmi_size" : "83"
    },
    "childrenrange" : "0-0",
    "children" : [
        "MyDataObject.txt"
    ]
}
```

6.5 Read the Contents of a Data Object

GET from the data object URI:

```
GET /MyContainer/MyDataObject.txt HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi_object
X-CDMI-Specification-Version: 1.0
```

The response looks like:

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0

{
    "objectURI" : "/MyContainer/MyDataObject.txt",
    "objectID" : "0000706D0010734CE0BAEB29DD542B51",
    "objectName" : "MyDataObject.txt",
    "parentURI" : "/MyContainer/",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/DataObject/",
    "completionStatus" : "Complete",
    "mimetype" : "text/plain",
    "metadata" : {
        "cdmi_size" : "17"
    },
    "valuerange" : "0-16",
    "value" : "Hello CDMI World!"
}
```

6.6 Read Only the Value of a Data Object

Perform a GET to the data object URI:

```
GET /MyContainer/MyDataObject.txt HTTP/1.1
Host: cloud.example.com
```

The response looks like:

```
HTTP/1.1 200 OK
Content-Type: text/plain

Hello CDMI World!
```

6.7 Delete a Data Object

Perform a DELETE to the data object URI:

```
DELETE /MyContainer/MyDataObject.txt HTTP/1.1
Host: cloud.example.com
X-CDMI-Specification-Version: 1.0
```

The response looks like:

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

7 Interface Standard

7.1 HTTP Status Codes

HTTP status codes are used to convey the results of the RESTful operations and to follow the basic semantics of HTTP with minimal overloading, as described in each operation description below (see Table 5). Other status codes are not part of this standard and retain their original semantics from HTTP 1.1.

Table 5 - HTTP Status Codes

Status Code	HTTP Name	Used for
200	OK	Resource retrieved successfully.
201	Created	Resource created successfully.
202	Accepted	Long running operation accepted for processing.
204	No Content	Operation successful, no data.
400	Bad Request	Missing or invalid request contents.
401	Unauthorized	Invalid authentication/authorization credentials.
403	Forbidden	This user is not allowed to perform this request.
404	Not Found	Requested resource not found.
405	Method Not Allowed	Requested HTTP verb not allowed on this resource.
406	Not Acceptable	No content type may be produced at this URI that matches the request.
409	Conflict	The operation conflicts with a non-CDMI access protocol lock, or may cause a state transition error on the server.
500	Internal Server Error	An unexpected implementor-specific error.
501	Not Implemented	A CDMI operation or metadata value was attempted that is not implemented.

7.2 Types of Objects in the Model

The five types of resource objects in the model include data objects, container objects, domain objects, and capability objects (see Table 6). The Content-Type in any given operation is specific to each type of resource object.

Table 6 - Types of Objects in the Model

Object Type	Description
Data objects	Data objects have a value but have no children. For more information, see Chapter 8, "Data Object Resource Operations".
Container objects	Container objects may have child objects but have no value. Container objects may be exported via protocols other than CDMI for data path operations, but the associated value is not represented in container objects via the CDMI data path. For more information, see Chapter 9, "Container Object Resource Operations".

Table 6 - Types of Objects in the Model

Object Type	Description
Domain objects	Domain objects may have child objects but have no value. For more information, see Chapter 10, "Domain Object Resource Operations".
Queue objects	Queue objects have a value but have no children. For more information, see Chapter 11, "Queue Object Resource Operations".
Capability objects	Capability objects may have child objects but have no value. For more information, see Chapter 12, "Capability Object Resource Operations".

The HTTP verbs overloaded by CDMI for each of these objects varies by object type. Non-overloaded HTTP operations may also be allowed for certain objects.

7.3 Object References

Object references are URIs within the cloud storage namespace that point to another URI within the same or another cloud storage namespace. References are similar to soft links in a file system, and the cloud does not guarantee that the referenced URI will be valid after the time of creation.

References are visible as children in a container and are distinguished from non-references by a trailing "?" character added to the reference name. Performing an operation (with the exception of create or delete) to a reference URI will result in a `302 Found` HTTP redirect, with the "Location" HTTP header containing the redirect destination URI that was specified at the time the reference was created. The reference's destination URI may not be altered once a reference has been created.

To continue, when CDMI clients receive a `302 Found` redirect, they should retry the operation on the URI contained with the "Location" header.

A delete operation on a reference URI shall delete the reference.

Example 7-1

GET to a URI, where the URI is a reference:

```
GET /MyContainer/MyDataObject.txt HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
X-CDMI-Specification-Version: 1.0
```

The response looks like:

```
HTTP/1.1 302 Found
Location: http://cloud.example.com/MyContainer/MyOtherDataObject.txt
```

8 Data Object Resource Operations

8.1 Overview

Data objects are the fundamental storage component within CDMI and are analogous to files within a filesystem. Each data object has a set of well-defined fields that include a single value and optional metadata that is generated by the cloud storage system and specified by the cloud user.

Data objects are addressed in CDMI in one of the two following ways:

- `http://cloud.example.com/dataobject`
- `http://cloud.example.com/cdmi_objectid/0000706D0010B84FAD185C425D8B537E`

The first example addresses the data object by URI, and the second addresses the data object by object ID. Every data object has a single, globally-unique object identifier (OID) that remains constant for the life of the object. Each data object may also have one or more URI addresses that allow the object to be accessed.

Every data object has a parent object from which the data object inherits data system metadata that is not explicitly specified in the data object itself. For example, the "`budget.xls`" data object stored at "`http://cloud.example.com/finance/budget.xls`" would inherit data system metadata from its parent container, "`finance`".

Individual fields within a data object may be accessed by specifying the field name after a question mark "?" that is appended to the end of the data object URI. For example, the following URI would return just the *value* field in the response body:

```
http://cloud.example.com/dataobject?value
```

Specific ranges of the *value* field may be accessed by specifying a byte range after the *value* field name. For example, the following URI would return the first thousand bytes of the *value* field:

```
http://cloud.example.com/dataobject?value:0-1000
```

Byte ranges are specified as per Section 14.35.1 of [RFC2616].

A list of unique fields, separated by a semicolon ";" may be specified, allowing multiple fields to be accessed in a single request. For example, the following URI would return the *value* and *metadata* fields in the response body:

```
http://cloud.example.com/dataobject?value;metadata
```

8.1.1 Data Object Metadata

Data object metadata may also include arbitrary user-supplied metadata and data system metadata, as specified in Chapter 16, "Metadata".

8.1.2 Data Object Consistency

Writing to a data object is an atomic operation. If a client were to read an object simultaneously with a write to that same object, it shall get either the old version or the new version, but not a mix of both. Writes are also atomic in the face of errors. Multiple simultaneous writes that complete without errors shall be ordered by the timestamps on the returning responses, which is to say, by the timestamps placed on the responses by the server-side implementation (i.e., according to the principle of eventual consistency).

Reading from an uninitialized data object byte shall return zero. If a client performs a data object value read from a specific byte range that has not previously been written to, the byte value returned shall be zero.

Note: Conformant implementations only need to support this atomicity via the CDMI object interface and are free to provide other semantics through other interfaces to the data in the container.

8.1.3 Data Object Representations

The representations in this section are shown using JSON notation. A conforming implementation shall support the mandatory parameters and may support the optional parameters. The parameter fields may be specified or returned in any order. Both clients and servers shall support JSON representation.

8.2 Create a Data Object (CDMI Content Type)

Synopsis:

Creates a new data object at the specified URI.

```
PUT <root URI>/<ContainerName>/<DataObjectName>
```

- `<root URI>` is the path to the CDMI cloud.
- `<ContainerName>` is zero or more intermediate containers that already exist.
- `<DataObjectName>` is the name specified for the data object to be created.

Once created, the object may also be accessed at `<root URI>/cdmi_objectid/<objectID>`.

Delayed Completion of Create:

On a create operation for a data object, the server may return a response of `202 Accepted`. In this case, the object is in the process of being created. This response is particularly useful for long-running operations, for instance, copying a large data object from a source URI. Such a response has the following implications:

- The server returns an OID along with the `202 Accepted`.
- With `202 Accepted`, the server implies that the following checks have passed:
  - User authorization for creating the object
  - User authorization for read access to any source object for move, copy, serialize, or deserialize
  - Availability of space to create the object or at least enough space to create a URI to report an error
- Future accesses to the URI created (or the OID) shall succeed modulo any delays due to use of eventual consistency.

The client performs GET operations to the URI to track the progress of the operation. In particular, the server returns two fields in its response body to indicate progress:

- A mandatory *completionStatus* text field contains either “`Processing`”, “`Complete`”, or an error string starting with the value “`Error`”.

- An optional *percentComplete* field that indicates the percentage to which the last PUT has completed (0 to 100).

GET does not return any value for the object when *completionStatus* is not `Complete`. When the final result of the create operation is an error, the URI is created with the *completionStatus* field set to the error message. It is the client's responsibility to delete the URI after the error has been noted.

Capabilities:

The following capabilities describe the supported operations that may be performed when creating a new data object:

- Support for the ability to create a new data object is indicated by the presence of the "cdmi_create_dataobject" capability in the parent container.
- If the object being created in the parent container is a reference, support for that ability is indicated by the presence of the "cdmi_create_reference" capability in the parent container.
- If the new data object is a copy of an existing data object, support for the ability to copy is indicated by the presence of the "cdmi_create_copy" capability in the parent container.
- If the new data object is the destination of a move, support for the ability to move the data object is indicated by the presence of the "cdmi_create_move" capability in the parent container.
- If the new data object is the destination of a deserialize operation, support for the ability to deserialize the source data object is indicated by the presence of the "cdmi_deserialize_dataobject" capability in the parent container.
- If the new data object is the destination of a serialize operation, support for the ability to serialize the source data object is indicated by the presence of the "cdmi_serialize_dataobject", "cdmi_serialize_container", "cdmi_serialize_domain", or "cdmi_serialize_queue" capability in the parent container.

Request Headers:

Header	Type	Description	Requirement
Accept	Header String	"application/cdmi-object".	Mandatory
Content-Type	Header String	"application/cdmi-object".	Mandatory
X-CDMI-Specification-Version	Header String	A comma separated list of versions supported by the client, for example, "1.0, 1.5, 2.0".	Mandatory
X-CDMI-Partial	Header String	"true". Indicates that the newly created object is part of a series of writes and the value has not yet been fully populated. When set, the *completionStatus* field shall be set to “`Processing`”.	Optional

Request Message Body:

Field Name	Type	Description	Requirement
mimetype	JSON String	MIME type of the data contained within the *value* field of the data object. • This field shall be kept as part of the metadata and shall be included when deserializing, serializing, copying, moving, or referencing a data object. • If this field is not specified, the value of "`text/plain`" shall be assigned as the field value. • This field shall not be included when referencing a data object.	Optional
metadata	JSON Object	Metadata for the data object. • If this field is included when deserializing, serializing, copying, or moving a data object, the value provided in this field shall replace the metadata from the source URI. • If this field is not included when deserializing, serializing, copying, or moving a data object, the metadata from the source URI shall be used. • If this field is included when creating a new data object by specifying a value, the value provided in this field shall be used as the metadata. • If this field is not included when creating a new data object by specifying a value, an empty JSON object ("{}") shall be assigned as the field value. • This field shall not be included when referencing a data object.	Optional
domainURI	JSON String	URI of the owning domain. If different from the parent domain, the user shall have the "cross_domain" privilege. If not specified, the parent domain shall be used.	Optional
deserialize	JSON String	URI of a serialized CDMI data object that shall be deserialized to create the new data object.	Optional*
serialize	JSON String	URI of a CDMI object that shall be serialized into the new data object.	Optional*
copy	JSON String	URI of a CDMI data object or queue that shall be copied into the new data object.	Optional*
move	JSON String	URI of a CDMI data object or queue that shall be copied into the new data object, then removing the data object or queue value at the source URI upon the successful completion of the copy.	Optional*
reference	JSON String	URI of a CDMI data object that shall be pointed to by a reference. If other fields from this table are supplied when creating a reference, the server shall respond with a `400 Bad Request` error response.	Optional*
deserializevalue	JSON String	A data object serialized as specified in Chapter 15, "Serialization/Deserialization".	Optional*
*Only one of these parameters shall be specified in any given operation, and except for *value*, these fields are not persisted.			

Field Name	Type	Description	Requirement
value	JSON String	JSON-encoded data. • If this field is not included, an empty JSON String ("") shall be assigned as the field value. • Binary data shall be escaped as per the JSON escaping rules described in [RFC4627].	Optional*
*Only one of these parameters shall be specified in any given operation, and except for *value*, these fields are not persisted.			

Response Headers:

Header	Type	Description	Requirement
Content-Type	Header String	"application/cdmi-object".	Mandatory
X-CDMI-Specification-Version	Header String	The server shall respond with the highest version supported by both the client and the server, for example, "1.0".	Mandatory

Response Message Body:

Field Name	Type	Description	Requirement
objectURI	JSON String	URI of the object as specified in the request.	Mandatory
objectID	JSON String	Object ID of the object.	Mandatory
objectName	JSON String	Name of the object. • If the object has a path, the name shall be the last part of the path. • If the object does not have a path and is only accessible by ID, then the name shall be the Object ID of the object. • If an implementor chooses to always return the Object ID as the name of an object, even if one or more paths exists, the *parentURI* shall be set to `/cdmi_objectid/`.	Mandatory
parentURI	JSON String	URI for the parent object. • If the object has a path, the *parentURI* shall be the URI path to the parent object. OR • If the object does not have a path and is only accessible by ID, the *parentURI* shall be set to `/cdmi_objectid/`, and *objectName* shall be set to the Object ID of the object."	Mandatory
domainURI	JSON String	URI of the owning domain.	Mandatory
capabilitiesURI	JSON String	URI to the capabilities for the object. The capabilities URI returned is based on the object type and requested data system *metadata* fields.	Mandatory

Field Name	Type	Description	Requirement
completionStatus	JSON String	A string indicating if the object is still in the process of being created, and once the operation is complete, if it was created successfully or an error occurred. The value shall be the string "`Processing`", the string "`Complete`", or an error string starting with the value "`Error`".	Mandatory
percentComplete	JSON String	The value shall be an integer numeric value from 0 through 100.	Optional
mimetype	JSON String	MIME type of the value of the data object.	Mandatory
metadata	JSON Object	Metadata for the data object. This field includes any user and data system metadata specified in the request body *metadata* field, along with storage system metadata generated by the cloud storage system. See Chapter 16, "Metadata" for a further description of metadata.	Mandatory

Response Status:

HTTP Status	Description
201 Created	New data object was created.
202 Accepted	Data object is in the process of being created. Investigate completionStatus and percentComplete parameters to determine the current status of the operation.
304 Not Modified	The operation conflicts because the object already exists.
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
409 Conflict	The operation conflicts with a non-CDMI access protocol lock or may cause a state transition error on the server.

Example 8-1

PUT to the container URI the data object name and contents

```
PUT /MyContainer/MyDataObject.txt HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0

{
    "mimetype" : "text/plain",
    "metadata" : {

    },
    "value" : "This is the Value of this Data Object"
}
```

The response looks like:

```
HTTP/1.1 201 Created
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0

{
    "objectURI" : "/MyContainer/MyDataObject.txt",
    "objectID" : "0000706D0010B84FAD185C425D8B537E",
    "objectName" : "MyDataObject.txt",
    "parentURI" : "/MyContainer/",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/DataObject",
    "completionStatus" : "Complete",
    "mimetype" : "text/plain",
    "metadata" : {
        "cdmi_size" : "37"
    }
}
```

Example 8-2

PUT to the container URI a move from an existing data object

```
PUT /MyContainer/MySecondDataObject.txt HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0

{
    "move" : "/MyContainer/MyDataObject.txt"
}
```

The response looks like:

```
HTTP/1.1 201 Created
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0

{
    "objectURI" : "/MyContainer/MySecondDataObject.txt",
    "objectID" : "0000706D0010B84FAD185C425D8B537E",
    "objectName" : "MySecondDataObject.txt",
    "parentURI" : "/MyContainer/",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/DataObject",
    "completionStatus" : "Complete",
    "mimetype" : "text/plain",
    "metadata" : {
        "cdmi_size" : "37"
    }
}
```

8.3 Create a Data Object (Non-CDMI Content Type)

Synopsis:

Creates a new data object at the specified URI.

```
PUT <root URI>/<ContainerName>/<DataObjectName>
```

- `<root URI>` is the path to the CDMI cloud.
- `<ContainerName>` is zero or more intermediate containers that already exist.
- `<DataObjectName>` is the name specified for the data object to be created.

Once created, the object may also be accessed at `<root URI>/cdmi_objectid/<objectID>`.

Capabilities:

The following capabilities describe the supported operations that may be performed when creating a new data object:

- Support for the ability to create a new data object is indicated by the presence of the "cdmi_create_dataobject" capability in the parent container.

Request Headers:

Header	Type	Description	Requirement
Content-Type	Header String	The content-type of the data to be stored as a data object. The value specified here shall be used in the *mimetype* field of the CDMI data object.	Mandatory
X-CDMI-Partial	Header String	"true". Indicates that the newly created object is part of a series of writes and has not yet been fully created. When set, the *completionStatus* field shall be set to "`Processing`".	Optional

Request Message Body:

The data to be stored in the value of the data object.

Response Headers:

None specified.

Response Message Body:

None specified.

Response Status:

HTTP Status	Description
201 Created	New data object was created.
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
409 Conflict	The operation conflicts with a non-CDMI access protocol lock or may cause a state transition error on the server.

Example 8-3

PUT to the container URI the data object name and contents

```
PUT /MyContainer/MyDataObject.txt HTTP/1.1
Host: cloud.example.com
Content-Type: text/plain
Content-Length: 37

This is the Value of this Data Object
```

The response looks like:

```
HTTP/1.1 201 Created
```

8.4 Read a Data Object (CDMI Content Type)

Synopsis:

Reads from an existing data object at the specified URI.

```
GET <root URI>/<ContainerName>/<DataObjectName>
GET <root URI>/<ContainerName>/<DataObjectName>?<fieldname>;<fieldname>;...
GET <root URI>/<ContainerName>/<DataObjectName>?value:<range>;...
GET <root URI>/<ContainerName>/<DataObjectName>?metadata:<prefix>;...
```

- `<root URI>` is the path to the CDMI cloud.
- `<ContainerName>` is zero or more intermediate containers.
- `<DataObjectName>` is the name of the data object to be read from.
- `<fieldname>` is the name of a field.
- `<range>` is a byte range within the data object *value* field.
- `<prefix>` is a matching prefix that returns all metadata items that start with the prefix value.

The object may also be accessed at `<root URI>/cdmi_objectid/<objectID>`.

Capabilities:

The following capabilities describe the supported operations that may be performed when reading an existing data object:

- Support for the ability to read the metadata of an existing data object is indicated by the presence of the "cdmi_read_metadata" capability in the specified object.
- Support for the ability to read the value of an existing data object is indicated by the presence of the "cdmi_read_value" capability in the specified object. Any read from a specific byte location not previously written to by a create or update operation shall return zero for the byte value.
- Support for the ability to read the value of an existing data object in specific byte ranges is indicated by the presence of the "cdmi_read_value_range" capability in the specified object. Any read from a specific byte location within the value range specified not previously written to by a create or update operation shall return zero for the byte value.

Request Headers:

Header	Type	Description	Requirement
Accept	Header String	"application/cdmi-object" Shall contain a list of one or more of the five CDMI MIME types.	Mandatory
X-CDMI-Specification-Version	Header String	A comma separated list of versions supported by the client, for example, "1.0, 1.5, 2.0".	Mandatory

Request Message Body:

None specified.

Response Headers:

Header	Type	Description	Requirement
X-CDMI-Specification-Version	Header String	The server shall respond with the highest version supported by both the client and the server, for example, "1.0".	Mandatory
Content-Type	Header String	"application/cdmi-object".	Mandatory
Location	Header String	The server shall respond with the URL that the reference points to if the object is a reference.	Mandatory

Response Message Body:

Field Name	Type	Description	Requirement
objectURI	JSON String	URI of the object as specified in the request.	Mandatory
objectID	JSON String	Object ID of the object.	Mandatory

Field Name	Type	Description	Requirement
objectName	JSON String	Name of the object. • If the object has a path, the name shall be the last part of the path. • If the object does not have a path and is only accessible by ID, then the name shall be the Object ID of the object. • If an implementor chooses to always return the Object ID as the name of an object, even if one or more paths exists, the *parentURI* shall be set to `/cdmi_objectid/`.	Mandatory
parentURI	JSON String	URI for the parent object. • If the object has a path, the *parentURI* shall be the URI path to the parent object. OR • If the object does not have a path and is only accessible by ID, the *parentURI* shall be set to `/cdmi_objectid/`, and *objectName* shall be set to the Object ID of the object."	Mandatory
domainURI	JSON String	URI of the owning domain.	Mandatory
capabilitiesURI	JSON String	URI to the capabilities for the object. The capabilities URI returned is based on the object type and requested data system *metadata* fields.	Mandatory
completionStatus	JSON String	A string indicating if the object is still in the process of being created, and once the operation is complete, if it was created successfully or an error occurred. The value shall be the string "`Processing`", the string "`Complete`", or an error string starting with the value "`Error`".	Mandatory
percentComplete	JSON String	The value shall be an integer numeric value from 0 through 100.	Optional
mimetype	JSON String	MIME type of the value of the data object.	Mandatory
metadata	JSON Object	Metadata for the data object. This field includes any user and data system metadata specified in the request body *metadata* field, along with storage system metadata generated by the cloud storage system. See Chapter 16, "Metadata" for a further description of metadata.	Mandatory
valuerange	JSON String	The range of bytes of the value returned in the *value* field. If the object value has gaps (due to PUTs with non-contigious value ranges), the *valuerange* will indicate the range to the first gap in the object value. The *cdmi_size* storage system metadata of the data object will indicate the size of the object including gaps.	Mandatory
value	String	The data object value encoded in JSON. The *value* field is only provided when *completionStatus* is `Complete`.	Mandatory

If individual fields are specified in the GET request, only these fields are returned in the result body. Optional fields that are requested but do not exist are omitted from the result body.

Response Status:

HTTP Status	Description
200 OK	Valid response is enclosed.
302 Found	The URI is a reference to another URI.
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
404 Not Found	A data object was not found at the specified URI.
406 Not Acceptable	The server is unable to provide the object in the content-type specified in the Accept header.

Example 8-4

GET to the data object URI to read all fields of the data object

```
GET /MyContainer/MyDataObject.txt HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
X-CDMI-Specification-Version: 1.0
```

The response looks like:

```
HTTP/1.1 200 OK
X-CDMI-Specification-Version: 1.0
Content-Type: application/cdmi-object

{
    "objectURI" : "/MyContainer/MyDataObject.txt",
    "objectID" : "0000706D0010B84FAD185C425D8B537E",
    "objectName" : "MyDataObject.txt",
    "parentURI" : "/MyContainer/",
    "domainURI" : "/cdmi_domains/MyDomain",
    "capabilitiesURI" : "/cdmi_capabilities/DataObject",
    "completionStatus" : "Complete",
    "mimetype" : "text/plain",
    "metadata" : {
        "cdmi_size" : "37"
    },
    "valuerange" : "0-36",
    "value" : "This is the Value of this Data Object"
}
```

Example 8-5

GET to the data object URI by ID to read all fields of the data object

```
GET /cdmi_objectid/0000706D0010B84FAD185C425D8B537E
HTTP/1.1 Host: cloud.example.com
Accept: application/cdmi-object
X-CDMI-Specification-Version: 1.0
```

The response looks like:

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0

{
    "objectURI" : "/cdmi_objectid/0000706D0010B84FAD185C425D8B537E",
    "objectID" : "0000706D0010B84FAD185C425D8B537E",
    "objectName" : "MyDataObject.txt",
    "parentURI" : "/MyContainer/",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/DataObject/",
    "completionStatus" : "Complete",
    "mimetype" : "text/plain",
    "metadata" : {
        "cdmi_size" : "37"
    },
    "valuerange" : "0-36",
    "value" : "This is the Value of this Data Object"
}
```

Example 8-6

GET to the data object URI to read the value and mimetype of the data object

```
GET /MyContainer/MyDataObject.txt?value;mimetype HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
X-CDMI-Specification-Version: 1.0
```

The response looks like:

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0

{
    "value" : "This is the Value of this Data Object",
    "mimetype" : "text/plain"
}
```

Example 8-7

GET to the data object URI to read the first ten bytes of the value of the data object

```
GET /MyContainer/MyDataObject.txt?valuerange;value:0-10 HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
X-CDMI-Specification-Version: 1.0
```

The response looks like:

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0

{
    "valuerange" : "0-10",
    "value" : "This is the"
}
```

8.5 Read a Data Object (Non-CDMI Content Type)

Synopsis:

Reads from an existing data object at the specified URI.

```
GET <root URI>/<ContainerName>/<DataObjectName>
GET <root URI>/<ContainerName>/<DataObjectName>?<fieldname>
GET <root URI>/<ContainerName>/<DataObjectName>?metadata:<prefix>
```

- `<root URI>` is the path to the CDMI cloud.
- `<ContainerName>` is zero or more intermediate containers.
- `<DataObjectName>` is the name of the data object to be read from.
- `<fieldname>` is the name of a field.
- `<prefix>` is a matching prefix that returns all metadata items that start with the prefix value.

The object may also be accessed at `<root URI>/cdmi_objectid/<objectID>`.

Capabilities:

The following capabilities describe the supported operations that may be performed when reading an existing data object:

- Support for the ability to read the metadata of an existing data object is indicated by the presence of the "cdmi_read_metadata" capability in the specified object.
- Support for the ability to read the value of an existing data object is indicated by the presence of the "cdmi_read_value" capability in the specified object. Any read from a specific byte location not previously written to by a create or update operation shall return zero for the byte value.
- Support for the ability to read the value of an existing data object in specific byte ranges is indicated by the presence of the "cdmi_read_value_range" capability in the specified object. Any read from a specific byte location within the value range specified not previously written to by a create or update operation shall return zero for the byte value.

Request Header:

Header	Type	Description	Requirement
Range	Header String	A valid ranges-specifier (see [RFC2616] Section 14.35.1).	Optional

Request Message Body:

None specified.

Response Headers:

Header	Type	Description	Requirement
Content-Type	Header String	If a field was not specified, or the *value* field was specified, the content-type returned shall be the *mimetype* field in the data object. If a non-value field was specified, the content-type shall be "application/json".	Mandatory
Location	Header String	The server shall respond with the URL that the reference points to if the object is a reference.	Mandatory

Response Message Body:

- If no fields were specified in the request or the *value* field was specified in the request, the response message body shall be the contents of the data object's *value* field.
- If a field other than the *value* field was specified in the request, the value of that specified field shall be returned in JSON format.
- Requesting an optional field that is not present results in an HTTP status code of 404 Not Found.
- Requesting an undefined field results in an HTTP status codeof 400 Bad Request .

Response Status:

HTTP Status	Description
200 OK	Data object contents in response.
204 No Content	Data object exists but has no content.
206 Partial Content	A requested range of the data object content was returned in response.
302 Found	The URI is a reference to another URI.
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
404 Not Found	A data object was not found at the specified URI, or a requested field within the data object was not found.

Example 8-8

GET to the data object URI to read the value of the data object

```
GET /MyContainer/MyDataObject.txt HTTP/1.1
Host: cloud.example.com
```

The response looks like:

```
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 37

This is the Value of this Data Object
```

Example 8-9

GET to the data object URI to read the *mimetype* field of the data object

```
GET /MyContainer/MyDataObject.txt?mimetype HTTP/1.1
Host: cloud.example.com
```

The response looks like:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "mimetype" : "text/plain"
}
```

Example 8-10

GET to the data object URI to read the first ten bytes of the value of the data object

```
GET /MyContainer/MyDataObject.txt HTTP/1.1
Host: cloud.example.com
Range: bytes=0-10
```

The response looks like:

```
HTTP/1.1 206 Partial Content
Content-Type: text/plain
Content-Range: bytes 0-10
Content-Length: 11

This is the
```

8.6 Update a Data Object (CDMI Content Type)

Synopsis:

Updates an existing data object at the specified URI.

```
PUT <root URI>/<ContainerName>/<DataObjectName>
PUT <root URI>/<ContainerName>/<DataObjectName>?metadata
PUT <root URI>/<ContainerName>/<DataObjectName>?value
PUT <root URI>/<ContainerName>/<DataObjectName>?value:<range>
PUT <root URI>/<ContainerName>/<DataObjectName>?<fieldname>
```

- `<root URI>` is the path to the CDMI cloud.
- `<ContainerName>` is zero or more intermediate containers.

- `<DataObjectName>` is the name of the data object to be updated.
- `<range>` is a byte range within the data object *value* field.

The object may also be accessed at `<root URI>/cdmi_objectid/<objectID>`, and an update shall not result in a change to the Object ID.

Capabilities:

The following capabilities describe the supported operations that may be performed when updating an existing data object:

- Support for the ability to modify the metadata of an existing data object is indicated by the presence of the "cdmi_modify_metadata" capability in the specified object.
- Support for the ability to modify the value of an existing data object is indicated by the presence of the "cdmi_modify_value" capability in the specified object.
- Support for the ability to modify the value of an existing data object in specified byte ranges is indicated by the presence of the "cdmi_modify_value_range" capability in the specified object.

Request Headers:

Header	Type	Description	Requirement
Accept	Header String	"application/cdmi-object".	Mandatory
Content-Type	Header String	"application/cdmi-object".	Mandatory
X-CDMI-Specification-Version	Header String	A comma separated list of versions supported by the client, for example, "1.0, 1.5, 2.0".	Mandatory
X-CDMI-Partial	Header String	"true". Indicates that the object is in the process of being updated, and has not yet been fully updated. When set, the *completionStatus* field shall be set to "`Processing`". If the *completionStatus* field had previously been set to "`Processing`" by including this header in a create or update, the next update without this field shall change the *completionStatus* field back to "`Complete`".	Optional

Request Message Body:

Field Name	Type	Description	Requirement
mimetype	JSON String	MIME type of the data contained within the *value* field of the data object. If present, this replaces the existing mimetype.	Optional
metadata	JSON Object	Metadata for the data object. If present, this replaces the existing metadata. See Chapter 16, "Metadata" for a further description of metadata.	Optional

Field Name	Type	Description	Requirement
domainURI	JSON String	URI of the owning domain. • If different from the parent domain, the user shall have the "cross_domain" privilege. • If not specified, the parent domain shall be used.	Optional
value	JSON String	This is the new data for the object. If present, this replaces the existing value. • If a value range was specified in the request, the new data is inserted at the location specified by the range. Any resulting gaps between ranges shall be treated as if zeros had been written and shall be included when calculating the size of the value. • Binary data shall be escaped as per the JSON escaping rules described in [RFC4627].	Optional

Response Headers:

Header	Type	Description	Requirement
Location	Header String	The server shall respond with the URL that the reference points to if the object is a reference.	Mandatory

Response Message Body:

None specified.

Response Status:

HTTP Status	Description
200 OK	New metadata and/or content accepted.
302 Found	The URI is a reference to another URI.
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
404 Not Found	An update was attempted on an object that does not exist.
409 Conflict	The operation conflicts with a non-CDMI access protocol lock or may cause a state transition error on the server.

Example 8-11

PUT to the data object URI to set new field values

```
PUT /MyContainer/MyDataObject.txt HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0

{
    "mimetype" : "text/plain",
    "metadata" : {

    },
    "value" : "This is the Value of this Data Object"
}
```

The response looks like:

```
HTTP/1.1 200 OK
```

Example 8-12

PUT to the data object URI to set a new mimetype

```
PUT /MyContainer/MyDataObject.txt?mimetype HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0

{
    "mimetype" : "text/plain"
}
```

The response looks like:

```
HTTP/1.1 200 OK
```

Example 8-13

PUT to the data object URI to update a range of the value

```
PUT /MyContainer/MyDataObject.txt?value:21-24 HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0

{
    "value" : "that"
}
```

The response looks like:

```
HTTP/1.1 200 OK
```

8.7 Update a Data Object (Non-CDMI Content Type)

Synopsis:

Updates an existing data object at the specified URI.

```
PUT <root URI>/<ContainerName>/<DataObjectName>
```

- `<root URI>` is the path to the CDMI cloud.
- `<ContainerName>` is zero or more intermediate containers.
- `<DataObjectName>` is the name of the data object to be updated.

The object may also be accessed at `<root URI>/cdmi_objectid/<objectID>`, and an update shall not result in a change to the Object ID.

Capabilities:

The following capabilities describe the supported operations that may be performed when updating an existing data object:

- Support for the ability to modify the value of an existing data object is indicated by the presence of the "cdmi_modify_value" capability in the specified object.
- Support for the ability to modify the value of an existing data object in specified byte ranges is indicated by the presence of the "cdmi_modify_value_range" capability in the specified object.

Request Headers:

Header	Type	Description	Requirement
Content-Type	Header String	The content-type of the data to be stored as a data object. The value specified here shall be used in the *mimetype* field of the CDMI data object.	Mandatory
X-CDMI-Partial	Header String	"true". Indicates that the object is in the process of being updated and has not yet been fully updated. When set, the *completionStatus* field shall be set to "`Processing`". If the *completionStatus* field had previously been set to "`Processing`" by including this header in a create or update, the next update without this field shall change the *completionStatus* field back to "`Complete`".	Optional

Request Message Body:

The data to be stored in the value of the data object.

Response Headers:

Header	Type	Description	Requirement
Location	Header String	The server shall respond with the URL that the reference points to if the object is a reference.	Mandatory

Response Message Body:

None specified.

Response Status:

HTTP Status	Description
200 OK	New value accepted.
302 Found	The URI is a reference to another URI.
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
409 Conflict	The operation conflicts with a non-CDMI access protocol lock or may cause a state transition error on the server.

Example 8-14

PUT to the data object URI to update the value of the data object

```
PUT /MyContainer/MyDataObject.txt HTTP/1.1
Host: cloud.example.com
Content-Type: text/plain
Content-Length: 37

This is the value of this data object
```

The response looks like:

```
HTTP/1.1 200 OK
```

Example 8-15

PUT to the data object URI to update four bytes within of the value of the data object

```
PUT /MyContainer/MyDataObject.txt HTTP/1.1
Host: cloud.example.com
Content-Range: bytes=21-24
Content-Type: text/plain
Content-Length: 4

that
```

The response looks like:

```
HTTP/1.1 200 OK
```

8.8 Delete a Data Object

Synopsis:

Deletes an existing data object at the specified URI.

```
DELETE <root URI>/<ContainerName>/<DataObjectName>
```

- `<root URI>` is the path to the CDMI cloud.
- `<ContainerName>` is zero or more intermediate containers.
- `<DataObjectName>` is the name of the data object to be deleted.

The object may also be accessed at `<root URI>/cdmi_objectid/<objectID>`.

Capabilities:

The following capabilities describe the supported operations that may be performed when deleting an existing data object:

- Support for the ability to delete an existing data object is indicated by the presence of the "cdmi_delete_object" capability in the specified object.

Request Headers:

Header	Type	Description	Requirement
X-CDMI-Specification-Version	Header String	A comma separated list of versions supported by the client, for example, "1.0, 1.5, 2.0".	Mandatory

Request Message Body:

None specified.

Response Headers:

None specified.

Response Message Body:

None specified.

Response Status:

HTTP Status	Description
204 No Content	Data object was successfully deleted.
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.

HTTP Status	Description
404 Not Found	The resource specified was not found.
409 Conflict	The operation conflicts with a non-CDMI access protocol lock or may cause a state transition error on the server or he data object may not be deleted.

Example 8-16

DELETE to the data object URI

```
DELETE /MyContainer/MyDataObject.txt HTTP/1.1
Host: cloud.example.com
X-CDMI-Specification-Version: 1.0
```

The response looks like:

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

9 Container Object Resource Operations

9.1 Overview

Container objects are the fundamental grouping of stored data within CDMI and are analogous to directories within a filesystem. Each container has zero or more child objects and a set of well-defined fields that include standardized and optional metadata generated by the cloud storage system and specified by the cloud user.

Containers are addressed in CDMI in the two following ways:

- `http://cloud.example.com/container/`
- `http://cloud.example.com/cdmi_objectid/0000706D0010B84FAD185C425D8B537E`

The first example addresses the container by URI, and the second addresses the container by CDMI object ID. Every container has a single, globally-unique object identifier (OID) that remains constant for the life of the object. Each container may also have one or more URI addresses that allow the container to be accessed.

Containers may also be nested, as follows:

```
http://cloud.example.com/container/sub-container/
```

A container may have a parent object. In this case, the container inherits data system metadata from its parent object. For example, the "sub-container" container may inherit its data system metadata from the parent container "container".

This model allows direct mapping between CDMI-managed cloud storage and filesystems (for example), as CDMI containers may be exported as NFSv4 or WebDAV filesystems, with all metadata visible. As files are created, the files and directories are then visible through the CDMI interface acting as a data path. This mapping, though possible, is not further described in this standard.

Individual fields within a container may be accessed by specifying the field name after a question mark "?" appended to the end of the container URI. For example, the below URI would return just the *children* field in the response body:

```
http://cloud.example.com/container?children
```

By specifying a range after the *children* field name, specific ranges of the *children* field may be accessed. For example, the below URI returns the first two children from the *children* field:

```
http://cloud.example.com/container?children:0-2
```

Children ranges are specified in a similar fashion to byte ranges as per Section 14.35.1 of [RFC2616].

A list of fields, separated by a semicolon ";" may be specified, allowing multiple fields to be accessed in a single request. For example, the below URI would return the *children* and *metadata* fields in the response body:

```
http://cloud.example.com/container?children;metadata
```

9.1.1 Container Metadata

The following optional data system metadata may be provided:

Metadata Name	Type	Description	Requirement
cdmi_assignedsize	JSON String	The number of bytes that is reported via exported protocols (may be thin provisioned). This number may limit *cdmi_size*.	Optional

Container metadata may also include arbitrary user-supplied metadata and data system metadata as described in Chapter 16, "Metadata".

9.1.2 Container Object Addressing

Each container object is addressed via one or more unique URIs, and all operations may be performed through any of these URIs. For example, a data object may be accessible via multiple virtual hosting paths, where "`http://cloud.example.com/users/snia/cdmi/`" is also accessible through "`http://snia.example.com/cdmi/`". Conflicting writes via different paths are managed the same way that conflicting writes via one path are managed, via the principle of eventual consistency (see Section 9.2, "Create a Container (CDMI Content Type)").

9.1.3 Container Object Representations

The representations in this section are shown using JSON notation. A conforming implementation shall support the mandatory fields and may support the optional fields. The parameter fields may be specified or returned in any order. Both clients and servers shall support JSON representation.

9.2 Create a Container (CDMI Content Type)

Synopsis:

Creates a new container at the specified URI.

```
PUT <root URI>/<ContainerName>/<NewContainerName>/
```

- `<root URI>` is the path to the CDMI cloud.
- `<ContainerName>` is zero or more intermediate containers that already exist.
- `<NewContainerName>` is the name specified for the container to be created.

Once created, the container may also be accessed at `<root URI>/cdmi_objectid/<objectID>`.

Delayed Completion of Create:

On a create operation for a container, the server may return a response of `202 Accepted`. In this case, the container is in the process of being created. This response is particularly useful for long-running operations, for instance, for deserializing a source data object to create a large container hierarchy. Such a response has the following implications:

- The server returns an OID along with `202 Accepted`.

- With `202 Accepted`, the server implies that the following checks have passed:
  - User authorization for creating the container
  - User authorization for read access to any source object for move, copy, serialize, or deserialize
  - Availability of space to create the container or at least enough space to create a URI to report an error
- Future accesses to the URI created (or the object ID) shall succeed modulo any delays due to use of eventual consistency.

The client performs GET operations to the URI to track the progress of the operation. In particular, the server returns two fields in its response body to indicate progress:

- A mandatory *completionStatus* text field contains either “`Processing`”, “`Complete`”, or an error string starting with the value "`Error`".
- An optional *percentComplete* field that indicates the percentage to which the last PUT has completed (0 to 100). GET does not return any children for the container when *completionStatus* is not “`Complete`”.

When the final result of the create operation is an error, the URI is created with the *completionStatus* field set to the error message. It is the client's responsibility to delete the URI after the error has been noted.

Capabilities:

The following capabilities describe the supported operations that may be performed when creating a new container:

- Support for the ability to create a new container object is indicated by the presence of the "cdmi_create_container" capability in the parent container.
- If the object being created in the parent container is a reference, support for that ability is indicated by the presence of the "cdmi_create_reference" capability in the parent container.
- If the new container object is a copy of an existing container, support for the ability to copy is indicated by the presence of the "cdmi_create_copy" capability in the parent container.
- If the new container is the destination of a move, support for the ability to move the container is indicated by the presence of the "cdmi_create_move" capability in the parent container.
- If the new container is the destination of a deserialize operation, support for the ability to deserialize the source data object serialization of a container is indicated by the presence of the "cdmi_deserialize_container" capability in the parent container.

Request Headers:

Header	Type	Description	Requirement
Accept	Header String	"application/cdmi-container".	Mandatory
Content-Type	Header String	"application/cdmi-container".	Mandatory
X-CDMI-Specification-Version	Header String	A comma separated list of versions supported by the client, for example, "1.0, 1.5, 2.0".	Mandatory

Request Message Body:

Field Name	Type	Description	Requirement
metadata	JSON Object	Metadata for the container. • If this field is included when deserializing, serializing, copying, or moving a container, the value provided in this field shall replace the metadata from the source URI. • If this field is not included when deserializing, serializing, copying, or moving a container, the metadata from the source URI shall be used. • If this field is included when creating a new container by specifying a value, the value provided in this field shall be used as the metadata. • If this field is not included when creating a new container by specifying a value, an empty JSON object ("{}") shall be assigned as the field value. • This field shall not be included when referencing a container.	Optional
domainURI	JSON String	URI of the owning domain. • If different from the parent domain, the user shall have the "cross_domain" privilege. • If not specified, the parent domain shall be used.	Optional
exports	JSON Object	A structure for each protocol enabled for this container. This field shall not be included when referencing a data object.	Optional
deserialize	JSON String	URI of a serialized CDMI data object that shall be deserialized to create the new container, including all child objects inside the source serialized data object. When deserializing a container, any exported protocols from the original serialized container are not applied to the newly created container(s).	Optional*
copy	JSON String	URI of a CDMI container that shall be copied into the new container, including all child objects under the source container. When copying a container, exported protocols are not preserved across the copy.	Optional*
*If present, only one of these parameters shall be specified in any given operation, and these fields are not persisted.			

Field Name	Type	Description	Requirement
move	JSON String	URI of a CDMI container that shall be copied into the new container, including all child objects under the source container., then removing the container at the source URI upon the successful completion of the copy.	Optional*
reference	JSON String	URI of a CDMI data object that shall be pointed to by a reference. If other fields from this table are supplied when creating a reference, the server shall respond with a `400 Bad Request` error response.	Optional*
deserializevalue	JSON String	A container object serialized as specified in Chapter 15, "Serialization/Deserialization".	Optional*
*If present, only one of these parameters shall be specified in any given operation, and these fields are not persisted.			

Response Headers:

Header	Type	Description	Requirement
Content-Type	Header String	"application/cdmi-container".	Mandatory
X-CDMI-Specification-Version	Header String	The server shall respond with the highest version supported by both the client and the server, for example, "1.0".	Mandatory

Response Message Body:

Field Name	Type	Description	Requirement
objectURI	JSON String	URI of the container as specified in the request.	Mandatory
objectID	JSON String	Object ID of the object.	Mandatory
objectName	JSON String	Name of the object. • If the object has a path, the name shall be the last part of the path. • If the object does not have a path and is only accessible by ID, then the name shall be the Object ID of the object. • If an implementor chooses to always return the Object ID as the name of an object, even if one or more paths exists, the *parentURI* shall be set to `/cdmi_objectid/`.	Mandatory
parentURI	JSON String	URI for the parent object. • If the object has a path, the *parentURI* shall be the URI path to the parent object. OR • If the object does not have a path and is only accessible by ID, the *parentURI* shall be set to `/cdmi_objectid/`, and *objectName* shall be set to the Object ID of the object."	Mandatory

Field Name	Type	Description	Requirement
domainURI	JSON String	URI of the owning domain.	Mandatory
capabilitiesURI	JSON String	URI to the capabilities for the object. The capabilities URI returned is based on the object type and requested data system *metadata* fields.	Mandatory
completionStatus	JSON String	A string indicating if the object is still in the process of being created, and once the operation is complete, if it was created successfully or an error occurred. The value shall be the string "`Processing`", the string "`Complete`", or an error string starting with the value "`Error`".	Mandatory
percentComplete	JSON String	The value shall be an integer numeric value from 0 through 100.	Optional
metadata	JSON Object	Metadata for the container. This field includes any user and data system metadata specified in the request body *metadata* field, along with storage system metadata generated by the cloud storage system. See Chapter 16, "Metadata" for a further description of metadata.	Mandatory
exports	JSON Object	A structure for each protocol that is enabled for this container. See Chapter 13, "Exported Protocols".	Optional (returned only if present)
snapshots	JSON Array	URI(s) of the SnapShot containers. See Chapter 14, "Snapshots".	Optional (returned only if present)
childrenrange	JSON String	The range of the children returned in the *children* field.	Mandatory
children	JSON Array	Names of the children objects in the container. Child containers end with "/".	Mandatory

Response Status:

HTTP Status	Description
201 Created	New container was created.
202 Accepted	Container is in the process of being created. Investigate *completionStatus* and *percentComplete* parameters to determine the current status of the operation.
304 Not Modified	The operation conflicts because the container already exists.
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
409 Conflict	The container name already exists.

Example 9-1

PUT to the URI the container name and metadata

```
PUT /MyContainer HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-container
Content-Type: application/cdmi-container
X-CDMI-Specification-Version: 1.0

{
    "metadata" : {

    },
    "exports" : {
        "OCCI/iSCSI" : {
            "identifier" : "0000706D0010B84FAD185C425D8B537E",
            "permissions" : "0000706D00107B85BFE6D20B84D603CA"
        },
        "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

{
    "objectURI" : "/MyContainer/",
    "objectID" : "0000706D0010B84FAD185C425D8B537E",
    "objectName" : "MyContainer/",
    "parentURI" : "/",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/Container",
    "completionStatus" : "Complete",
    "metadata" : {

    },
    "exports" : {
        "OCCI/iSCSI" : {
            "identifier" : "0000706D0010B84FAD185C425D8B537E",
            "permissions" : "0000706D00107B85BFE6D20B84D603CA"
        },
        "Network/NFSv4" : {
            "identifier" : "/users",
            "permissions" : "domain"
        }
    },
    "childrenrange" : "",
    "children" : [

    ]
}
```

9.3 Create a Container (Non-CDMI Content Type)

Synopsis:

Creates a new container at the specified URI.

```
PUT <root URI>/<ContainerName>/<NewContainerName>/
```

- `<root URI>` is the path to the CDMI cloud.
- `<ContainerName>` is zero or more intermediate containers that already exist.
- `<NewContainerName>` is the name specified for the container to be created.

Once created, the container may also be accessed at `<root URI>/cdmi_objectid/<objectID>`.

Capabilities:

The following capabilities describe the supported operations that may be performed when creating a new container:

- Support for the ability to create a new container object is indicated by the presence of the "cdmi_create_container" capability in the parent container.

Request Headers:

None specified.

Request Message Body:

None specified.

Response Headers:

None specified.

Response Message Body:

None specified.

Response Status:

HTTP Status	Description
201 Created	New container was created.
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
409 Conflict	The container name already exists.

Example 9-2

PUT to the URI the container name.

```
PUT /MyContainer/ HTTP/1.1
Host: cloud.example.com
```

The response looks like:

```
HTTP/1.1 201 Created
```

9.4 Read a Container Object (CDMI Content Type)

Synopsis:

Reads from an existing container at the specified URI.

```
GET <root URI>/<ContainerName>/<TheContainerName>/
GET <root URI>/<ContainerName>/<TheContainerName>/?<fieldname>;<fieldname>;...
GET <root URI>/<ContainerName>/<TheContainerName>/?children:<range>;...
GET <root URI>/<ContainerName>/<TheContainerName>/?metadata:<prefix>;...
```

- `<root URI>` is the path to the CDMI cloud.
- `<ContainerName>` is zero or more intermediate containers.
- `<TheContainerName>` is the name specified for the container to be read from.
- `<fieldname>` is the name of a field.
- `<range>` is a numeric range within the list of children.
- `<prefix>` is a matching prefix that returns all metadata items that start with the prefix value.

The object may also be accessed at `<root URI>/cdmi_objectid/<objectID>`.

Capabilities:

The following capabilities describe the supported operations that may be performed when reading an existing container:

- Support for the ability to read the metadata of an existing container object is indicated by the presence of the "cdmi_read_metadata" capability in the specified container.
- Support for the ability to list the children of an existing container is indicated by the presence of the "cdmi_list_children" capability in the specified container.
- Support for the ability to list ranges of the children of an existing container is indicated by the presence of the "cdmi_list_children_range" capability in the specified container.

Request Headers:

Header	Type	Description	Requirement
Accept	Header String	"application/cdmi-container". Shall contain a list of one or more of the five CDMI MIME types.	Mandatory
X-CDMI-Specification-Version	Header String	A comma separated list of versions supported by the client, for example, "1.0, 1.5, 2.0".	Mandatory

Request Message Body:

None specified.

Response Headers:

Header	Type	Description	Requirement
X-CDMI-Specification-Version	Header String	The server shall respond with the highest version supported by both the client and the server, for example, "1.0".	Mandatory
Content-Type	Header String	"application/cdmi-container".	Mandatory
Location	Header String	The server shall respond with the URL that the reference points to if the object is a reference.	Mandatory

Response Message Body:

Field Name	Type	Description	Requirement
objectURI	JSON String	URI of the object as specified in the request.	Mandatory
objectID	JSON String	Object ID of the object.	Mandatory
objectName	JSON String	Name of the object. • If the object has a path, the name shall be the last part of the path. • If the object does not have a path and is only accessible by ID, then the name shall be the Object ID of the object. • If an implementor chooses to always return the Object ID as the name of an object, even if one or more paths exists, the *parentURI* shall be set to `/cdmi_objectid/`.	Mandatory
parentURI	JSON String	URI for the parent object. • If the object has a path, the *parentURI* shall be the URI path to the parent object. OR • If the object does not have a path and is only accessible by ID, the *parentURI* shall be set to `/cdmi_objectid/`, and *objectName* shall be set to the Object ID of the object."	Mandatory

Field Name	Type	Description	Requirement
domainURI	JSON String	URI of the owning domain.	Mandatory
capabilitiesURI	JSON String	URI to the capabilities for the object. The capabilities URI returned is based on the object type and requested data system *metadata* fields.	Mandatory
completionStatus	JSON String	A string indicating if the object is still in the process of being created, and once the operation is complete, if it was created successfully or an error occurred. The value shall be the string "`Processing`", the string "`Complete`", or an error string starting with the value "`Error`".	Mandatory
percentComplete	JSON String	The value shall be an integer numeric value from 0 through 100.	Optional
metadata	JSON Object	Metadata for the container. This field includes any user and data system metadata specified in the request body *metadata* field, along with storage system metadata generated by the cloud storage system. See Chapter 16, "Metadata" for a further description of metadata.	Mandatory
exports	JSON Object	A structure for each protocol that is enabled for this container.	Optional (returned only if present)
snapshots	JSON Array	URI(s) of the SnapShot containers.	Optional (returned only if present)
childrenrange	JSON String	The range of the children returned in the *children* field.	Mandatory
children	JSON Array	Names of the children objects in the container. Child containers end with "/".	Mandatory

If individual fields are specified in the GET request, only these fields are returned in the result body. Optional fields that are requested but do not exist are omitted from the result body.

Response Status:

HTTP Status	Description
200 OK	Metadata for the container object provided in the message body.
302 Found	The URI is a reference to another URI.
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
404 Not Found	A container was not found at the specified URI.
406 Not Acceptable	The server is unable to provide the object in the content-type specified in the Accept header.

Example 9-3

GET to the container URI to read all the fields of the container

```
GET /MyContainer HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-container
X-CDMI-Specification-Version: 1.0
```

The response looks like:

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-container
X-CDMI-Specification-Version: 1.0

{
    "objectURI" : "/MyContainer",
    "objectID" : "0000706D0010B84FAD185C425D8B537E",
    "objectName" : "MyContainer",
    "parentURI" : "/",
    "domainURI" : "/cdmi_domains/MyDomain",
    "capabilitiesURI" : "/cdmi_capabilities/Container",
    "completionStatus" : "Complete",
    "metadata" : {

    },
    "exports" : {
        "OCCI/iSCSI" : {
            "identifier" : "0000706D0010B84FAD185C425D8B537E",
            "permissions" : "0000706D00107B85BFE6D20B84D603CA"
        },
        "Network/NFSv4" : {
            "identifier" : "/users",
            "permissions" : "domain"
        },
        "childrenrange" : "0-4",
        "children" : [
            "red",
            "green",
            "yellow",
            "orange/",
            "purple/"
        ]
    }
}
```

Example 9-4

GET to the container URI to read *parentURI* and children of the container

```
GET /MyContainer/?parentURI;children HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-container
X-CDMI-Specification-Version: 1.0
```

The response looks like:

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-container
X-CDMI-Specification-Version: 1.0

{
    "parentURI" : "/",
    "children" : [
        "red",
        "green",
        "yellow",
        "orange/",
        "purple/"
    ]
}
```

Example 9-5

GET to the container URI to read children 0..2 of the container

```
GET /MyContainer/?childrenrange;children:0-2 HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-container
X-CDMI-Specification-Version: 1.0
```

The response looks like:

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-container
X-CDMI-Specification-Version: 1.0

{
    "childrenrange" : "0-2",
    "children" : [
        "red",
        "green",
        "yellow"
    ]
}
```

9.5 Read a Container Object (Non-CDMI Content Type)

Synopsis:

Reads from an existing container at the specified URI.

```
GET <root URI>/<ContainerName>/<TheContainerName>/?<fieldname>
GET <root URI>/<ContainerName>/<TheContainerName>/?children:<range>
GET <root URI>/<ContainerName>/<TheContainerName>/?metadata:<prefix>
```

- `<root URI>` is the path to the CDMI cloud.
- `<ContainerName>` is zero or more intermediate containers.
- `<TheContainerName>` is the name specified for the container to be read from.
- `<fieldname>` is the name of a field.

- `<range>` is a numeric range within the list of children.
- `<prefix>` is a matching prefix that returns all metadata items that start with the prefix value.

The object may also be accessed at `<root URI>/cdmi_objectid/<objectID>`.

Note: CDMI does not define the format for a GET of a container when fields are not being requested.

Capabilities:

The following capabilities describe the supported operations that may be performed when reading an existing container:

- Support for the ability to read the metadata of an existing container object is indicated by the presence of the "cdmi_read_metadata" capability in the specified container.
- Support for the ability to list the children of an existing container is indicated by the presence of the "cdmi_list_children" capability in the specified container.
- Support for the ability to list ranges of the children of an existing container is indicated by the presence of the "cdmi_list_children_range" capability in the specified container.

Request Headers:

None specified.

Request Message Body:

None specified.

Response Headers:

Header	Type	Description	Requirement
Content-Type	Header String	"application/json".	Mandatory
Location	Header String	The server shall respond with the URL that the reference points to if the object is a reference.	Mandatory

Response Message Body:

- The value of the specified field shall be returned in JSON format.
- Requesting an optional field that is not present results in a 404 Not Found HTTP status code.
- Requesting an undefined field results in a 400 Bad Request HTTP status code.

Response Status:

HTTP Status	Description
200 OK	Metadata for the container object provided in the message body.
302 Found	The URI is a reference to another URI.
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
404 Not Found	A container was not found at the specified URI.

Example 9-6

GET to the container URI to read *parentURI* of the container

```
GET /MyContainer/?parentURI HTTP/1.1
Host: cloud.example.com
```

The response looks like:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "parentURI" : "/"
}
```

Example 9-7

GET to the container URI to read children 0..2 of the container

```
GET /MyContainer/?childrenrange;children:0-2 HTTP/1.1
Host: cloud.example.com
```

The response looks like:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "childrenrange" : "0-2",
    "children" : [
        "red",
        "green",
        "yellow"
    ]
}
```

9.6 Update a Container (CDMI Content Type)

Synopsis:

Updates an existing container at the specified URI.

```
PUT <root URI>/<ContainerName>/<TheContainerName>/
PUT <root URI>/<ContainerName>/<TheContainerName>/?metadata
```

- `<root URI>` is the path to the CDMI cloud.
- `<ContainerName>` is zero or more intermediate containers.
- `<TheContainerName>` is the name of the container to be updated.

The object may also be accessed at `<root URI>/cdmi_objectid/<objectID>`, and an update shall not result in a change to the Object ID.

Delayed completion of snapshot: On a snapshot operation for a container, the server may return a response of `202 Accepted`. Such a response has the following implications:

- With `202 Accepted`, the server implies that the following checks have passed:
  - User authorization for creating the snapshot
  - User authorization for read access to the container
- Availability of space to create the snapshot or at least enough space to create a URI to report an error.
- Future accesses to the snapshot URI shall succeed modulo any delays due to use of eventual consistency.

The client performs GET operations to the snapshot URI to track the progress of the operation. In particular, the server returns two fields in its response body to indicate progress:

- A *completionStatus* text field contains either “`Processing`”, “`Complete`”, or an error string starting with the value "`Error`".
- An optional *percentComplete* field that indicates the percentage to which the last PUT has completed (0 to 100). GET does not return any value for the object when *completionStatus* is not “`Complete`”.

When the final result of the snapshot operation is an error, the snapshot URI is created with the *completionStatus* field set to the error message. It is the client's responsibility to delete the URI after the error has been noted.

Capabilities:

The following capabilities describe the supported operations that may be performed when updating an existing container:

- Support for the ability to modify the metadata of an existing container is indicated by the presence of the "cdmi_modify_metadata" capability in the specified container.
- Support for the ability to snapshot the contents of an existing container is indicated by the presence of the "cdmi_snapshot" capability in the specified container.

- Support for the ability to add an exported protocol to an existing container is indicated by the presence of the "cdmi_export_<protocol>" capabilities for the specified container.

Request Headers:

Header	Type	Description	Requirement
Accept	Header String	"application/cdmi-container".	Mandatory
Content-Type	Header String	"application/cdmi-container".	Mandatory
X-CDMI-Specification-Version	Header String	A comma separated list of versions supported by the client, for example, "1.0, 1.5, 2.0".	Mandatory

Request Message Body:

Field Name	Type	Description	Requirement
metadata	JSON Object	Metadata for the container. If present, this replaces the existing metadata. See Chapter 16, "Metadata" for a further description of metadata.	Optional
domainURI	JSON String	URI of the owning domain. • If different from the parent domain, the user shall have the "cross_domain" privilege. • If not specified, the parent domain shall be used.	Optional
snapshot	JSON String	Name of the snapshot to be taken. If a snapshot is added or altered, the PUT operation only returns after the snapshot is added to the snapshot list. Once created, snapshots may be accessed as children containers under the cdmi_snapshots child container of the container receiving a snapshot.	Optional
exports	JSON Object	A structure for each protocol that is enabled for this container. If an exported protocol is added or altered, the PUT operation only returns after the export operation has completed.	Optional

Response Headers:

Header	Type	Description	Requirement
Location	Header String	The server shall respond with the URL that the reference points to if the object is a reference.	Mandatory

Response Message Body:

None specified.

Response Status:

HTTP Status	Description
200 OK	Container was updated.
202 Accepted	Container or a snapshot (sub-container) is in the process of being created. Investigate *completionStatus* and *percentComplete* parameters to determine the current status of the operation.
302 Found	The URI is a reference to another URI.
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
404 Not Found	An update was attempted on a container that does not exist.
409 Conflict	The operation conflicts with a non-CDMI access protocol lock or may cause a state transition error on the server.

Example 9-8

PUT to the container URI to set new field values

```
PUT /MyContainer/ HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-container
Content-Type: application/cdmi-container
X-CDMI-Specification-Version: 1.0

{
    "metadata" : {

    } ,
    "exports" : {
        "OCCI/iSCSI" : {
            "identifier" : "0000706D0010B84FAD185C425D8B537E",
            "permissions" : "0000706D00107B85BFE6D20B84D603CA"
        },
        "Network/NFSv4" : {
            "identifier" : "/users",
            "permissions" : "domain"
        }
    }
}
```

The response looks like:

```
HTTP/1.1 200 OK
```

Example 9-9

PUT to the container URI to set a new exported protocol value

```
PUT /MyContainer/?exports HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-container
Content-Type: application/cdmi-container
X-CDMI-Specification-Version: 1.0

{
    "exports" : {
        "OCCI/iSCSI" : {
            "identifier" : "0000706D0010B84FAD185C425D8B537E",
            "permissions" : "0000706D00107B85BFE6D20B84D603CA"
        } ,
        "Network/NFSv4" : {
            "identifier" : "/users",
            "permissions" : "domain"
        }
    }
}
```

The response looks like:

```
HTTP/1.1 200 OK
```

9.7 Delete a Container Object

Synopsis:

Deletes an existing container, all contained children, and snapshots at the specified URI.

```
DELETE <root URI>/<ContainerName>/<TheContainerName>/
```

- `<root URI>` is the path to the CDMI cloud.
- `<ContainerName>` is zero or more intermediate containers.
- `<TheContainerName>` is the name of the container to be deleted.

The object may also be accessed at `<root URI>/cdmi_objectid/<objectID>`.

Capabilities:

The following capabilities describe the supported operations that may be performed when deleting an existing container:

- Support for the ability to delete an existing data object is indicated by the presence of the "cdmi_delete_container" capability in the specified container.

Request Headers:

Header	Type	Description	Requirement
X-CDMI-Specification-Version	Header String	A comma separated list of versions supported by the client, for example, "1.0, 1.5, 2.0".	Mandatory

Request Message Body:

None specified.

Response Headers:

None specified.

Response Message Body:

None specified.

Response Status:

HTTP Status	Description
204 No Content	Container object was successfully deleted.
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
404 Not Found	The resource specified was not found
409 Conflict	The container object may not be deleted

Example 9-10

DELETE to the container URI

```
DELETE /MyContainer/ HTTP/1.1
Host: cloud.example.com
X-CDMI-Specification-Version: 1.0
```

The response looks like:

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

9.8 Create (POST) a New Data Object (CDMI Content Type)

Synopsis:

Creates a new data object as a child of the specified container URI.

```
POST <root URI>/<ContainerName>/
POST <root URI>/cdmi_objectid/
```

- `<root URI>` is the path to the CDMI cloud.
- `<ContainerName>` is zero or more intermediate containers that already exist.

Once created, the object may also be accessed at `<root URI>/cdmi_objectid/<objectID>`.

Delayed Completion of Create:

On a create operation for a data object, the server may return a response of `202 Accepted`. In this case, the object is in the process of being created. This response is particularly useful for long-running operations, for instance, copying a large data object from a source URI. Such a response has the following implications:

- The server returns an OID along with the `202 Accepted`.
- With `202 Accepted`, the server implies that the following checks have passed:
  - User authorization for creating the object
  - User authorization for read access to any source object for move, copy, serialize, or deserialize
  - Availability of space to create the object or at least enough space to create a URI to report an error
- Future accesses to the URI created (or the object ID) shall succeed modulo any delays due to use of eventual consistency.

The client performs GET operations to the URI to track the progress of the operation. In particular, the server returns two fields in its response body to indicate progress:

- A mandatory *completionStatus* text field contains either “`Processing`”, “`Complete`”, or an error string starting with the value "`Error`".
- An optional *percentComplete* field that indicates the percentage to which the last PUT has completed (0 to 100).

GET does not return any value for the object when *completionStatus* is not “Complete”. When the final result of the create operation is an error, the URI is created with the *completionStatus* field set to the error message. It is the client's responsibility to delete the URI after the error has been noted.

Capabilities:

The following capabilities describe the supported operations that may be performed when creating a new data object:

- Support for the ability to create data objects through this operation is indicated by both the presence of the "cdmi_post_dataobject" capability the presence of the "cdmi_create_dataobject" capability in the specified container.
- If the object being created in the parent container is a reference, support for that ability is indicated by the presence of the "cdmi_create_reference" capability in the parent container.
- If the new data object is a copy of an existing data object, support for the ability to copy is indicated by the presence of the "cdmi_create_copy" capability in the parent container.
- If the new data object is the destination of a move, support for the ability to move the data object is indicated by the presence of the "cdmi_create_move" capability in the parent container.

- If the new data object is the destination of a deserialize operation, support for the ability to deserialize the source data object is indicated by the presence of the "cdmi_deserialize_dataobject" capability in the parent container.
- If the new data object is the destination of a serialize operation, support for the ability to serialize the source data object is indicated by the presence of the "cdmi_serialize_dataobject", "cdmi_serialize_container", "cdmi_serialize_domain", or "cdmi_serialize_queue" capability in the parent container.

Request Headers:

Header	Type	Description	Requirement
Accept	Header String	"application/cdmi-object".	Mandatory
Content-Type	Header String	"application/cdmi-object".	Mandatory
X-CDMI-Specification-Version	Header String	A comma separated list of versions supported by the client, for example, "1.0, 1.5, 2.0".	Mandatory

Request Message Body:

Field Name	Type	Description	Requirement
mimetype	JSON String	MIME type of the data contained within the *value* field of the data object. • This field shall not be included when deserializing, serializing, copying, moving, or referencing a data object. • If this field is not specified, the value of "text/plain" shall be assigned as the field value.	Optional
metadata	JSON Object	Metadata for the data object. • If this field is included when deserializing, serializing, copying, or moving a data object, the value provided in this field shall replace the metadata from the source URI. • If this field is not included when deserializing, serializing, copying, or moving a data object, the metadata from the source URI shall be used. • If this field is included when creating a new data object by specifying a value, the value provided in this field shall be used as the metadata. • If this field is not included when creating a new data object by specifying a value, an empty JSON object ("{}") shall be assigned as the field value. • This field shall not be included when referencing a data object.	Optional
*Only one of these parameters shall be specified in any given operation, and except for *value*, these fields are not persisted. If more than one of these fields are supplied, the server shall respond with a `400 Bad Request` error response.			

Field Name	Type	Description	Requirement
domainURI	JSON String	URI of the owning domain. • If different from the parent domain, the user shall have the "cross_domain" privilege. • If not specified, the parent domain shall be used. • If creating an object by ID using `/cdmi_objectid/`, there is no parent container, so the domain must be specified.	Optional
deserialize	JSON String	URI of a serialized CDMI data object that shall be deserialized to create the new data object.	Optional*
serialize	JSON String	URI of a CDMI object that shall be serialized into the new data object.	Optional*
copy	JSON String	URI of a CDMI data object or queue that shall be copied into the new data object.	Optional*
move	JSON String	URI of a CDMI data object or queue that shall be copied into the new data object, then removing the data object or queue value at the source URI upon the successful completion of the copy.	Optional*
reference	JSON String	URI of a CDMI data object that shall be pointed to by a reference. If other fields from this table are supplied when creating a reference, the server shall respond with a `400 Bad Request` error response.	Optional*
deserializevalue	JSON String	A data object serialized as specified in Chapter 15, "Serialization/Deserialization".	Optional*
value	JSON String	JSON-encoded data. If this field is not included, an empty JSON String ("") shall be assigned as the field value. Binary data shall be escaped as per the JSON escaping rules described in [RFC4627].	Optional*
*Only one of these parameters shall be specified in any given operation, and except for *value*, these fields are not persisted. If more than one of these fields are supplied, the server shall respond with a `400 Bad Request` error response.			

Response Headers:

Header	Type	Description	Requirement
Accept	Header String	"application/cdmi-object".	Mandatory
X-CDMI-Specification-Version	Header String	The server shall respond with the highest version supported by both the client and the server, for example, "1.0".	Mandatory
Location	Header String	The unique URL for the new data object as assigned by the system. In the absence of file name information from the client the system shall assign the URL in the form: `<root URL>/<ContainerName>/<ObjectID>`.	Mandatory

Response Message Body:

Field Name	Type	Description	Requirement
objectURI	JSON String	URI of the object as assigned by the system.	Mandatory
objectID	JSON String	Object ID of the object.	Mandatory
objectName	JSON String	Name of the object. • If the object has a path, the name shall be the last part of the path. • If the object does not have a path and is only accessible by ID, then the name shall be the Object ID of the object. • If an implementor chooses to always return the Object ID as the name of an object, even if one or more paths exists, the *parentURI* shall be set to `/cdmi_objectID/`.	Mandatory
parentURI	JSON String	URI for the parent object. • If the object has a path, the *parentURI* shall be the URI path to the parent object. OR • If the object does not have a path and is only accessible by ID, the *parentURI* shall be set to `/cdmi_objectid/`, and *objectName* shall be set to the Object ID of the object."	Mandatory
domainURI	JSON String	URI of the owning domain.	Mandatory
capabilitiesURI	JSON String	URI to the capabilities for the object. The capabilities URI returned is based on the object type and requested data system *metadata* fields.	Mandatory
completionStatus	JSON String	A string indicating if the object is still in the process of being created, and once the operation is complete, if it was created successfully or an error occurred. The value shall be the string "`Processing`", the string "`Complete`", or an error string starting with the value "`Error`".	Mandatory
percentComplete	JSON String	The value shall be an integer numeric value from 0 through 100.	Optional
mimetype	JSON String	MIME type of the value of the data object.	Mandatory
metadata	JSON Object	Metadata for the data object. This field includes any user and data system metadata specified in the request body *metadata* field, along with storage system metadata generated by the cloud storage system. See Chapter 16, "Metadata" for a further description of metadata.	Mandatory

Response Status:

HTTP Status	Description
201 Created	New data object was created.
202 Accepted	Data object is in the process of being created. Investigate *completionStatus* and *percentComplete* parameters to determine the current status of the operation.
400 Bad Request	Invalid parameter in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
404 Not Found	The specified container URI does not exist.
409 Conflict	The operation conflicts with a non-CDMI access protocol lock or may cause a state transition error on the server.

Example 9-11

POST to the container URI the data object contents

```
POST /MyContainer/ HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0

{
    "mimetype" : "text/plain",
    "metadata" : {

    },
    "value" : "This is the Value of this Data Object"
}
```

The response looks like:

```
HTTP/1.1 201 Created
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0
http://cloud.example.com/MyContainer/0000706D0010B84FAD185C425D8B537E

{
    "objectURI" : "/MyContainer/0000706D0010B84FAD185C425D8B537E",
    "objectID" : "0000706D0010B84FAD185C425D8B537E",
    "objectName" : "0000706D0010B84FAD185C425D8B537E",
    "parentURI" : "/MyContainer/",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/DataObject",
    "completionStatus" : "Complete",
    "mimetype" : "text/plain",
    "metadata" : {

    }
}
```

Example 9-12

POST to the object ID URI the data object contents

```
POST /cdmi_objectid/ HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0

{
    "mimetype" : "text/plain",
    "metadata" : {

    },
    "value" : "This is the Value of this Data Object"
}
```

The response looks like:

```
HTTP/1.1 201 Created
Location: http://cloud.example.com/cdmi_objectid/0000706D0010B84FAD185C425D8B537
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0

{
    "objectURI" : "/cdmi_objectid/0000706D0010B84FAD185C425D8B537E",
    "objectID" : "0000706D0010B84FAD185C425D8B537E",
    "objectName" : "0000706D0010B84FAD185C425D8B537E",
    "parentURI" : "/cdmi_objectid/",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/DataObject/",
    "completionStatus" : "Complete",
    "mimetype" : "text/plain",
    "metadata" : {

    }
}
```

9.9 Create (POST) a New Data Object (Non-CDMI Content Type)

Synopsis:

Creates a new data object as a child of the specified container URI.

```
POST <root URI>/<ContainerName>/
```

- `<root URI>` is the path to the CDMI cloud.
- `<ContainerName>` is zero or more intermediate containers that already exist.

Capabilities:

The following capabilities describe the supported operations that may be performed when creating a new data object:

- Support for the ability to create data objects through this operation is indicated by both the presence of the "cdmi_post_dataobject" capability the presence of the "cdmi_create_dataobject" capability in the specified container.

Request Headers:

Header	Type	Description	Requirement
Content-Type	Header String	Supply the MIME type of the object to be created.	Mandatory

Request Message Body:

The message body shall contain the contents (value) of the data object to be created.

Response Headers:

Header	Type	Description	Requirement
Location	Header String	The unique URL for the new data object as assigned by the system. In the absence of file name information from the client the system shall assign the URL in the form: `<root URL>/<ContainerName>/<ObjectID>`.	Mandatory

Response Message Body:

None specified.

Response Status:

HTTP Status	Description
201 Created	New data object was created.
400 Bad Request	Invalid parameter in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
404 Not Found	The specified container URI does not exist.

Example 9-13

POST to the container URI the data object contents

```
POST /MyContainer/ HTTP/1.1
Host: cloud.example.com
Content-Type: text/plain

<object contents>
```

The response looks like:

```
HTTP/1.1 201 Created
Location: http://cloud.example.com/MyContainer/0000706D0010B84FAD185C425D8B537E
```

Example 9-14

POST to the object ID URI the data object contents

```
POST /cdmi_objectid/ HTTP/1.1
Host: cloud.example.com
Content-Type: text/plain

<object contents>
```

The response looks like:

```
HTTP/1.1 201 Created
Location: http://cloud.example.com/cdmi_objectid/0000706D0010B84FAD185C425D8B537
```

9.10 Create (POST) a New Queue Object (CDMI Content Type)

Synopsis:

Creates a new queue object as a child of the specified container URI.

```
POST <root URI>/<ContainerName>/
POST <root URI>/cdmi_objectid/
```

- `<root URI>` is the path to the CDMI cloud.
- `<ContainerName>` is zero or more intermediate containers that already exist.

Once created, the queue object can also be accessed at `<root URI>/cdmi_objectid/<objectID>`.

Delayed Completion of Create:

On a create operation for a queue object, the server may return a response of `202 Accepted`. In this case, the object is in the process of being created. This response is particularly useful for long-running operations, for instance, copying a large number of queue items from a source URI. Such a response has the following implications:

- The server returns an OID along with the `202 Accepted`.
- With `202 Accepted`, the server implies that the following checks have passed:
  - User authorization for creating the object
  - User authorization for read access to any source object for move, copy, serialize, or deserialize
  - Availability of space to create the object or at least enough space to create a URI to report an error
- Future accesses to the URI created (or the object ID) will succeed modulo any delays due to use of eventual consistency.

The client performs GET operations to the URI to track the progress of the operation. In particular, the server returns two fields in its response body to indicate progress:

- A mandatory *completionStatus* text field contains either “`Processing`”, “`Complete`”, or an error string starting with the value "`Error`".

- An optional *percentComplete* field that indicates the percentage to which the last PUT has completed (0 to 100).

GET does not return any value for the object when *completionStatus* is not "`Complete`". When the final result of the create operation is an error, the URI is created with the *completionStatus* field set to the error message. It is the client's responsibility to delete the URI after the error has been noted.

Capabilities:

The following capabilities describe the supported operations that can be performed when creating a new queue object:

- Support for the ability to create queue objects through this operation is indicated by both the presence of the "cdmi_post_queue" capability the presence of the "cdmi_create_queue" capability in the specified container.
- If the object being created in the parent container is a reference, support for that ability is indicated by the presence of the "cdmi_create_reference" capability in the parent container.
- If the new queue object is a copy of an existing queue object, support for the ability to copy is indicated by the presence of the "cdmi_copy_queue" capability in the parent container.
- If the new queue object is the destination of a move, support for the ability to move the queue object is indicated by the presence of the "cdmi_move_queue" capability in the parent container.
- If the new queue object is the destination of a deserialize operation, support for the ability to deserialize the source data object is indicated by the presence of the "cdmi_deserialize_queue" capability in the parent container.

Request Headers:

Header	Type	Description	Requirement
Accept	Header String	"application/cdmi-queue"	Mandatory
Content-Type	Header String	"application/cdmi-queue"	Mandatory
X-CDMI-Specification-Version	Header String	A comma separated list of versions supported by the client, for example, "1.0, 1.5, 2.0".	Mandatory

Request Message Body:

Field Name	Type	Description	Requirement
metadata	JSON Object	Metadata for the queue object. • If this field is included when deserializing, serializing, copying, or moving a queue object, the value provided in this field shall replace the metadata from the source URI. • If this field is not included when deserializing, serializing, copying, or moving a queue object, the metadata from the source URI shall be used. • If this field is included when creating a new queue object by specifying a value, the value provided in this field shall be used as the metadata. • If this field is not included when creating a new queue object by specifying a value, an empty JSON object ("{}") will be assigned as the field value. • This field shall not be included when referencing a queue object.	Optional
domainURI	JSON String	URI of the owning domain. • If different from the parent domain, the user must have the "cross_domain" privilege. • If not specified, the parent domain will be used. • If creating an object by ID using `/cdmi_objectid/`, there is no parent container, so the domain must be specified.	Optional
deserialize	JSON String	URI of a serialized CDMI data object that will be deserialized to create the new queue object.	Optional*
copy	JSON String	URI of a CDMI queue object that will be copied into the new queue object.	Optional*
move	JSON String	URI of a CDMI queue object that will be copied into the new queue object. When the copy is successfully completed, the queue object at the source URI is removed.	Optional*
reference	JSON String	URI of a CDMI queue object to which a reference points. No other fields may be specified when creating a reference.	Optional*
deserializevalue	JSON String	A queue object serialized as specified in Chapter 15, "Serialization/Deserialization".	Optional*
*If present, only one of these parameters shall be specified in any given operation, and these fields are not persisted.			

Response Headers:

Header	Type	Description	Requirement
Content-Type	Header String	"application/cdmi-queue"	Mandatory

Header	Type	Description	Requirement
X-CDMI-Specification-Version	Header String	The server shall respond with the highest version supported by both the client and the server, for example, "1.0".	Mandatory
Location	Header String	The unique URL for the new queue object as assigned by the system. In the absence of file name information from the client, the system shall assign the URL in the form: `<root URL>/<ContainerName>/<ObjectID>`.	Mandatory

Response Message Body:

Field Name	Type	Description	Requirement
objectURI	JSON String	URI of the queue as specified in the request.	Mandatory
objectID	JSON String	Object ID of the object.	Mandatory
objectName	JSON String	Name of the object. • If the object has a path, the name shall be the last part of the path. • If the object does not have a path and is only accessible by ID, then the name shall be the Object ID of the object. • If an implementor chooses to always return the Object ID as the name of an object, even if one or more paths exists, the *parentURI* shall be set to `/cdmi_objectID/`.	Mandatory
parentURI	JSON String	URI for the parent object. • If the object has a path, the *parentURI* shall be the URI path to the parent object. OR • If the object does not have a path and is only accessible by ID, the *parentURI* shall be set to `/cdmi_objectid/`, and *objectName* shall be set to the Object ID of the object."	Mandatory
domainURI	JSON String	URI of the owning domain.	Mandatory
capabilitiesURI	JSON String	URI to the capabilities for the object. The capabilities URI returned is based on the object type and requested data system *metadata* fields.	Mandatory
completionStatus	JSON String	A string indicating if the object is still in the process of being created, and once the operation is complete, if it was created successfully or an error occurred. The value shall be the string "`Processing`", the string "`Complete`", or an error string starting with the value "`Error`".	Mandatory
percentComplete	JSON String	The value shall be an integer numeric value from 0 through 100.	Optional

Field Name	Type	Description	Requirement
metadata	JSON Object	Metadata for the queue object. This field includes any user and data system metadata specified in the request body *metadata* field, along with storage system metadata generated by the cloud storage system. See Chapter 16, "Metadata" for a further description of metadata.	Mandatory
queueValues	JSON String	A range of values enqueued in the queue. The first value goes up as items are deleted, and the second value goes up as items are enqueued, i.e., • Create: -> "" • Enqueue: -> "0-0" • Enqueue: -> "0-1" • Enqueue: -> "0-2" • Delete: -> "1-2" • Delete: -> "2-2" • Delete: -> "" • Enqueue: -> "3-3"	Mandatory

Response Status:

HTTP Status	Description
201 Created	New queue object was created.
202 Accepted	Queue object is in the process of being created. Investigate *completionStatus* and *percentComplete* parameters to determine the current status of the operation.
304 Not Modified	The operation conflicts because the queue object already exists.
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
409 Conflict	The operation conflicts with a non-CDMI access protocol lock or could cause a state transition error on the server.

Example 9-15

POST to the container URI the queue object name and metadata

```
POST /MyContainer/ HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-queue
X-CDMI-Specification-Version: 1.0

{
}
```

The response looks like:

```
HTTP/1.1 201 Created
Content-Type: application/cdmi-queue
X-CDMI-Specification-Version: 1.0
Location: http://cloud.example.com/MyContainer/0000706D0010B84FAD185C425D8B537E

{
    "objectURI" : "/MyContainer/0000706D0010B84FAD185C425D8B537E",
    "objectID" : "0000706D0010B84FAD185C425D8B537E",
    "objectName" : "0000706D0010B84FAD185C425D8B537E",
    "parentURI" : "/MyContainer/",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/Queue",
    "completionStatus" : "Complete",
    "metadata" : {
    },
    "queueValues" : ""
}
```

10 Domain Object Resource Operations

10.1 Overview

Domain objects represent the concept of administrative ownership of stored data within a CDMI storage system. A CDMI offering may include a hierarchy of domains that provide access to domain-related information within a CDMI context. This domain hierarchy is a series of CDMI objects that correspond to parent and child domains, with each domain corresponding to logical groupings of objects that are to be managed together. Domain measurement information about objects that are associated with each domain flow up to parent domains, facilitating billing and management operations that are typical for a cloud storage environment.

A CDMI URI may optionally include domains using the following form:

```
http://example.com/cdmi_domains/parent_domain/child_domain/
```

Domains are a special class of container that are created in the cdmi_domains container found in the root URI for the cloud storage system. If the "cdmi_create_domain" capability is present for the URI of a given domain, then the cloud storage system supports the ability to create child domains under the URI. If a cloud storage system supports domains, the cdmi_domains container shall be present.

10.1.1 Domain Metadata

The following domain-specific field shall be present for each domain:

Metadata Name	Type	Description	Requirement
cdmi_domain_enabled	JSON String	Indicates if the domain is enabled and specified at the time of creation. Values may be "true" or "false". • If a domain is disabled, the cloud storage system shall not permit any operations to be performed against any URI managed by that domain. • If this metadata item is not present at the time of domain creation, the value is set to "false".	Mandatory

10.1.2 Domain Summaries

Domain summaries provide summary measurement information about domain usage and billing. They are not intended to provide reporting functionality, but rather to provide a simple mechanism for gathering information about the storage operations that are associated with a domain. If supported, a domain summary container named "cdmi_domain_summary" shall be present under each domain container. Like any container, the domain summary sub-container may have standard storage system metadata, such as an Access Control List (ACL) that permits access to this information to be restricted.

Within each domain summary container are a series of domain summary data objects that are generated by the cloud storage system. The "yearly", "monthly", and "daily" containers of these data objects contain domain summary data objects corresponding to each year, month, and day, respectively. These containers are organized into the following structures:

```
http://example.com/cdmi_domains/domain/

http://example.com/cdmi_domains/domain/cdmi_domain_summary/

http://example.com/cdmi_domains/domain/cdmi_domain_summary/cumulative

http://example.com/cdmi_domains/domain/cdmi_domain_summary/daily/
```

```
http://example.com/cdmi_domains/domain/cdmi_domain_summary/daily/2009-07-01
http://example.com/cdmi_domains/domain/cdmi_domain_summary/daily/2009-07-02
http://example.com/cdmi_domains/domain/cdmi_domain_summary/daily/2009-07-03
http://example.com/cdmi_domains/domain/cdmi_domain_summary/monthly/
http://example.com/cdmi_domains/domain/cdmi_domain_summary/monthly/2009-07
http://example.com/cdmi_domains/domain/cdmi_domain_summary/monthly/2009-08
http://example.com/cdmi_domains/domain/cdmi_domain_summary/monthly/2009-10
http://example.com/cdmi_domains/domain/cdmi_domain_summary/yearly/
http://example.com/cdmi_domains/domain/cdmi_domain_summary/yearly/2009
http://example.com/cdmi_domains/domain/cdmi_domain_summary/yearly/2010
```

All time and duration values, unless otherwise specified, are in the [ISO-8601] extended representation (YYYY-MM-DDThh:mm:ss.ssssssZ). The full precision must be specified, the sub-second separator must be a ".", the Z UTC zone indicator must be included, and all timestamps must be in UTC time zone.

The "cumulative" summary data object covers the entire time period, from the time the domain is created to the time it is accessed. Each data object at the daily, monthly, and yearly level contains domain summary information for the time period specified, bounded by domain creation time and access time.

If a time period extends earlier than the domain creation time, the summary information includes the time from when the domain was created until the end of the time period. For example, if a domain was created on July 4, 2009, at noon, the daily summary "2009-07-04" would contain information from noon until midnight, the monthly summary "2009-07" would contain information from noon on July 4 until midnight on July 31, and the yearly summary "2009" would contain information from noon on July 4 until midnight on December 31.

If a time period starts after the time when the domain was created and ends earlier than the time of access, the summary data object contains complete information for that time period. For example, if a domain was created on July 4, 2009, and on July 10, the "2009-07-06" daily summary data object was accessed, it would contain information for the complete day.

If a time period ends after the current access time, the domain summary data object contains partial information from the start of the time period (or the time the domain was created) until the time of access. For example, if a domain was created on July 4, 2009, and at noon on July 10, the "2009-07-10" daily summary data object was accessed, it would contain information from the beginning of the day until noon.

The information in Table 7 shall be present within the contents of each domain summary object, which is in JSON representation.

Table 7 - Contents of Domain Summary Objects

Metadata Name	Type	Description	Requirement
cdmi_domainURI	JSON String	Domain name corresponding to the domain that is summarized.	Mandatory
cdmi_summary_start	JSON String	An [ISO-8601] time indicating the start of the time range that the summary information is presenting.	Mandatory
cdmi_summary_end	JSON String	An [ISO-8601] time indicating the end of the time range that the summary information is presenting.	Mandatory
cdmi_summary_objecthours	JSON String	The sum of the time each object belonging to the domain existed during the summary time period.	Optional

Table 7 - Contents of Domain Summary Objects

Metadata Name	Type	Description	Requirement
cdmi_summary_objectsmin	JSON String	The minimum number of objects belonging to the domain during the summary time period.	Optional
cdmi_summary_objectsmax	JSON String	The maximum number of objects belonging to the domain during the summary time period.	Optional
cdmi_summary_objectsaverage	JSON String	The average number of objects belonging to the domain during the summary time period.	Optional
cdmi_summary_puts	JSON String	The number of objects written to the domain.	Optional
cdmi_summary_gets	JSON String	The number of objects read from the domain.	Optional
cdmi_summary_bytehours	JSON String	The sum of the time each byte belonging to the domain existed during the summary time period.	Optional
cdmi_summary_bytesmin	JSON String	The minimum number of bytes belonging to the domain during the summary time period.	Optional
cdmi_summary_bytesmax	JSON String	The maximum number of bytes belonging to the domain during the summary time period.	Optional
cdmi_summary_bytesaverage	JSON String	The average number of bytes belonging to the domain .during the summary time period	Optional
cdmi_summary_writes	JSON String	The number of bytes written to the domain.	Optional
cdmi_summary_reads	JSON String	The number of bytes read from the domain.	Optional
cdmi_summary_charge	JSON String	A free-form monetary measurement of the charge for the use of the service to the user of the domain.	Optional
cdmi_summary_kwhours	JSON String	The sum of power consumed by the domain during the summary time period.	Optional
cdmi_summary_kwmin	JSON String	The minimum power consumed by the domain during the summary time period.	Optional
cdmi_summary_kwmax	JSON String	The maximum power consumed by the domain during the summary time period.	Optional
cdmi_summary_kwaverage	JSON String	The average power consumed by the domain during the summary time period.	Optional

An example of a daily domain summary object is as follows:

```
{
    "cdmi_domainURI" : "/cdmi_domains/MyDomain/",
    "cdmi_summary_start" : "2009-12-10T00:00:00",
    "cdmi_summary_end" : "2009-12-10T23:59:59",
    "cdmi_summary_objecthours" : "382239734",
    "cdmi_summary_puts" : "234234",
    "cdmi_summary_gets" : "489432",
    "cdmi_summary_bytehours" : "334895798347",
    "cdmi_summary_writes" : "7218368343",
    "cdmi_summary_reads" : "11283974933",
    "cdmi_summary_charge" : "4289.23 USD"
}
```

If the charge value is provided, the value is for the operational cost (excluding fixed fees) of service already performed and storage and bandwidth already consumed. Pricing of services is handled separately. Domain summary information may be extended by offerings to include additional metadata or domain reports beyond the metadata items standardized here.

10.1.3 Domain Membership

In cloud storage environments, just as domains are often created programmatically, domain user membership and credential mapping also shall be populated using such interfaces. By providing access to user membership, this capability enables self-enrollment, automatic provisioning, and other advanced self-service capabilities, either directly through the CDMI standard or through software systems that interface using CDMI.

The domain membership capability provides information about and allows the specification of end users and groups of users that are allowed to access the domain via CDMI and other access protocols. The concept of domain membership is not intended to replace or supplant ACLs, which form the fundamental basis for access control and object/container ownership, but rather to provide a single, unified place to map identities and credentials to principals used by ACLs within the context of a domain. It also provides a place for authentication mappings to external authentication providers, such as LDAP and AD, to be specified.

If supported, a domain membership container named "cdmi_domain_members" shall be present under each domain. Like any container, the domain membership container may have standard metadata, such as ACLs that permit access to this information to be restricted.

Within each domain membership container are a series of user objects that are specified through CDMI to define each user known to the domain. These objects are formatted into the following structure:

```
http://example.com/cdmi_domains/domain/

http://example.com/cdmi_domains/domain/cdmi_domain_members/

http://example.com/cdmi_domains/domain/cdmi_domain_members/john_doe

http://example.com/cdmi_domains/domain/cdmi_domain_members/john_smith
```

The domain membership container may also contain sub-containers with user data objects. User data objects in these sub-containers are treated the same as user data objects in the domain membership container, and no meaning is inferred from the sub-container name. This is allowed to create different access security relationships for groups of user objects (via container ACLs) and to allow delegation to common user lists.

Table 8 lists the domain settings that shall be present within each domain member user object:

Table 8 - Required Settings for Domain Member User Objects

Metadata Name	Type	Description	Requirement
cdmi_member_enabled	JSON String	Indicates if the member is enabled.	Mandatory
cdmi_member_type	JSON String	The type of member record. Values include "`user`" and "`delegation`".	Mandatory
cdmi_member_name	JSON String	This field contains the user name as presented by the client.	Mandatory
cdmi_member_credentials	JSON String	This field contains credentials to be matched against the credentials as presented by the client. If this field is not present, one or more delegations must be present and shall be used to resolve user credentials.	Optional
cdmi_member_principal	JSON String	This field indicates to which principal name (used in ACLs) the user is mapped. If this field is not present, one or more delegations must be present and shall be used to resolve the user principal.	Optional
cdmi_member_privileges	JSON Array of JSON Strings	This field contains a JSON list of special privileges associated with the user. The following privileges are defined: • "administrator". All ACL access checks are always successful. • "backup_operator". All read ACL access checks are always successful. • "cross_domain". Operations specifying a domain other than the domain of the parent object are permitted.	Mandatory
cdmi_member_groups	JSON Array of JSON Strings	This field contains a JSON array of group names to which the user belongs.	Optional

Table 9 lists the domain settings that shall be present within each domain member delegation object:

Table 9 - Required Settings for Domain Member Delegation Objects

Metadata Name	Type	Description	Requirement
cdmi_member_enabled	JSON String	Indicates if the member is enabled.	Mandatory
cdmi_member_type	JSON String	The type of member record. Values include "user" and "delegation".	Mandatory
cdmi_delegation_URI	JSON String	This field contains the URI of an external identity resolution provider (such as LDAP or Active Directory) or the URI of a Domain Membership Container. External delegations are expressed in the form of `ldap://` or `ad://`, etc.	Mandatory

An example of a domain membership object for a user is as follows:

```
{
    "cdmi_member_enabled" : "true",
    "cdmi_member_type" : "user",
    "cdmi_member_name" : "John Doe",
    "cdmi_member_credentials" : "p+5/oX1cmExfOIrUxhX1lw==",
    "cdmi_groups" : [
        "users"
    ],
    "cdmi_member_principal" : "jdoe",
    "cdmi_privileges" : [
        "administrator",
        "cross_domain"
    ]
}
```

An example of a domain membership object for a delegation is as follows:

```
{
    "cdmi_member_enabled" : "true",
    "cdmi_member_type" : "delegation",
    "cdmi_member_uri" : "/cdmi_accounts/MyAccount/",

}
```

When a transaction is initiated against a CDMI URI, the domain associated with the object at the URI (as indicated by the domainURI) is used as the authentication context. The user identity and credentials presented as part of the transaction are compared against the domain membership list to determine if the user is authorized within the domain and to resolve the user's principal. If resolved, the principal is then evaluated against the object's ACL to determine if the transaction is permitted.

When evaluating members within a domain, delegations are evaluated first, in any order, followed by user records, in any order. If there is at least one matching record and none of the matching records indicate that the user is disabled, the user is considered to be a member of the domain.

When a sub-domain is initially created, the membership container contains one member record, a delegation, where the delegation URI is set to the URI of the parent domain.

10.1.4 Domain Object Representations

The representations in this section are shown using JSON notation. A conforming implementation shall support the mandatory parameters and may support the optional parameters. The parameter fields may be specified or returned in any order. Both clients and servers shall support JSON representation.

10.2 Create a Domain Object (CDMI Content Type)

Synopsis:

Creates a new domain at the specified URI.

```
PUT <root URI>/cdmi_domains/<DomainName>/<NewDomainName>/
```

- `<root URI>` is the path to the CDMI cloud.
- `<DomainName>` is zero or more intermediate domains that already exist.

- <NewDomainName> is the name specified for the domain to be created.

Once created, the container may also be accessed at <root URI>/cdmi_objectid/<objectID>.

Capabilities:

The following capabilities describe the supported operations that may be performed when creating a new domain:

- Support for the ability to create a new domain object is indicated by the presence of the "cdmi_domain_create " capability in the parent container.
- If the new domain object is a copy of an existing domain object, support for the ability to copy is indicated by the presence of the "cdmi_copy_domain" capability in the source domain.
- If the new domain is the destination of a deserialize operation, support for the ability to deserialize the source data object serialization of a domain is indicated by the presence of the "cdmi_deserialize_domain" capability in the parent domain.

Request Headers:

Header	Type	Description	Requirement
Accept	Header String	"application/cdmi-domain".	Mandatory
Content-Type	Header String	"application/cdmi-domain".	Mandatory
X-CDMI-Specification-Version	Header String	A comma separated list of versions supported by the client, for example, "1.0, 1.5, 2.0".	Mandatory

Request Message Body:

Field Name	Type	Description	Requirement
metadata	JSON Object	Metadata for the domain. If this field is included when copying a domain object, the value provided shall be replace the metadata from the source URI. If this field is not specified, an empty JSON object ("{}") shall be assigned as the field value.	Optional
copy	JSON String	URI of a CDMI domain that shall be copied into the new domain, including all child domains and membership from the source domain.	Optional*
deserialize	JSON String	URI of a serialized CDMI data object that shall be deserialized to create the new domain, including all child objects inside the source serialized data object.	Optional*
deserializevalue	JSON String	A domain object serialized as specified in Chapter 15, "Serialization/Deserialization".	Optional*
*If present, only one of these parameters shall be specified in any given operation, and these fields are not persisted.			

Response Headers:

Header	Type	Description	Requirement
Content-Type	Header String	"application/cdmi-domain".	Mandatory
X-CDMI-Specification-Version	Header String	The server shall respond with the highest version supported by both the client and the server, for example, "1.0".	Mandatory

Response Message Body:

Field Name	Type	Description	Requirement
objectURI	JSON String	URI of the domain as specified in the request.	Mandatory
objectID	JSON String	Object ID of the domain.	Mandatory
objectName	JSON String	Name of the object. • If the object has a path, the name shall be the last part of the path. • If the object does not have a path and is only accessible by ID, then the name shall be the Object ID of the object. • If an implementor chooses to always return the Object ID as the name of an object, even if one or more paths exists, the *parentURI* shall be set to `/cdmi_objectID/`.	Mandatory
parentURI	JSON String	URI for the parent object. • If the object has a path, the *parentURI* shall be the URI path to the parent object. OR • If the object does not have a path and is only accessible by ID, the *parentURI* shall be set to `/cdmi_objectid/`, and *objectName* shall be set to the Object ID of the object."	Mandatory
domainURI	JSON String	URI of the owning domain. A domain object is always owned by itself.	Mandatory
capabilitiesURI	JSON String	URI to the capabilities for the object. The capabilities URI returned is based on the object type and requested data system *metadata* fields.	Mandatory
metadata	JSON Object	Metadata for the domain. This field includes any user and data system metadata specified in the request body *metadata* field, along with storage system metadata generated by the cloud storage system. See Chapter 16, "Metadata" for a further description of metadata.	Mandatory
childrenrange	JSON String	The range of the children returned in the *children* field.	Mandatory
children	JSON Array	Names of the children domains in the domain. Child containers end with "/".	Mandatory

Response Status:

HTTP Status	Description
201 Created	New domain object was created.
304 Not Modified	The operation conflicts because the domain already exists.
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
409 Conflict	The domain name already exists.

Example 10-1

PUT to the domain URI the domain name and metadata

```
PUT /cdmi_domains/MyDomain/ HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-domain
Content-Type: application/cdmi-domain
X-CDMI-Specification-Version: 1.0

{
    "metadata" : {

    }
}
```

The response looks like:

```
HTTP/1.1 201 Created
Content-Type: application/cdmi-domain
X-CDMI-Specification-Version: 1.0

{
    "objectURI" : "/cdmi_domains/MyDomain",
    "objectID" : "0000706D00101ADEBC119D1BFE98672A",
    "objectName" : "MyDomain",
    "parentURI" : "/cdmi_domains",
    "domainURI" : "/cdmi_domains/MyDomain",
    "capabilitiesURI" : "/cdmi_capabilities/Domain",
    "metadata" : {

    },
    "childrenrange" : "1-2",
    "children" : [
        "cdmi_domain_summary/",
        "cdmi_domain_members/"
    ]
}
```

10.3 Read a Domain Object (CDMI Content Type)

Synopsis:

Reads from an existing domain at the specified URI.

```
GET <root URI>/cdmi_domain/<DomainName>/<TheDomainName>/
GET <root URI>/cdmi_domain/<DomainName>/<TheDomainName>/
   ?<fieldname>;<fieldname>;...
GET <root URI>/cdmi_domain/<DomainName>/<TheDomainName>/?children:<range>;...
GET <root URI>/cdmi_domain/<DomainName>/<TheDomainName>/?metadata:<prefix>;...
```

- `<root URI>` is the path to the CDMI cloud.
- `<DomainName>` is zero or more parent domains.
- `<TheDomainName>` is the name specified for the domain to be read from.
- `<fieldname>` is the name of a field.
- `<range>` is a numeric range within the list of children.
- `<prefix>` is a matching prefix that returns all metadata items that start with the prefix value.

The object may also be accessed at `<root URI>/cdmi_objectid/<objectID>`.

Capabilities:

The following capabilities describe the supported operations that may be performed when reading an existing domain:

- Support for the ability to read the metadata of an existing domain object is indicated by the presence of the "cdmi_read_metadata" capability in the specified domain.
- Support for the ability to list the children of an existing domain object is indicated by the presence of the "cdmi_list_children" capability in the specified domain.

Request Headers:

Header	Type	Description	Requirement
Accept	Header String	"application/cdmi-domain". Shall contain a list of one or more of the five CDMI MIME types.	Mandatory
X-CDMI-Specification-Version	Header String	A comma separated list of versions supported by the client, for example, "1.0, 1.5, 2.0".	Mandatory

Request Message Body:

None specified.

Response Headers:

Header	Type	Description	Requirement
X-CDMI-Specification-Version	Header String	The server shall respond with the highest version supported by both the client and the server, for example, "1.0".	Mandatory
Content-Type	Header String	"application/cdmi-domain".	Mandatory
Location	Header String	The server shall respond with the URL that the reference points to if the object is a reference.	Mandatory

Response Message Body:

Field Name	Type	Description	Requirement
objectURI	JSON String	URI of the domain as specified in the request.	Mandatory
objectID	JSON String	Object ID of the domain.	Mandatory
objectName	JSON String	Name of the object. • If the object has a path, the name shall be the last part of the path. • If the object does not have a path and is only accessible by ID, then the name shall be the Object ID of the object. • If an implementor chooses to always return the Object ID as the name of an object, even if one or more paths exists, the *parentURI* shall be set to `/cdmi_objectID/`.	Mandatory
parentURI	JSON String	URI for the parent object. • If the object has a path, the *parentURI* shall be the URI path to the parent object. OR • If the object does not have a path and is only accessible by ID, the *parentURI* shall be set to `/cdmi_objectid/`, and *objectName* shall be set to the Object ID of the object."	Mandatory
domainURI	JSON String	URI of the owning domain. A domain object is always owned by itself.	Mandatory
capabilitiesURI	JSON String	URI to the capabilities for the object. The capabilities URI returned is based on the object type and requested data system *metadata* fields.	Mandatory
metadata	JSON Object	Metadata for the domain. This field includes any user and data system metadata specified in the request body *metadata* field, along with storage system metadata generated by the cloud storage system. See Chapter 16, "Metadata" for a further description of metadata.	Mandatory
childrenrange	JSON String	The range of the children returned in the *children* field.	Mandatory

Field Name	Type	Description	Requirement
children	JSON Array	The children of the domain. Sub-domains end with "/".	Mandatory

If individual fields are specified in the GET request, only these fields are returned in the result body. Optional fields that are requested but do not exist are omitted from the result body.

Response Status:

HTTP Status	Description
200 OK	Domain contents in response.
302 Found	The URI is a reference to another URI.
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
404 Not Found	A domain was not found at the specified URI.
406 Not Acceptable	The server is unable to provide the object in the content-type specified in the Accept header.

Example 10-2

GET to the domain URI to read all the fields of the domain

```
GET /MyDomain/ HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-domain
X-CDMI-Specification-Version: 1.0
```

The response looks like:

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-domain
X-CDMI-Specification-Version: 1.0

{
    "objectURI" : "/cdmi_domains/MyDomain",
    "objectID" : "0000706D00101ADEBC119D1BFE98672A",
    "objectName" : "MyDomain",
    "parentURI" : "/cdmi_domains",
    "domainURI" : "/cdmi_domains/MyDomain",
    "capabilitiesURI" : "/cdmi_capabilities/Domain",
    "metadata" : {

    },
    "childrenrange" : "0-1",
    "children" : [
        "cdmi_domain_summary/",
        "cdmi_domain_members/"
    ]
}
```

Example 10-3

GET to the domain URI to read all the *parentURI* and children of the domain

```
GET /MyDomain/?parentURI;children HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-domain
X-CDMI-Specification-Version: 1.0
```

The response looks like:

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-domain
X-CDMI-Specification-Version: 1.0

{
    "parentURI" : "/cdmi_domains",
    "children" : [
        "cdmi_domain_summary/",
        "cdmi_domain_members/"
    ]
}
```

Example 10-4

GET to the domain URI to read the first two children of the domain

```
GET /MyDomain/?childrenrange;children:0-1 HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-domain
X-CDMI-Specification-Version: 1.0
```

The response looks like:

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-domain
X-CDMI-Specification-Version: 1.0

{
    "childrenrange" : "0-1",
    "children" : [
        "cdmi_domain_summary/",
        "cdmi_domain_members/"
    ]
}
```

10.4 Update a Domain (CDMI Content Type)

Synopsis:

Updates an existing domain at the specified URI.

```
PUT <root URI>/cdmi_domains/<DomainName>/<TheDomainName>/
PUT <root URI>/cdmi_domains/<DomainName>/<TheDomainName>/?metadata
```

- `<root URI>` is the path to the CDMI cloud.
- `<DomainName>` is zero or more parent domains.

- `<TheDomainName>` is the name specified for the domain to be updated.

The object may also be accessed at `<root URI>/cdmi_objectid/<objectID>`, and an update shall not result in a change to the Object ID.

Capabilities:

The following capabilities describe the supported operations that may be performed when updating an existing domain:

- Support for the ability to modify the metadata of an existing domain object is indicated by the presence of the "cdmi_modify_metadata" capability in the specified domain.

Request Headers:

Header	Type	Description	Requirement
Accept	Header String	"application/cdmi-domain".	Mandatory
Content-Type	Header String	"application/cdmi-domain".	Mandatory
X-CDMI-Specification-Version	Header String	A comma separated list of versions supported by the client, for example, "1.0, 1.5, 2.0".	Mandatory

Request Message Body:

Field Name	Type	Description	Requirement
metadata	JSON Object	Metadata for the container. If present, this replaces the existing metadata. See Chapter 16, "Metadata" for a further description of metadata.	Optional

Response Headers:

Header	Type	Description	Requirement
Location	Header String	The server shall respond with the URL that the reference points to if the object is a reference.	Mandatory

Response Message Body:

None specified.

Response Status:

HTTP Status	Description
200 OK	Domain was updated.
302 Found	The URI is a reference to another URI.
400 Bad Request	Invalid parameter or field names in the request.

HTTP Status	Description
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
404 Not Found	An update was attempted on a domain that does not exist.
409 Conflict	The operation conflicts with a non-CDMI access protocol lock or may cause a state transition error on the server.

Example 10-5

PUT to the domain URI to set new field values

```
PUT /cdmi_domains/myDomain/ HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-domain
X-CDMI-Specification-Version: 1.0

{
    "metadata" : {
        "test" : "value"
    }
}
```

The response looks like:

```
HTTP/1.1 200 OK
```

10.5 Delete a Domain (CDMI Content Type)

Synopsis:

Deletes an existing domain, and all contained children domains under the specified URI.

```
DELETE <root URI>/cdmi_domains/<DomainName>/<TheDomainName>/
```

- `<root URI>` is the path to the CDMI cloud.
- `<DomainName>` is zero or more parent domains.
- `<TheDomainName>` is the name specified for the domain to be deleted.

The object may also be accessed at `<root URI>/cdmi_objectid/<objectID>`.

Capabilities:

The following capabilities describe the supported operations that may be performed when deleting an existing container:

- Support for the ability to delete an existing data object is indicated by the presence of the "cdmi_domain_delete " capability in the specified domain.

Request Headers:

Header	Type	Description	Requirement
X-CDMI-Specification-Version	Header String	A comma separated list of versions supported by the client, for example, "1.0, 1.5, 2.0".	Mandatory

Request Message Body:

Field Name	Type	Description	Requirement
newDomain	JSON String	The new domainURI that shall own the objects being deleted, including the objects that the domain owns and any children domains. This shall be a valid domain URI.	Mandatory

Response Headers:

None specified.

Response Message Body:

None specified.

Response Status:

HTTP Status	Description
204 No Content	Domain was successfully deleted.
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
404 Not Found	The resource specified was not found.
409 Conflict	The domain may not be deleted (may be immutable), or the specified newDomain URI is invalid or unusable.

Example 10-6

DELETE to the domain URI

```
DELETE /cdmi_domains/MyDomain/ HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-domain
X-CDMI-Specification-Version: 1.0

{
    "newDomain" : "/cdmi_domains/secondDomain"
}
```

The response looks like:

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

11 Queue Object Resource Operations

11.1 Overview

Queues are a special class of container object and are used to provide first-in, first-out access when storing and retrieving data. A queue writer PUTs objects to the queue, and a queue reader GETs objects from the queue, acknowledging the receipt of the last object that it received. Queuing provides a simple mechanism for one or more writers to send data to a single reader in a reliable way. If supported by the cloud storage system, cloud clients create the queue objects by using the same mechanism used to create data objects.

Every queue object has a parent object from which the queue object inherits data system metadata. For example, the "receipts.queue" queue object stored at "http://cloud.example.com/finance/receipts.queue" would inherit data system metadata from its parent container, "finance".

11.1.1 Queue Object Metadata

Queue object metadata may also include arbitrary user-supplied metadata and data system metadata, as specified in Chapter 16, "Metadata".

11.1.2 Queue Object Addressing

Each queue object is addressed via one or more unique URIs, and all operations may be performed through any of these URIs.

11.1.3 Queue Object Representations

The representations in this section are shown using JSON notation. A conforming implementation shall support the mandatory parameters and may support the optional parameters. The parameter fields may be specified or returned in any order. Both clients and servers shall support JSON representation.

11.2 Create a Queue Object (CDMI Content Type)

Synopsis:

Creates a new queue object at the specified URI.

```
PUT <root URI>/<ContainerName>/<QueueName>
```

- `<root URI>` is the path to the CDMI cloud.
- `<ContainerName>` is zero or more intermediate containers that already exist.
- `<QueueName>` is the name specified for the queue object to be created.

Once created, the object may also be accessed at `<root URI>/cdmi_objectid/<objectID>`.

Delayed Completion of Create:

On a create operation for a queue, the server may return a response of `202 Accepted`. In this case, the queue is in the process of being created. This response is particularly useful for long-running operations, for instance, for copying a large data object from a source URI. Such a response has the following implications:

- The server returns an object identifier (OID) along with the `202 Accepted`.
- With `202 Accepted`, the server implies that the following checks have passed:
  - User authorization for creating the queue object
  - User authorization for read access to any source object for move, copy, serialize, or deserialize
  - Availability of space to create the queue or at least enough space to create a URI to report an error
- Future accesses to the URI created (or the object ID) shall succeed modulo any delays due to use of eventual consistency.

The client performs GET operations to the URI to track the progress of the operation. In particular, the server returns two fields in its response body to indicate progress:

- A *completionStatus* text field contains either “`Processing`”, “`Complete`”, or an error string starting with the value "`Error`".
- An optional *percentComplete* field that indicates the percentage to which the last PUT has completed (0 to 100)

GET does not return any value for the object when completionStatus is not `Complete`. When the final result of the create operation is an error, the URI is created with the *completionStatus* field set to the error message. It is the client's responsibility to delete the URI after the error has been noted.

Capabilities:

The following capabilities describe the supported operations that may be performed when creating a new queue object:

- Support for the ability to create a new queue object is indicated by the presence of the "cdmi_create_queue" capability in the parent container.
- If the object being created in the parent container is a reference, support for that ability is indicated by the presence of the "cdmi_create_reference" capability in the parent container.
- If the new queue object is a copy of an existing queue object, support for the ability to copy is indicated by the presence of the "cdmi_copy_queue" capability in the patent container.
- If the new queue object is the destination of a move, support for the ability to move the queue object is indicated by the presence of the "cdmi_move_queue" capability in the parent container.
- If the new queue object is the destination of a deserialize operation, support for the ability to deserialize the source data object is indicated by the presence of the "cdmi_deserialize_queue" capability in the parent container.

Request Headers:

Header	Type	Description	Requirement
Accept	Header String	"application/cdmi-queue".	Mandatory

Header	Type	Description	Requirement
Content-Type	Header String	"application/cdmi-queue".	Mandatory
X-CDMI-Specification-Version	Header String	A comma separated list of versions supported by the client, for example, "1.0, 1.5, 2.0".	Mandatory

Request Message Body:

Field Name	Type	Description	Requirement
metadata	JSON Object	Metadata for the queue object. • If this field is included when deserializing, serializing, copying, or moving a queue object, the value provided in this field shall replace the metadata from the source URI. • If this field is not included when deserializing, serializing, copying, or moving a queue object, the metadata from the source URI shall be used. • If this field is included when creating a new queue object by specifying a value, the value provided in this field shall be used as the metadata. • If this field is not included when creating a new queue object by specifying a value, an empty JSON object ("{}") shall be assigned as the field value. • This field shall not be included when referencing a queue object.	Optional
domainURI	JSON String	URI of the owning domain. • If different from the parent domain, the user shall have the "cross_domain" privilege. • If not specified, the parent domain shall be used.	Optional
deserialize	JSON String	URI of a serialized CDMI data object that shall be deserialized to create the new queue object.	Optional*
copy	JSON String	URI of a CDMI queue object that shall be copied into the new queue object.	Optional*
move	JSON String	URI of a CDMI queue object that shall be copied into the new queue object. When the copy is successfully completed, the queue object at the source URI is removed.	Optional*
reference	JSON String	URI of a CDMI queue object to which a reference points. No other fields may be specified when creating a reference.	Optional*
deserializevalue	JSON String	A queue object serialized as specified in Chapter 15, "Serialization/Deserialization".	Optional*
*If present, only one of these parameters shall be specified in any given operation, and these fields are not persisted.			

Response Headers:

Header	Type	Description	Requirement
Content-Type	Header String	"application/cdmi-queue".	Mandatory
X-CDMI-Specification-Version	Header String	The server shall respond with the highest version supported by both the client and the server, for example, "1.0".	Mandatory

Response Message Body:

Field Name	Type	Description	Requirement
objectURI	JSON String	URI of the queue as specified in the request.	Mandatory
objectID	JSON String	Object ID of the object.	Mandatory
objectName	JSON String	Name of the object. • If the object has a path, the name shall be the last part of the path. • If the object does not have a path and is only accessible by ID, then the name shall be the Object ID of the object. • If an implementor chooses to always return the Object ID as the name of an object, even if one or more paths exists, the *parentURI* shall be set to `/cdmi_objectID/`.	Mandatory
parentURI	JSON String	URI for the parent object. • If the object has a path, the *parentURI* shall be the URI path to the parent object. OR • If the object does not have a path and is only accessible by ID, the *parentURI* shall be set to `/cdmi_objectid/`, and *objectName* shall be set to the Object ID of the object."	Mandatory
domainURI	JSON String	URI of the owning domain.	Mandatory
capabilitiesURI	JSON String	URI to the capabilities for the object. The capabilities URI returned is based on the object type and requested data system *metadata* fields.	Mandatory
completionStatus	JSON String	A string indicating if the object is still in the process of being created, and once the operation is complete, if it was created successfully or an error occurred. The value shall be the string "`Processing`", the string "`Complete`", or an error string starting with the value "`Error`".	Mandatory
percentComplete	JSON String	The value shall be an integer numeric value from 0 through 100.	Optional

Field Name	Type	Description	Requirement
metadata	JSON Object	Metadata for the queue object. This field includes any user and data system metadata specified in the request body *metadata* field, along with storage system metadata generated by the cloud storage system. See Chapter 16, "Metadata" for a further description of metadata.	Mandatory
queueValues	JSON String	A range of values enqueued in the queue. The first value goes up as items are deleted, and the second value goes up as items are enqueued, i.e., • Create: -> "" • Enqueue: -> "0-0" • Enqueue: -> "0-1" • Enqueue: -> "0-2" • Delete: -> "1-2" • Delete: -> "2-2" • Delete: -> "" • Enqueue: -> "3-3"	Mandatory

Response Status:

HTTP Status	Description
201 Created	New queue object was created.
202 Accepted	Queue object is in the process of being created. Investigate *completionStatus* and *percentComplete* parameters to determine the current status of the operation.
304 Not Modified	The operation conflicts because the queue object already exists.
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
409 Conflict	The operation conflicts with a non-CDMI access protocol lock or may cause a state transition error on the server.

Example 11-1

PUT to the container URI the queue object name and contents

```
PUT /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-queue
Content-Type: application/cdmi-queue
X-CDMI-Specification-Version: 1.0

{
    "metadata" : {

    }
}
```

The response looks like:

```
HTTP/1.1 201 Created
Content-Type: application/cdmi-queue
X-CDMI-Specification-Version: 1.0

{
    "objectURI" : "/MyContainer/MyQueue",
    "objectID" : "0000706D00101ADEBC119D1BFE98672A",
    "objectName" : "MyQueue",
    "parentURI " : "/MyContainer/",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/Queue",
    "completionStatus" : "Complete",
    "metadata" : {

    },
    "queuevalues" : ""
}
```

11.3 Read a Queue Object (CDMI Content Type)

Synopsis:

Reads from an existing queue object at the specified URI.

```
GET <root URI>/<ContainerName>/<QueueName>
GET <root URI>/<ContainerName>/<QueueName>?<fieldname>;<fieldname>;...
GET <root URI>/<ContainerName>/<QueueName>?value:<range>;...
GET <root URI>/<ContainerName>/<QueueName>?metadata:<prefix>;...
GET <root URI>/<ContainerName>/<QueueName>?values:<count>
```

- `<root URI>` is the path to the CDMI cloud.
- `<ContainerName>` is zero or more intermediate containers.
- `<QueueName>` is the name of the queue object to be read from.
- `<fieldname>` is the name of a field.
- `<range>` is a byte range within the queue element value field.
- `<prefix>` is a matching prefix that returns all metadata items that start with the prefix value.
- `<count>` is the number of values to be retrieved from the queue. If more queue entries are requested to be retrieved than exist in the queue, the count is considered equal to the number of entries in the queue.

The object may also be accessed at `<root URI>/cdmi_objectid/<objectID>`.

Capabilities:

The following capabilities describe the supported operations that may be performed when reading an existing queue object:

- Support for the ability to read the metadata of an existing queue object is indicated by the presence of the "cdmi_read_metadata" capability in the specified queue.

- Support for the ability to read the value of an existing queue object is indicated by the presence of the "cdmi_read_value" capability in the specified queue.

Request Headers:

Header	Type	Description	Requirement
Accept	Header String	"application/cdmi-queue". Shall contain a list of one or more of the five CDMI MIME types.	Mandatory
X-CDMI-Specification-Version	Header String	A comma separated list of versions supported by the client, for example, "1.0, 1.5, 2.0".	Mandatory

Request Message Body:

None specified.

Response Headers:

Header	Type	Description	Requirement
X-CDMI-Specification-Version	Header String	The server shall respond with the highest version supported by both the client and the server, for example, "1.0".	Mandatory
Content-Type	Header String	"application/cdmi-queue".	Mandatory
Location	Header String	The server shall respond with the URL that the reference points to if the object is a reference.	Mandatory

Response Message Body:

Field Name	Type	Description	Requirement
objectURI	JSON String	URI of the queue object as specified in the request.	Mandatory
objectID	JSON String	Object ID of the object.	Mandatory
objectName	JSON String	Name of the object. • If the object has a path, the name shall be the last part of the path. • If the object does not have a path and is only accessible by ID, then the name shall be the Object ID of the object. • If an wimplementor chooses to always return the Object ID as the name of an object, even if one or more paths exists, the *parentURI* shall be set to `/cdmi_objectID/`.	Mandatory

Field Name	Type	Description	Requirement
parentURI	JSON String	URI for the parent object. • If the object has a path, the *parentURI* shall be the URI path to the parent object. OR • If the object does not have a path and is only accessible by ID, the *parentURI* shall be set to `/cdmi_objectid/`, and *objectName* shall be set to the Object ID of the object."	Mandatory
domainURI	JSON String	URI of the owning domain.	Mandatory
capabilitiesURI	JSON String	URI to the capabilities for the object. The capabilities URI returned is based on the object type and requested data system *metadata* fields.	Mandatory
completionStatus	JSON String	A string indicating if the object is still in the process of being created, and once the operation is complete, if it was created successfully or an error occurred. The value shall be the string "`Processing`", the string "`Complete`", or an error string starting with the value "`Error`".	Mandatory
percentComplete	JSON String	The value shall be an integer numeric value from 0 through 100.	Optional
metadata	JSON Object	Metadata for the queue object. This field includes any user and data system metadata specified in the request body *metadata* field, along with storage system metadata generated by the cloud storage system. See Chapter 16, "Metadata" for a further description of metadata.	Mandatory
queueValues	JSON String	A range of values enqueued in the queue. The first value goes up as items are deleted, and the second value goes up as items are enqueued, i.e., • Create: -> "" • Enqueue: -> "0-0" • Enqueue: -> "0-1" • Enqueue: -> "0-2" • Delete: -> "1-2" • Delete: -> "2-2" • Delete: -> "" • Enqueue: -> "3-3"	Mandatory
mimetype	JSON Array of JSON Strings	MIME type of the oldest queue value. This field shall only be provided when *completionStatus* is `Complete`, and when there are one or more values enqueued. If two or more values are requested, the MIME types of the values are returned, each corresponding to the value in the same position in the JSON array.	Optional

Field Name	Type	Description	Requirement
valuerange	JSON Array of JSON Strings	The range of bytes of the value returned in the *value* field. This field shall only be provided when *completionStatus* is `Complete`, and when there are one or more values enqueued. If two or more values are requested, the value range for the values are returned, each corresponding to the value in the same position in the JSON array.	Optional
value	JSON Array of JSON Strings	Value of the oldest enqueued item. This field shall only be provided when *completionStatus* is `Complete` and when there are one or more values enqueued. If two or more values are requested, the value for each entry are returned in a JSON array, from oldest to newest.	Optional

If individual fields are specified in the GET request, only these fields are returned in the result body. Optional fields that are requested but do not exist are omitted from the result body.

Response Status:

HTTP Status	Description
200 OK	Valid response is enclosed.
302 Found	The URI is a reference to another URI.
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
404 Not Found	A queue object was not found at the specified URI.
406 Not Acceptable	The server is unable to provide the object in the content-type specified in the Accept header.

Example 11-2

GET to the queue object URI to read all fields of the queue object

```
GET /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-queue
X-CDMI-Specification-Version: 1.0
```

The response looks like:

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-queue
X-CDMI-Specification-Version: 1.0

{
    "objectURI" : "/MyContainer/MyQueue",
    "objectID" : "0000706D00101ADEBC119D1BFE98672A",
    "objectName" : "MyQueue",
    "parentURI" : "/MyContainer/",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/Queue/",
    "completionStatus" : "Complete",
    "metadata" : {

    },
    "queueValues" : "1-2",
    "mimetype" : [
        "text/plain"
    ],
    "valuerange" : [
        "0-19"
    ],
    "value" : [
        "First Enqueued Value"
    ]
}
```

Example 11-3

GET to the queue object URI to read the value and queue items of the queue object

```
GET /MyContainer/MyQueue?value;queueValues HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-queue
X-CDMI-Specification-Version: 1.0
```

The response looks like:

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-queue
X-CDMI-Specification-Version: 1.0

{
    "queueValues" : "1-2",
    "value" : [
        "First Enqueued Value"
    ]
}
```

Example 11-4

GET to the queue object URI to read the first five bytes of the value of the queue object

```
GET /MyContainer/MyQueue?value:0-5 HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-queue
X-CDMI-Specification-Version: 1.0
```

The response looks like:

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-queue
X-CDMI-Specification-Version: 1.0

{
    "value" : [
        "First"
    ]
}
```

Example 11-5

GET to the queue object URI to read two values of the queue object

```
GET /MyContainer/MyQueue?mimetype;valuerange;values:2 HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-queue
X-CDMI-Specification-Version: 1.0
```

The response looks like:

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-queue
X-CDMI-Specification-Version: 1.0

{
    "mimetype" : [
        "text/plain",
        "text/plain"
    ],
    "valuerange" : [
        "0-19",
        "0-20"
    ],
    "value" : [
        "First Enqueued Value",
        "Second Enqueued Value"
    ]
}
```

11.4 Update a Queue Object (CDMI Content Type)

Synopsis:

Updates an existing queue at the specified URI.

```
PUT <root URI>/<ContainerName>/<QueueName>
PUT <root URI>/<ContainerName>/<QueueName>?metadata
```

- `<root URI>` is the path to the CDMI cloud.
- `<ContainerName>` is zero or more intermediate containers.
- `<QueueName>` is the name of the queue to be updated.

The object may also be accessed at `<root URI>/cdmi_objectid/<objectID>`, and an update shall not result in a change to the Object ID.

Capabilities:

The following capabilities describe the supported operations that may be performed when updating an existing queue object:

- Support for the ability to modify the metadata of an existing queue object is indicated by the presence of the "cdmi_modify_metadata" capability in the specified queue.

Request Headers:

Header	Type	Description	Requirement
Accept	Header String	"application/cdmi-queue".	Mandatory
Content-Type	Header String	"application/cdmi-queue ".	Mandatory
X-CDMI-Specification-Version	Header String	A comma separated list of versions supported by the client, for example, "1.0, 1.5, 2.0".	Mandatory

Request Message Body:

Field Name	Type	Description	Requirement
metadata	JSON Object	Metadata for the queue object. If present this replaces the existing metadata. See Chapter 16, "Metadata" for a further description of metadata.	Optional
domainURI	JSON String	URI of the owning domain. If different from the parent domain, the user shall have the "cross_domain" privilege. If not specified, the parent domain shall be used.	Optional

Response Headers:

Header	Type	Description	Requirement
Location	Header String	The server shall respond with the URL that the reference points to if the object is a reference.	Mandatory

Response Message Body:

None specified.

Response Status:

HTTP Status	Description
200 OK	New metadata and/or content accepted.
302 Found	The URI is a reference to another URI.

HTTP Status	Description
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
404 Not Found	An update was attempted on an object which does not exist.
409 Conflict	The operation conflicts with a non-CDMI access protocol lock or may cause a state transition error on the server.

Example 11-6

PUT to the queue object URI to set new metadata

```
PUT /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-queue
X-CDMI-Specification-Version: 1.0

{
    "metadata" : {

    }
}
```

The response looks like:

```
HTTP/1.1 200 OK
```

11.5 Delete a Queue Object (CDMI Content Type)

Synopsis:

Deletes an existing queue object and all enqueued values at the specified URI.

```
DELETE <root URI>/<ContainerName>/<QueueName>
```

- `<root URI>` is the path to the CDMI cloud.
- `<ContainerName>` is zero or more intermediate containers.
- `<QueueName>` is the name of the queue object to be deleted.

The object may also be accessed at `<root URI>/cdmi_objectid/<objectID>`.

Capabilities:

The following capabilities describe the supported operations that may be performed when deleting an existing data object:

- Support for the ability to delete an existing queue object is indicated by the presence of the "cdmi_delete_queue" capability in the specified queue.

Request Headers:

Header	Type	Description	Requirement
X-CDMI-Specification-Version	Header String	A comma separated list of versions supported by the client, for example, "1.0, 1.5, 2.0".	Mandatory

Request Message Body:

None specified.

Response Headers:

None specified.

Response Message Body:

None specified.

Response Status:

HTTP Status	Description
204 No Content	Queue object was successfully deleted.
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
404 Not Found	The resource specified was not found.
409 Conflict	The queue object may not be deleted (may be immutable).

Example 11-7

DELETE to the queue URI

```
DELETE /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
X-CDMI-Specification-Version: 1.0
```

The response looks like:

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

11.6 Enqueue a New Queue Value (CDMI Content Type)

Synopsis:

Enqueues a new data value in an existing Queue at the specified container URI.

```
POST <root URI>/<ContainerName>/<QueueName>
```

- `<root URI>` is the path to the CDMI cloud.
- `<ContainerName>` is zero or more intermediate containers that already exist.
- `<QueueName>` is the name of the queue object to be read from.

Capabilities:

The following capabilities describe the supported operations that may be performed when enqueuing a new value into an existing queue object:

- Support for the ability to modify the value of an existing queue object is indicated by the presence of the "cdmi_modify_value" capability in the specified queue.

Request Headers:

Header	Type	Description	Requirement
Content-Type	Header String	"application/cdmi-queue".	Mandatory
X-CDMI-Specification-Version	Header String	A comma separated list of versions supported by the client, for example, "1.0, 1.5, 2.0".	Mandatory

Request Message Body:

Field Name	Type	Description	Requirement
mimetype	JSON Array of JSON Strings	MIME type of the data contained within the *value* field of the data object. This field shall not be included when copying or moving a data object. • If this field is not specified, the value of "text/plain" shall be assigned as the field value. • If this field is a JSON array, the same number of array elements shall be present in the *value* field, and the mimetype shall be associated with the value in the corresponding position.	Optional
copy	JSON String	URI of a CDMI data object or queue from which the value shall be copied and enqueued.	Optional*
move	JSON String	URI of a CDMI data object or queue from which the value shall be enqueued, then removing the data object or queue value at the source URI upon the successful completion of the enqueue.	Optional*
value	JSON Array of JSON Strings	JSON-encoded data. If this field is not included, an empty JSON String ("") shall be assigned as the field value. Binary data shall be escaped as per the JSON escaping rules described in [RFC4627]. If this field is a JSON array, each item in the array shall be considered to be an independent entry being enqueued.	Optional*
*Only one of these parameters shall be specified in any given operation.			

Response Headers:

None specified.

Response Message Body:

None specified.

Response Status:

HTTP Status	Description
201 Created	New queue value was enqueued.
400 Bad Request	Invalid parameter in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
404 Not Found	The specified container URI does not exist.
409 Conflict	The operation conflicts with a non-CDMI access protocol lock or may cause a state transition error on the server.

Example 11-8

POST to the queue URI a new value

```
POST /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-queue
X-CDMI-Specification-Version: 1.0

{
    "mimetype" : [
        "text/plain"
    ],
    "value" : [
        "Value to Enqueue"
    ]
}
```

The response looks like:

```
HTTP/1.1 201 Created
```

Example 11-9

POST to the queue URI to copy an existing value

```
POST /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0

{
    "copy" : "/MyContainer/MyDataObject"
}
```

The response looks like:

```
HTTP/1.1 201 Created
```

Example 11-10

POST to the queue URI to transfer twenty values from another queue

```
POST /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0

{
    "move" : "/MyContainer/FirstQueue;values:20"
}
```

The response looks like:

```
HTTP/1.1 201 Created
```

Example 11-11

POST to the queue URI two new values

```
POST /MyContainer/MyQueue
HTTP/1.1 Host: cloud.example.com
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0

{
    "mimetype" : [
        "text/plain",
        "text/plain"
    ],
    "value" : [
        "First",
        "Second"
    ]
}
```

The response looks like:

```
HTTP/1.1 201 Created
```

11.7 Delete a Queue Value (CDMI Content Type)

Synopsis:

Deletes the oldest enqueued value in an existing queue object at the specified URI.

```
DELETE <root URI>/<ContainerName>/<QueueName>?value
DELETE <root URI>/<ContainerName>/<QueueName>?values:<count>
```

- `<root URI>` is the path to the CDMI cloud.
- `<ContainerName>` is zero or more intermediate containers.

- <QueueName> is the name of the queue object to be deleted.
- <count> is the number of values to be removed from the queue. If more queue entries are requested to be deleted than exist in the queue, the count is considered equal to the number of entries in the queue.

The object may also be accessed at <root URI>/cdmi_objectid/<objectID>.

Capabilities:

The following capabilities describe the supported operations that may be performed when deleting an existing data object:

- Support for the ability to modify the value of an existing queue object is indicated by the presence of the "cdmi_modify_value" capability in the specified queue.

Request Headers:

Header	Type	Description	Requirement
X-CDMI-Specification-Version	Header String	A comma separated list of versions supported by the client, for example, "1.0, 1.5, 2.0".	Mandatory

Request Message Body:

None specified.

Response Headers:

None specified.

Response Message Body:

None specified.

Response Status:

HTTP Status	Description
204 No Content	Queue object was successfully deleted.
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
404 Not Found	The resource specified was not found.
409 Conflict	The queue object may not be deleted (may be immutable).

Example 11-12

DELETE to the queue URI value to access the next enqueued value

```
DELETE /MyContainer/MyQueue?value HTTP/1.1
Host: cloud.example.com
X-CDMI-Specification-Version: 1.0
```

The response looks like:

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

12 Capability Object Resource Operations

12.1 Overview

Capability objects are a special class of container object that allow a CDMI client to discover what subset of the CDMI standard is implemented by a CDMI provider.

For each URI in a CDMI system, the set of interactions that the system is capable of performing against that URI are described by the presence of named "capabilities". Each capability present for a given URI indicates what functionality the cloud storage system will allow against that URI. Capabilities are always static.

It is important to note that capabilities may differ from the operations permitted by an Access Control List (ACL) associated with a given URI, for example, a read-only cloud may not permit write access to a container or object, despite the presence of an ACL allowing write access.

Cloud clients may use capabilities to discover what operations are supported. If an operation is attempted to be performed against a CDMI object that does not have a corresponding capability, an HTTP 405 status code shall be returned to the client. All CDMI-compliant cloud storage systems shall implement the ability to list capabilities, but support for all other capabilities is optional.

Every CDMI data object, container domain, and queue shall have a *capabilitiesURI* field that contains a valid URI that points to a capabilities object. Within the capabilities object, the name of each capability confers a specific meaning that has been agreed to between the cloud storage provider and the cloud storage consumer, and the capabilities defined as part of the CDMI standard are described later in this section. Capabilities not listed in this standard shall not begin with the prefix "cdmi_", but are otherwise permitted to allow cloud storage system wimplementors to add additional capabilities.

The base set of CDMI capabilities are based on the operations defined in the previous sections, with additional cloud-specific capabilities added based on use cases for standard cloud storage. The hierarchy of capabilities (see Figure 8) shows the hierarchy of capabilities in an offering and shows how the capabilitiesURI links data objects and containers into the capabilities tree.

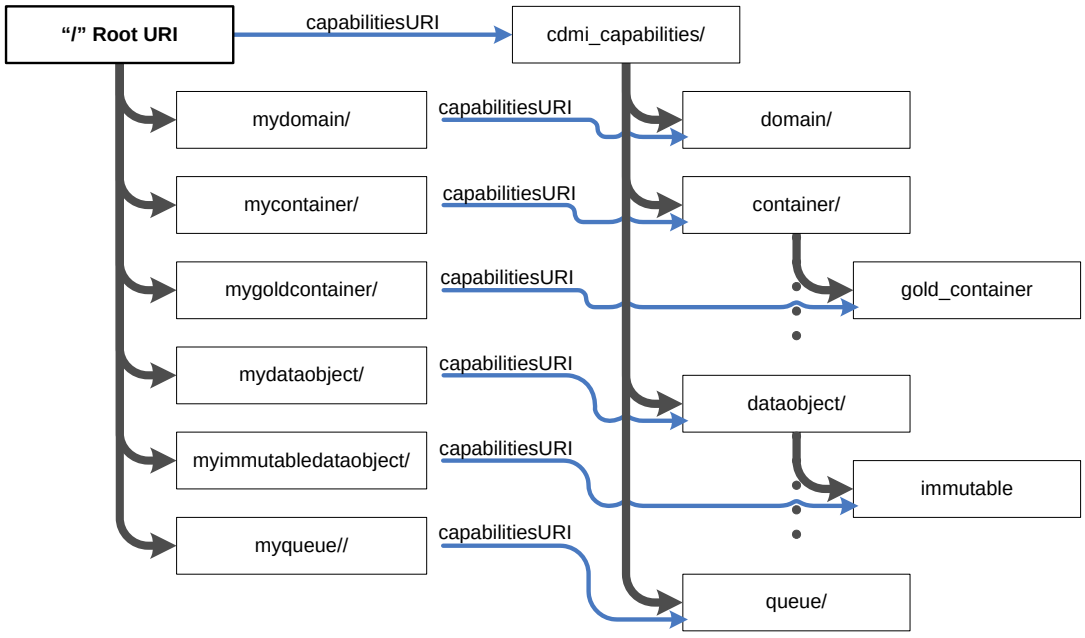


Figure 8 - Hierarchy of Capabilities

The capabilities container within the capabilities tree to which an object is linked is based on the type of the object and the data system *metadata* fields present in the object. For example, a container with no data system *metadata* fields specified may map to the "container" capabilities entry.

As an option, a CDMI implementation may map a container to a "gold_container" capabilities entry, if a data system *metadata* field is present and set to a given value, such as if the *cdmi_data_redundancy* field was set to the value of "4". This permits a cloud provider to create profiles of data system *metadata* fields and values.

Capabilities do not have CDMI metadata.

12.1.1 Cloud Storage System-Wide Capabilities

Table 10, "System-Wide Capabilities" defines the system-wide capabilities in a cloud storage system. These capabilities, which are found in the capabilities object, are referred to by the root URI (root capabilities).

Table 10 - System-Wide Capabilities

Capability Name	Type	Definition
cdmi_domains	JSON String	If present and "true", indicates that the cloud storage system supports domains. If not present, the *domainURI* field shall not be present in response bodies and the "cdmi_domains" URI shall not be present.
cdmi_export_cifs	JSON String	If present and "true", indicates that the cloud storage system supports CIFS exports.
cdmi_export_iscsi	JSON String	If present and "true", indicates that the cloud storage system supports FC exports.
cdmi_export_nfs	JSON String	If present and "true", indicates that the cloud storage system supports NFS protocol exports.
cdmi_export_occi_iscsi	JSON String	If present and "true", indicates that the cloud storage system supports OCCI/iSCSI exports.
cdmi_export_webdav	JSON String	If present and "true", indicates that the cloud storage system supports WebDAV exports.
cdmi_metadata_maxitems	JSON String	If present, this capability specifies the maximum number of user-defined metadata items supported by the cloud storage system. If absent, there is no limit placed on the number of user-defined metadata items.
cdmi_metadata_maxsize	JSON String	If present, this capability specifies the maximum size in bytes of each user-defined metadata item supported by the cloud storage system. If absent, there is no limit placed on the size of user-defined metadata items.
cdmi_notification	JSON String	If present and "true", indicates that the cloud storage system supports notification queues.
cdmi_logging	JSON String	If present and "true", indicates that the cloud storage system supports logging queues.
cdmi_query	JSON String	If present and "true", indicates that the cloud storage system supports query queues.
cdmi_query_regex	JSON String	If present and "true", indicates that the cloud storage system supports query with regular expressions.

Table 10 - System-Wide Capabilities

Capability Name	Type	Definition
cdmi_query_contains	JSON String	If present and "true", indicates that the cloud storage system supports query with the “contains” expressions.
cdmi_query_tags	JSON String	If present and "true", indicates that the cloud storage system supports query with tag-matching expressions.
cdmi_query_value	JSON String	If present and "true", indicates that the cloud storage system supports query of value fields.
cdmi_queues	JSON String	If present and "true", indicates that the cloud storage system supports queue objects.
cdmi_security_access_control	JSON String	If present and "true", indicates that the cloud storage system supports ACLs. See Section 12.1.3, "Data System Metadata Capabilities" for additional information.
cdmi_security_audit	JSON String	If present and "true", indicates that the cloud storage system supports audit logging. See Section 17.3, "Security Logging" for additional information.
cdmi_security_data_integrity	JSON String	If present and "true", indicates that the cloud storage system supports data integrity/authenticity. See Section 12.1.3, "Data System Metadata Capabilities" for additional information.
cdmi_security_encryption	JSON String	If present and "true", indicates that the cloud storage system supports data at-rest encryption. See Section 12.1.3, "Data System Metadata Capabilities" for additional information.
cdmi_security_immutability	JSON String	If present and "true", indicates that the cloud storage system supports data immutability/retentions. See Section 12.1.3, "Data System Metadata Capabilities" for additional information.
cdmi_security_sanitization	JSON String	If present and "true", indicates that the cloud storage system supports data/media sanitization. See Section 12.1.3, "Data System Metadata Capabilities" for additional information.
cdmi_serialization_json	JSON String	If present and "true", indicates that the cloud storage system supports json as a serialization format.
cdmi_snapshots	JSON String	If present and “true”, indicates that the cloud storage system supports snapshots.
cdmi_xmlrepresentation	JSON String	This capability is reserved for future use, as the xml representation is not defined in this version of CDMI. This capability shall never be present for CDMI 1.0.x.
cdmi_references	JSON String	If present and “true”, indicates that the cloud storage system supports references.

12.1.2 Storage System Metadata Capabilities

Table 11, "Capabilities for Storage System Metadata" defines the capabilities for storage system metadata in a cloud storage system. These capabilities are found in the capabilities objects for domains, data

objects, containers, and queues. See Section 16.3, "Support for Storage System Metadata" for a description of these storage system metadata elements.

Table 11 - Capabilities for Storage System Metadata

Capability Name	Type	Definition
cdmi_acl	JSON String	If present and "true", indicates that the cloud storage system supports ACLs. When a CDMI implementation supports ACLs for the purpose of access control, the system-wide capability of “cdmi_security_access_control” specified in Table 10 of Section 12.1.1, "Cloud Storage System-Wide Capabilities" shall be set to “true”. Otherwise, it shall not be present, indicating that there is no support for access control.
cdmi_size	JSON String	If present and "true", indicates that the cloud storage system shall generate a "size" storage system metadata for each stored object.
cdmi_ctime	JSON String	If present and "true", indicates that the cloud storage system shall generate a "ctime" storage system metadata for each stored object.
cdmi_atime	JSON String	If present and "true", indicates that the cloud storage system shall generate an "atime" storage system metadata for each stored object.
cdmi_mtime	JSON String	If present and "true", indicates that the cloud storage system shall generate an "mtime" storage system metadata for each stored object.
cdmi_acount	JSON String	If present and "true", indicates that the cloud storage system shall generate an "acount" storage system metadata for each stored object.
cdmi_mcount	JSON String	If present and "true", indicates that the cloud storage system shall generate an "mcount" storage system metadata for each stored object.
cdmi_hash	JSON String	If present, indicates that the cloud storage system shall generate a "cdmi_hash" storage system metadata for each stored object using the algorithm specified in the value of the cdmi_value_hash data system metadata.
cdmi_acl	JSON String	If present and "true", indicates that the cloud storage system shall support ACLs.

12.1.3 Data System Metadata Capabilities

Table 12, "Capabilities for Data System Metadata" defines the capabilities for data system metadata in a cloud storage system. These capabilities are found in the capabilities objects for domains, data objects,

containers, and queues. See Section 16.3, "Support for Storage System Metadata" for a description of these data system metadata elements.

Table 12 - Capabilities for Data System Metadata

Capability Name	Type	Definition
cdmi_data_redundancy	JSON String	If present, this capability specifies the maximum number of redundancy copies that may be specified. If absent, redundancy copies specified shall be ignored.
cdmi_infrastructure_redundancy	JSON String	If present, this capability specifies the maximum number of infrastructure redundancy copies that may be specified. If absent, infrastructure redundancy copies specified shall be ignored.
cdmi_data_dispersion	JSON String	If present and “true”, indicates that the cloud storage system shall disperse data. If absent, redundancy copies specified shall be ignored.
cdmi_data_retention	JSON String	If present and "true", indicates that the cloud storage system shall support retention.
cdmi_data_autodelete	JSON String	If present and “true”, indicates that the cloud storage system shall support the autodeletion of objects when retention ends.
cdmi_data_holds	JSON String	If present and "true", indicates that the cloud storage system shall support placing holds on objects. When a CDMI implementation supports holds for the purpose of making data immutable, the system-wide capability of “cdmi_security_immutability” specified in Table 10 of Section 12.1.1, "Cloud Storage System-Wide Capabilities" shall be set to “true”. Otherwise, it shall not be present, indicating that there is no support for data immutability.
cdmi_encryption	JSON String	If present, this capability lists the encryption algorithms and key lengths supported. If absent, objects shall not be encrypted. When a CDMI implementation supports at-rest encryption, the system-wide capability of “cdmi_security_encryption” specified in Table 10 of Section 12.1.1, "Cloud Storage System-Wide Capabilities" shall be set to “true”. Otherwise, it shall not be present, indicating that there is no support for at-rest encryption.
cdmi_value_hash	JSON String	This metadata is used to indicate if the object data is to be hashed and indicates the desired hash algorithm and length. Supported algorithm/length values are provided by the cdmi_value_hash capability.
cdmi_latency	JSON String	If present and "true", indicates that the cloud storage system shall tier data based on desired latency. If absent, the max latency specified shall be ignored.
cdmi_throughput	JSON String	If present and "true", indicates that the cloud storage system shall tier data based on desired throughput. If absent, the max throughput specified shall be ignored.
cdmi_sanitization_method	JSON String	If present, this capability lists the sanitization methods supported. When a CDMI implementation supports sanitization, the system-wide capability of “cdmi_security_sanitization” specified in Table 10 of Section 12.1.1, "Cloud Storage System-Wide Capabilities" shall be set to “true”. Otherwise, it shall not be present, indicating that there is no sanitization support.

Table 12 - Capabilities for Data System Metadata

Capability Name	Type	Definition
cdmi_RPO	JSON String	If present and "true", indicates that the cloud storage system shall manage data to achieve a specified RPO. If absent, the RPO specified shall be ignored.
cdmi_RTO	JSON String	If present and "true", indicates that the cloud storage system shall manage data to achieve a specified RTO. If absent, the RTO specified shall be ignored.

12.1.4 Data Object Capabilities

Table 13, "Capabilities for Data Objects" defines the capabilities for data objects in a cloud storage system.

Table 13 - Capabilities for Data Objects

Capability Name	Type	Definition
cdmi_read_value	JSON String	If present and "true", indicates that the object's value may be read.
cdmi_read_value_range	JSON String	If present and "true", indicates that the object's value may be read with byte ranges.
cdmi_read_metadata	JSON String	If present and "true", indicates that the object's metadata may be read.
cdmi_modify_value	JSON String	If present and "true", indicates that the object's value may be modified.
cdmi_modify_value_range	JSON String	If present and "true", indicates that the object's value may be modified with byte ranges.
cdmi_modify_metadata	JSON String	If present and "true", indicates that the object's metadata may be modified.
cdmi_serialize_dataobject	JSON String	If present and "true", indicates that the object may be serialized.
cdmi_delete_dataobject	JSON String	If present and "true", indicates that the object may be deleted.

12.1.5 Container Capabilities

Table 14, "Capabilities for Containers" defines the capabilities for containers in a cloud storage system.

Table 14 - Capabilities for Containers

Capability Name	Type	Definition
cdmi_list_children	JSON String	If present and "true", indicates that the container's children may be listed.
cdmi_list_children_range	JSON String	If present and "true", indicates that the container's children may be listed with ranges.
cdmi_read_metadata	JSON String	If present and "true", indicates that the container's metadata may be read.

Table 14 - Capabilities for Containers

Capability Name	Type	Definition
cdmi_modify_metadata	JSON String	If present and "true", indicates that the container's metadata may be modified.
cdmi_snapshot	JSON String	If present and "true", indicates that the container allows a new snapshot to be created.
cdmi_serialize_container	JSON String	If present and "true", indicates that the container and all children's contents may be serialized.
cdmi_deserialize_container	JSON String	If present and "true", indicates that the container permits the deserialization of serialized containers and associated serialized children into the container
cdmi_deserialize_queue	JSON String	If present and "true", indicates that the container permits the deserialization of serialized queues into the container.
cdmi_deserialize_dataobject	JSON String	If present and "true", indicates that the container permits the deserialization of serialized data objects into the container.
cdmi_create_dataobject	JSON String	If present and "true", indicates that the container allows a new object to be added.
cdmi_post_dataobject	JSON String	If present and "true", indicates that the container allows a new object to be added via POST.
cdmi_post_queue	JSON String	If present and "true", indicates that the container allows a new queue to be added via POST.
cdmi_create_container	JSON String	If present and "true", indicates that the container allows a new container may be added.
cdmi_create_queue	JSON String	If present and "true", indicates that the container allows queues to be created.
cdmi_create_reference	JSON String	If present and "true", indicates that the container allows a new child reference may be added.
cdmi_delete_container	JSON String	If present and "true", indicates that the container may be deleted.
cdmi_move_container	JSON String	If present and "true", indicates that the container may be moved (via PUT) to another URI.
cdmi_copy_container	JSON String	If present and "true", indicates that the container may be copied (via PUT) to another URI.

12.1.6 Domain Capabilities

Table 15, "Capabilities for Domains" defines the capabilities for domains in a cloud storage system. (All capabilities refer to what may be done via CDMI operations.)

Table 15 - Capabilities for Domains

Capability Name	Type	Definition
cdmi_create_domain	JSON String	If present and "true", indicates that the domain allows a new subdomain to be added.
cdmi_delete_domain	JSON String	If present and "true", indicates that the domain may be deleted.

Table 15 - Capabilities for Domains

Capability Name	Type	Definition
cdmi_domain_summary	JSON String	If present and "true", indicates that the domain supports domain summaries.
cdmi_domain_members	JSON String	If present and "true", indicates that the domain supports domain user management.
cdmi_list_children	JSON String	If present and "true", indicates that the domain's children may be listed.
cdmi_read_metadata	JSON String	If present and "true", indicates that the domain's metadata may be read.
cdmi_modify_metadata	JSON String	If present and "true", indicates that the domain's metadata may be modified.
cdmi_create_container	JSON String	If present and "true", indicates that the domain allows a new container may be added.
cdmi_create_queue	JSON String	If present and "true", indicates that the domain allows queues to be created.
cdmi_copy_domain	JSON String	If present and "true", indicates that the container may be copied (via PUT) to another URI.
cdmi_serialize_domain	JSON String	If present and "true", indicates that the domain and all child domains may be serialized.
cdmi_deserialize_domain	JSON String	If present and "true", indicates that the domain permits the deserialization of serialized domains and associated serialized children into the domain.

12.1.7 Queue Object Capabilities

Table 16, "Capabilities of Queue Objects" defines the capabilities for queue objects in a cloud storage system.

Table 16 - Capabilities of Queue Objects

Capability Name	Type	Definition
cdmi_read_value	JSON String	If present and "true", indicates that the queue 's value may be read.
cdmi_read_metadata	JSON String	If present and "true", indicates that the queue's metadata may be read.
cdmi_serialize_queue	JSON String	If present and "true", indicates that the queue may be serialized.
cdmi_modify_value	JSON String	If present and "true", indicates that the queue 's value and metadata may be modified.
cdmi_modify_metadata	JSON String	If present and "true", indicates that the queue's metadata may be modified.
cdmi_delete_queue	JSON String	If present and "true", indicates that the queue may be deleted.
cdmi_move_queue	JSON String	If present and "true", indicates that the queue may be moved to another URI.

Table 16 - Capabilities of Queue Objects

Capability Name	Type	Definition
cdmi_copy_queue	JSON String	If present and "true", indicates that the queue may be copied to another URI.
cdmi_reference_queue	JSON String	If present and "true", indicates that the queue may be referenced from another queue.

12.2 Read a Capabilities Object (CDMI Content Type)

Synopsis:

Reads from an existing capability object at the specified URI.

```
GET <root URI>/cdmi_capabilities/<Capability>/<TheCapability>/
GET <root URI>/cdmi_capabilities/<Capability>/<TheCapability>/
   ?<fieldname>;<fieldname>
GET <root URI>/cdmi_capabilities/<Capability>/<TheCapability>/?children:{range}
```

- `<root URI>` is the path to the CDMI cloud.
- `<Capability>` is zero or more intermediate capabilities containers.
- `<TheCapability>` is the name specified for the capabilities to be read from.
- `<fieldname>` is the name of a field.
- `<range>` is a numeric range within the list of children.

The object may also be accessed at `<root URI>/cdmi_objectid/<objectID>`.

Capabilities:

The following capabilities describe the supported operations that may be performed when reading an existing capabilities object:

- All CDMI implementations shall permit clients to read the metadata and contents of all capabilities objects.

Request Headers:

Header	Type	Description	Requirement
Accept	Header String	"application/cdmi-capability". Shall contain a list of one or more of the five CDMI MIME types.	Mandatory
X-CDMI-Specification-Version	String Array	A comma separated list of versions supported by the client, for example, "1.0, 1.5, 2.0".	Mandatory

Request Message Body:

None specified.

Response Headers:

Header	Type	Description	Requirement
X-CDMI-Specification-Version	Header String	The server shall respond with the highest version supported by both the client and the server, for example, "1.0".	Mandatory
Content-Type	Header String	"application/cdmi-capability".	Mandatory

Response Message Body:

Field Name	Type	Description	Requirement
objectURI	JSON String	URI of the object as specified in the request.	Mandatory
objectID	JSON String	Object ID of the object.	Mandatory
objectName	JSON String	Name of the object. • If the object has a path, the name shall be the last part of the path. • If the object does not have a path and is only accessible by ID, then the name shall be the Object ID of the object. • If an implementor chooses to always return the Object ID as the name of an object, even if one or more paths exists, the *parentURI* shall be set to `/cdmi_objectID/`.	Mandatory
parentURI	JSON String	URI for the parent object. • If the object has a path, the *parentURI* shall be the URI path to the parent object. OR • If the object does not have a path and is only accessible by ID, the *parentURI* shall be set to `/cdmi_objectid/`, and *objectName* shall be set to the Object ID of the object."	Mandatory
capabilites	JSON Object	A tag list of capabilities supported by the corresponding object. Capabilities in the "cdmi_capabilities" object are system-wide capabilities. Capabilities found in children objects correspond to the capabilities of a specific subset of objects.	Mandatory
childrenrange	JSON String	The range of the children returned in the *children* field.	Mandatory
children	JSON Array	Names of the children capabilities objects. For the root capabilities container, this includes "domain/", "container/", "dataobject/", and "queue/". Within each of these capabilities objects, further more specialized capabilities profiles may be specified by the cloud storage system.	Mandatory

If individual fields are specified in the GET request, only these fields are returned in the result body. Optional fields that are requested but do not exist are omitted from the result body.

Response Status:

HTTP Status	Description
200 OK	Capabilities object list in response.
400 Bad Request	Invalid parameter or field names in the request.
401 Unauthenticated	Incorrect or missing authentication credentials.
403 Unauthorized	Client lacks the proper authorization to perform this request.
404 Not Found	A container was not found at the specified URI.
406 Not Acceptable	The server is unable to provide the object in the content-type specified in the Accept header.

Example 12-1

GET to the root capabilities container URI to read all fields of the container

```
GET /cdmi_capabilities/ HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-capability
Location: application/cdmi-capability
X-CDMI-Specification-Version: 1.0
```

The response looks like:

```
HTTP/1.1 200 OK
Location: application/cdmi-capability
X-CDMI-Specification-Version: 1.0

{
    "objectURI" : "/cdmi_capabilities/",
    "objectID" : "0000706D00101ADEBC119D1BFE98672A",
    "objectName" : "cdmi_capabilities/",
    "parentURI" : "/",
    "capabilities" : {
        "cdmi_domains" : "true",
        "cdmi_export_nfs" : "true",
        "cdmi_export_webdav" : "true",
        "cdmi_export_iscsi" : "true",
        "cdmi_queues" : "true",
        "cdmi_notification" : "true",
        "cdmi_query" : "true",
        "cdmi_metadata_maxsize" : "4096",
        "cdmi_metadata_maxitems" : "1024"
    },
    "childrenrange" : "0-3",
    "children" : [
        "domain/",
        "container/",
        "dataobject/",
        "queue/"
    ]
}
```

Example 12-2

GET to the root capabilities container URI to read the capabilities and children of the container

```
GET /cdmi_capabilities/?capabilities;children HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-capability
Location: application/cdmi-capability
X-CDMI-Specification-Version: 1.0
```

The response looks like:

```
HTTP/1.1 200 OK
Location: application/cdmi-capability
X-CDMI-Specification-Version: 1.0

{
    "capabilities" : {
        "cdmi_domains" : "true",
        "cdmi_export_nfs" : "true",
        "cdmi_export_webdav" : "true",
        "cdmi_export_iscsi" : "true",
        "cdmi_queues" : "true",
        "cdmi_notification" : "true",
        "cdmi_query" : "true",
        "cdmi_metadata_maxsize" : "4096",
        "cdmi_metadata_maxitems" : "1024"
    },
    "children" : [
        "domain/",
        "container/",
        "dataobject/",
        "queue/"
    ]
}
```

Example 12-3

GET to the root capabilities container URI to read the first two children of the container

```
GET /cdmi_capabilities/?childrenrange;children:0-1 HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-capability
Location: application/cdmi-capability
X-CDMI-Specification-Version: 1.0
```

The response looks like:

```
HTTP/1.1 200 OK
Location: application/cdmi-capability
X-CDMI-Specification-Version: 1.0

{
    "childrenrange" : "0-1",
    "children" : [
        "domain/",
        "container/"
    ]
}
```

13 Exported Protocols

CDMI containers are accessible not only via CDMI as a data path, but also via other protocols as well. This access is especially useful for using CDMI as the storage interface for a cloud-computing environment, as Figure 9 shows.

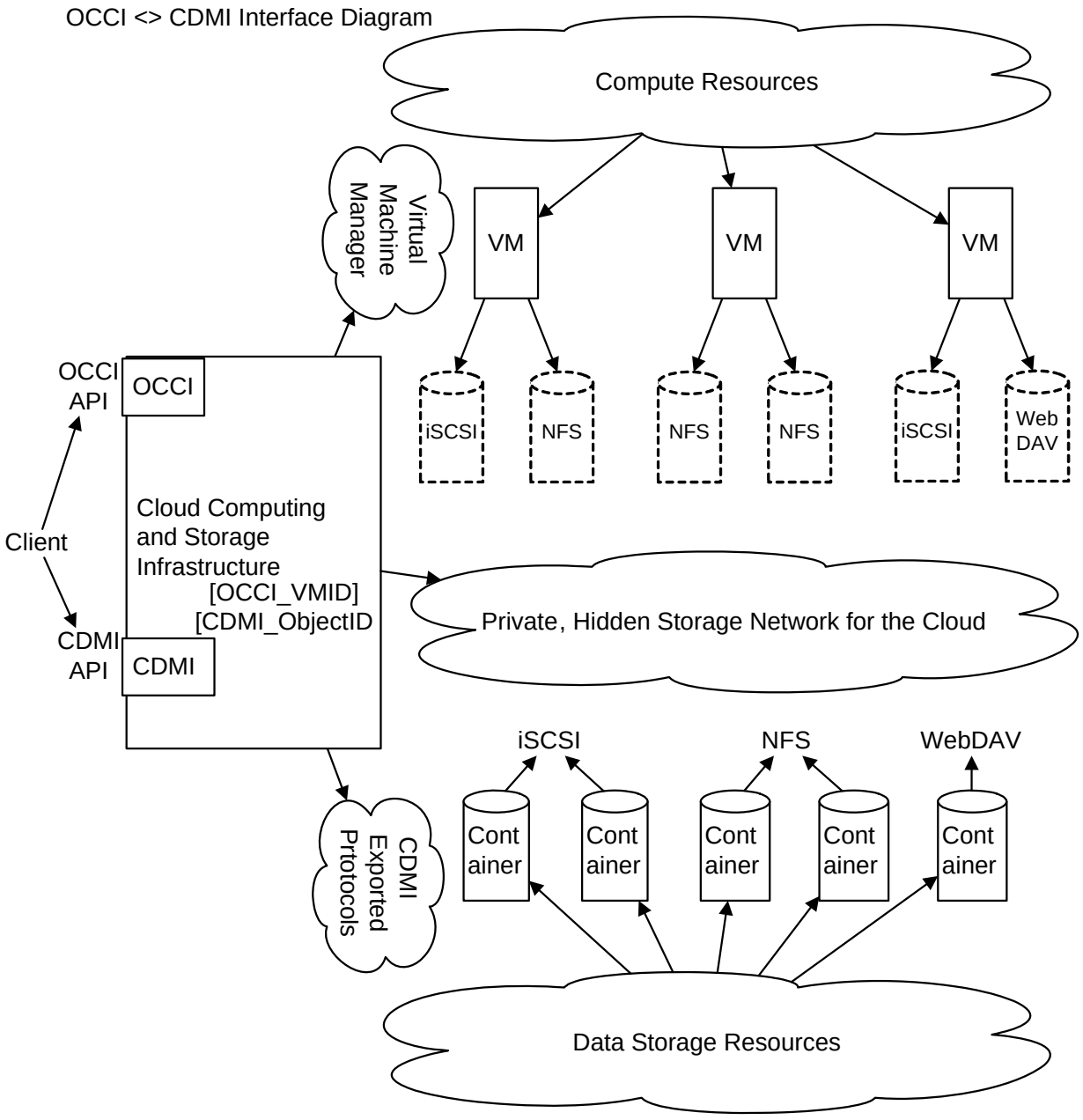


Figure 9 - CDMI and OCCI in an Integrated Cloud Computing Environment

The exported protocols from CDMI containers may be used by the virtual machines in the cloud-computing environment as virtual disks on each guest as shown. The cloud computing infrastructure management is shown as implementing both an Open Cloud Computer Interface (OCCI) and CDMI interfaces. With the internal knowledge of the network and the virtual machine manager's mapping of drives, this infrastructure may hook the CDMI containers to the guests using the appropriate exported protocol.

To do this interoperably, CDMI provides a type of exported protocol that contains information obtained via the OCCI interface. In addition, OCCI provides a type of storage that corresponds to a CDMI container that is exported with a specific type of protocol used by OCCI. A client of both interfaces would perform the following operations as an example:

1. The client creates a CDMI container through the CDMI interface and exports it as an OCCI export protocol type. The CDMI container objectID is returned as a result.
2. The client then creates a virtual machine through the OCCI interface and attaches a storage volume of type CDMI using the objectID and protocol type. The OCCI virtual machine ID is returned as a result.
3. The client then updates the export protocol structure of the CDMI container object with the OCCI virtual machine ID to allow the virtual machine access to the container.
4. The client then starts the virtual machine through the OCCI interface.

13.1 Exported Protocol Structure

The export of a container, via data path protocols other than CDMI, is done by creating or updating a container and supplying one or more export protocol structures, one for each such protocol.

The elements of the export protocol structure include:

- The protocol being used
- The identity of the container as standardized by the protocol
- The list of who may access that container via that protocol, identified as standardized by that protocol (may leverage the CDMI domains for this)

CDMI standardizes several export protocol structures for various protocols. Export protocol structures may also be defined for proprietary and vendor extensions of protocols. For more information, see Annex B, "(informative) Extending the Interface".

13.2 OCCI Exported Protocol

CDMI defines an export protocol structure for the OGF standard: Open Cloud Computing Interface (OCCI) as follows:

- Protocol is "OCCI/<protocol standard>" (i.e., OCCI/NFSv4)
- The identifier is the CDMI objectid
- The list of who may access is a list of OCCI VM IDs

An example of an OCCI export protocol structure in JSON is as follows:

```
"OCCI/iSCSI" : {"identifier" : "0000706D00101ADEBC119D1BFE98672A", "permissions" :
   "0000706D00107B85BFE6D20B84D603CA" }
```

13.3 iSCSI Export Modifications

CDMI defines the export of a container using the iSCSI protocol. Each container is exported as a single SCSI Logical Unit at a Logical Unit Number (LUN) to one or more iSCSI initiators through an iSCSI target node and port using one or more iSCSI network portals (IP addresses). Support for this export by a CDMI

implementation is advertised by the cdmi_export_iscsi capability within the system-wide capabilities (see Section 12.1.1, "Cloud Storage System-Wide Capabilities").

The export is described by the presence of an *export* field structure on the container that specifies the following information:

- Export protocol (“Network/iSCSI”)
- iSCSI target information (IP addresses or fully qualified domain names, target identifier, and LUN)
- Logical unit world-wide name
- iSCSI initiators having access

The target identifier may be in iqn, naa, or eui format and shall have the target portal group tag appended in hexadecimal.

13.3.1 Read Container

All of the information in the export structure is returned:

```
"exports" : {
    "Network/iSCSI" : {
        "portals" : [
            "192.168.1.101",
            "192.168.1.102",
         ],
        "target_identifier" :
            "iqn.2010-01.com.cloudprovider:acmeroot.container1,t,0x0001",
        "logical_unit_number" : "3",
        "logical_unit_name" : "0x60012340000000000000000000000001",
        "permissions" : [
            "iqn.2010-01.com.acme:host1" ,
            "iqn.2010-01.com.acme:host2,"
         ]
    }
},
```

13.3.2 Create Container

The follow code creates a container with iSCSI export or updates an existing container with new iSCSI export. Support for either of these operations is indicated by the cdmi_export_iscsi capability on the parent container of the created container or of the existing container, respectively.

For these export creation operations, the CDMI implementation selects the IP portals, iSCSI target, logical unit number and logical unit name; these are not supplied. Only the list of initiator identifiers that are to have access to the container are specified:

```
"exports" : {
    "Network/iSCSI" : {
        "permissions" : [
            "iqn.2010-01.com.acme:host1" ,
            "iqn.2010-01.com.acme:host2"
         ]
    }
},
```

13.3.3 Modify an Export

The following code modifies an export on an existing container. Support for this operation is indicated by the cdmi_export_iscsi on the parent container of the existing container. For this operation, only the current list of initiator identifiers that are to have access to the container are specified:

```
 "exports" : {
      "Network/iSCSI" : {
          "permissions" : [
              "iqn.2010-01.com.acme:host1" ,
              "iqn.2010-01.com.acme:host2"
           ]
      }
   },
```

13.4 NFS Exported Protocol

CDMI defines an export protocol structure for the NFS standard as follows:

- Protocol is "Network/NFSv4"
- The path of the share as presented to clients (including server host name)
- The list of who may access the share

This example shows an NFSv4 export protocol structure in JSON:

```
"Network/NFSv4" : { "identifier" : "/users", "permissions" : "domain" }
```

In this example, the value "domain" in the *permissions* field indicates that user credentials should be mapped through the domain membership in the domain of the CDMI container being exported.

For more information on the NFS version 4 protocol, see [RFC3530].

Note: This section is incomplete and needs further definition.

13.5 WebDAV Exported Protocol

CDMI defines an export protocol structure for the WebDAV [RFC4918] standard as follows:

- Protocol is "Network/WebDAV".
- The path of the WebDAV mount point as presented to clients (including server host name).
- The list of who may access the share is determined by the standard CDMI ACLs for each resource as exported via WebDAV.

The following example shows a WebDAV export protocol structure in JSON:

```
"Network/WebDAV" : { "identifier" : "/users", "permissions" : "domain" }
```

In this example, the value "domain" in the *permissions* field indicates that user credentials should be mapped through the domain membership in the domain of the CDMI container being exported.

WebDAV supports locking, but it is up to implementations to support any locking of access through the CDMI as a result, and the interaction between the two protocols is purposely not described in this standard.

14 Snapshots

A snapshot is a point-in-time image of a container and its contents. The client names a snapshot of a container at the time the snapshot is taken. The operation results in a child destination container of the **cdmi_snapshots** container under the source container. The snapshot does not include the cdmi_snapshots child container or its contents (see Figure 10, "Snapshot Operation").

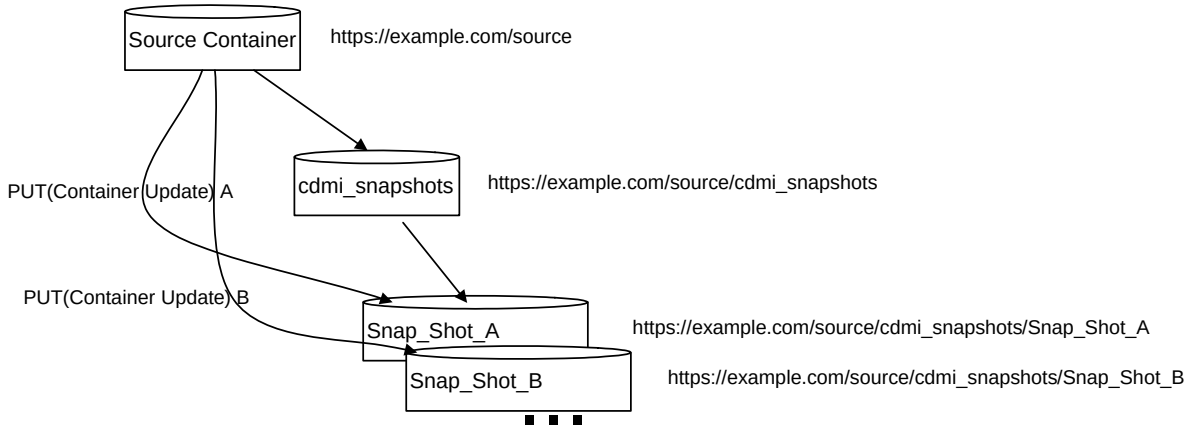


Figure 10 - Snapshot Operation

To take a snapshot, a Container Update operation is performed, as described in Section 9.6, "Update a Container (CDMI Content Type)", supplying the snapshot parameter (the name of the snapshot).

Table 17 describes the snapshot parameter of a container update operation.

Table 17 - Snapshot Parameter of the Container Update Operation

Name	Type	Description	Requirement
snapshot	JSON String	Name of the snapshot to be taken. If a snapshot is added or changed, the PUT operation only returns after the snapshot is added to the snapshot list. Once created, snapshots may be accessed as children containers under the *cdmi_snapshots* child container of the container receiving a snapshot.	Mandatory

Snapshots may be accessed in place or used as the source for copy operations, restoring a container to a previous point in time.

15 Serialization/Deserialization

Cloud storage provides benefits to users and applications due to the pervasive presence and "always on" availability. Occasionally, data is moved between, into, or out of clouds in bulk operations. Cloud serialization operations are enabled by normalizing data to a canonical, self-describing format. Use case examples of cloud serialization include, but are not limited to:

- Data migration between clouds
- Data migration during upgrades (or replacements) of cloud implementations
- Robust backup

The canonical format of serialized data describes how the data is to be represented in a byte stream. This stream may be transported in any desirable conveyance, and as long as no alteration occurs, the data may be reconstituted on the destination system.

15.1 Exporting Serialized Data

A canonical encoding of the data is obtained by creating a new data object and specifying that the source for the creation is to serialize a given CDMI data object, container, or queue. On a successful serialization, the result shall be a data object that is created with the serialized data as its value. If a container has an exported block protocol, the serialized data may contain the block-by-block contents of that container along with its metadata as if it were a data object.

The resulting data object that is produced is the canonical representation of the selected data object, container, and children or queue.

- If the source specified is a data object, the canonical format shall contain the data object contents and its metadata.
- If the source being specified is a queue, the queue contents and its metadata shall be included.
- If the source being specified is a container, all children containers, data objects, and queues of the specified container shall be included.

Only objects that the user who is performing the serialization operation has permissions to read shall be included in the resulting serialized object.

15.2 Importing Serialized Data

Canonical data may be deserialized back into the cloud by creating a new data object, container, or queue and by specifying that the source for the creation is to deserialize a given CDMI data object.

The destination may or may not exist previously. If not, a "create" operation is performed. If a container already exists, an "update" operation with serialized data is not permitted. Data objects are recreated as specified in the canonical format, including all metadata and the data object identifier (OID). If the serialized data object contains block-by-block data from a container that was exporting a block protocol, the container shall be created with that data as its new value, but the exported block protocol shall be set up with a separate update.

If the user who is deserializing a serialized data object has the "cross-domain" privilege and has not specified a domainURI as part of the deserialize operation, the original domainURIs from the serialized object shall be used. If any of the specified domainURIs are not valid in the context of the storage system on which the deserialization operation is being performed, the entire deserialize operation shall fail.

If the user who is deserializing a serialized object specifies a domainURI as part of the deserialize operation, the domainURI of every object being deserialized shall be set to the specified domainURI. If the user does not have the "cross_domain" privilege, only the domainURI of the parent object may be specified.

If the user who is deserializing a serialized object does not specify a domainURI and does not have the "cross_domain" privilege, then the deserialization operation shall only be successful if all objects have the same domainURI as the parent object on which the deserialization operation is being performed.

Deserialization operations shall restore all metadata from the specified source. If the original provider of the serialized data-supported vendor extensions is through custom metadata keys and values, then these customized requirements shall be restored when deserialized. However, the custom metadata keys and values may be treated as user metadata (preserved, but not interpreted) by the destination provider. Preservation allows custom data requirements to move between clouds without losing this information. For more information on vendor extensions, see Annex B, "(informative) Extending the Interface".

15.2.1 Canonical Format

The canonical format shall represent specified data objects and containers, as they exist within the storage system. Each object shall be represented by the metadata for the object, identifiers, and the data stream contents of the data object. Because metadata is inherited from enclosing containers, all parent metadata shall be represented in the canonical format (essentially flattening the hierarchy). To preserve the actual metadata values that apply to the data object that is being serialized, the non-overridden metadata is included from both the immediate parent container of the specified object and from the parent of each higher-level container.

The canonical format shall have the following characteristics:

- Recursive JSON for the data object, consistent with the rest of CDMI
- User and data system metadata for each data object/container
- Data stream contents for each data object and queue
- Binary data is represented using escaped JSON strings
- Typing of data elements is consistent with CDMI JSON representations

15.2.2 Example JSON Canonical Serialized Format

In this example, a data object and a queue in a container have been selected for serialization.

```
{
    "objectURI" : "/MyContainer/",
    "objectID" : "0000706D00101ADEBC119D1BFE98672A",
    "objectName" : "MyContainer/",
    "parentURI" : "/",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/Container/",
    "completionStatus" : "Complete",
    "metadata" : {

    },
"exports" : {
        "OCCI/iSCSI" : {
            "identifier" : "0000706D00101ADEBC119D1BFE98672A",
            "permissions" : "0000706D00107B85BFE6D20B84D603CA"
        },
        "Network/NFSv4" : {
            "identifier" : "/users",
            "permissions" : "domain"
        }
    },
 "childrenrange" : "0-1",
    "children" : [
        {
            "objectURI" : "/MyContainer/MyDataObject.txt",
            "objectID" : "0000706D00101ADEBC119D1BFE98672A",
            "objectName" : "MyDataObject.txt",
            "parentURI" : "/MyContainer/",
            "domainURI" : "/cdmi_domains/MyDomain/",
            "capabilitiesURI" : "/cdmi_capabilities/DataObject/",
            "completionStatus" : "Complete",
            "mimetype" : "text/plain",
            "metadata" : {

            },
            "valuerange" : "0-37",
            "value" : "This is the Value of this Data Object"
        },
{
            "objectURI" : "/MyContainer/MyQueue",
            "objectID" : "0000706D00101ADEBC119D1BFE98672A",
            "objectName" : "MyQueue",
            "parentURI" : "/MyContainer/",
            "domainURI" : "/cdmi_domains/MyDomain/",
            "capabilitiesURI" : "/cdmi_capabilities/Queue/",
            "completionStatus" : "Complete",
            "metadata" : {

            },
            "queueValues" : "0-1",
            "valuerange" : [
                "0-2",
                "0-4"
            ],
            "value" : [
                "red",
                "blue"
            ]
        }
    ]
}
```

To allow efficient deserialization in stream mode when serializing containers to JSON, the children array should be the last item in the container JSON object.

16 Metadata

16.1 Access Control

Access control comprises the mechanisms by which various types of access to objects and containers are authorized and permitted or denied. CDMI uses the well-known mechanism of an Access Control List (ACLs). ACLs are lists of permissions-granting or permissions-denying entries called access control entries (ACEs).

16.1.1 ACL and ACE Structure

An ACL is an ordered list of ACEs. The two types of ACEs in CDMI are ALLOW and DENY. An ALLOW ACE grants some form of access to a principal. Principals are either users or groups, and are represented by identifiers. A DENY ACE denies access of some kind to a principal. For instance, a DENY ACE may deny the ability to write the metadata or ACL of an object, but may remain silent on other forms of access. In that case, if another ACE ALLOWs write access to the object, the principal is allowed to write the object's data, but nothing else.

ACEs are composed of five fields. In C, their declaration would look like this:

```
/*
 * ACE structure
 */
typedef unsigned int   uint_t;
typedef char           utf8_t;
typedef char*          utf8str_t;

struct cdmi_ace_t {
       uint_t       type;
       utf8str_t    who;
       uint_t       flags;
       uint_t       access_mask;
};
```

While the type, flags, and access_mask are specified as unsigned integers, for efficiency, each defined bit in them has a corresponding string (in C, the label-pasted form of the macro), which may be used where desired for readability.

16.1.2 ACE Type

The following ACE types are defined, following NFSv4:

```
/*
 * ACE types
 *
 *      macro/constant form    bitwise form    string form
 */
const CDMI_ACE_ACCESS_ALLOW   = 0x00000000;    "ALLOW"
const CDMI_ACE_ACCESS_DENY    = 0x00000001;    "DENY"
const CDMI_ACE_SYSTEM_AUDIT   = 0x00000002;    "AUDIT"
```

Note: The reason that the string forms may be safely abbreviated is that they are local to the ACE structure type, as opposed to constants, which are relatively global in scope.

ACLs are customarily ordered with the DENY ACEs at the front of the list, but there are no known server implementations which enforce this. The reason for the custom is that an ACL like this

```
ALLOW        EVERYONE@    NONE       ALL_PERMS    200912250001
DENY         JIM          NONE       ALL_PERMS    200912250002
```

would not have the desired effect of allowing access to everyone except Jim, as ACLs are evaluated in order, and JIM, being a member of EVERYONE@, would be granted access before the DENY ACE was ever encountered.

However, the client is responsible for ordering the ACEs in an ACL. This ordering conforms to both Windows and NFSv4. The server shall not enforce any ordering, and shall store and evaluate the ACEs in the order given by the client.

16.1.3 ACE Who

The special "who" identifiers need to be understood universally, rather than in the context of a particular DNS domain (see Table 18, "Who Identifiers"). Some of these identifiers may not be understood when an NFS client accesses the server, but have meaning when a local process accesses the file. The ability to display and modify these permissions is permitted over NFS, even if none of the access methods on the server understands the identifiers.

Table 18 - Who Identifiers

Who	Description
"OWNER@"	The owner of the file.
"GROUP@"	The group associated with the file.
"EVERYONE@"	The world.
"ANONYMOUS@"	Accessed without any authentication.
"AUTHENTICATED@"	Any authenticated user (opposite of ANONYMOUS).
"ADMINISTRATOR@"	A user with administrative status, e.g., root.
"ADMINUSERS@"	A group whose members are given administrative status.

To avoid name conflicts, these special identifiers are distinguished by an appended "@" (with no domain name).

16.1.4 ACE Flags

CDMI allows for nested containers and mandates that objects and subcontainers be able to inherit access permissions from their parent containers. However, it is not enough to simply inherit all permissions from the parent; it might be desirable, for example, to have different default permissions on child objects and subcontainers of a given container. The following flags govern this behavior.

```
/*
 * ACE flag bits
 *
 *      macro/constant form                bitwise form    string form
 */
```

```
const CDMI_ACE_FLAGS_NONE                   = 0x00000000;   "NO_FLAGS"
const CDMI_ACE_FLAGS_OBJECT_INHERIT_ACE     = 0x00000001;   "OBJECT_INHERIT"
const CDMI_ACE_FLAGS_CONTAINER_INHERIT_ACE  = 0x00000002;   "CONTAINER_INHERIT"
const CDMI_ACE_FLAGS_NO_PROPAGATE_ACE       = 0x00000004;   "NO_PROPAGATE"
const CDMI_ACE_FLAGS_INHERIT_ONLY_ACE       = 0x00000008;   "INHERIT_ONLY"
const CDMI_ACE_FLAGS_INHERITED_ACE          = 0x00000100;
```

- An ACE on which CDMI_ACE_FLAGS_OBJECT_INHERIT_ACE is set is inherited by objects in a container, but not subcontainers. This effectively causes the inheritance to be one level deep.
- An ACE on which CDMI_ACE_FLAGS_CONTAINER_INHERIT_ACE is set is inherited by subcontainers of a container, but not the objects.
- An ACE on which CDMI_ACE_FLAGS_NO_PROPAGATE_ACE is not inherited by any objects or subcontainers. It applies only to the object or container on which it is set.
- An ACE on which CDMI_ACE_FLAGS_INHERIT_ONLY_ACE child objects or subcontainers. It is ignored when evaluating access for the object or container on which it is set. This flag is particularly useful for top-level containers, which may specify any kind of default permissions for child objects, but themselves may only be writable by their owners or administrators, for instance.

16.1.5 ACE Mask Bits

The *mask* field of an ACE contains 32 bits. The following are defined in CDMI; their values are taken from the IETF NFSv4 RFC 3530.

```
/*
 * ACE access_mask bits
 *
 *      macro/constant form          bitwise form    string form
 */
const CDMI_ACE_READ_OBJECT          = 0x00000001;    "READ_OBJECT"
const CDMI_ACE_LIST_CONTAINER       = 0x00000001;    "LIST_CONTAINER"
const CDMI_ACE_WRITE_OBJECT         = 0x00000002;    "WRITE_OBJECT"
const CDMI_ACE_ADD_OBJECT           = 0x00000002;    "ADD_OBJECT"
const CDMI_ACE_APPEND_DATA          = 0x00000004;    "APPEND_DATA"
const CDMI_ACE_ADD_SUBCONTAINER     = 0x00000004;    "ADD_SUBCONTAINER"
const CDMI_ACE_READ_METADATA        = 0x00000008;    "READ_METADATA"
const CDMI_ACE_WRITE_METADATA       = 0x00000010;    "WRITE_METADATA"
const CDMI_ACE_EXECUTE              = 0x00000020;    "EXECUTE"
const CDMI_ACE_DELETE_OBJECT        = 0x00000040;    "DELETE_OBJECT"
const CDMI_ACE_DELETE_SUBCONTAINER  = 0x00000040;    "DELETE_SUBCONTAINER"
const CDMI_ACE_READ_ATTRIBUTES      = 0x00000080;    "READ_ATTRIBUTES"
const CDMI_ACE_WRITE_ATTRIBUTES     = 0x00000100;    "WRITE_ATTRIBUTES"
const CDMI_ACE_DELETE               = 0x00010000;    "DELETE"
const CDMI_ACE_READ_ACL             = 0x00020000;    "READ_ACL"
const CDMI_ACE_WRITE_ACL            = 0x00040000;    "WRITE_ACL"
const CDMI_ACE_WRITE_OWNER          = 0x00080000;    "WRITE_OWNER"
const CDMI_ACE_SYNCHRONIZE          = 0x00100000;    "SYNCHRONIZE"
const CDMI_ACE_SET_RETENTION        = 0x10000000;    "SET_RETENTION"
```

Several summary values are useful for routine work:

```
const CDMI_ACE_READ                 = 0x00000009;    "READ"
const CDMI_ACE_READ_ALL             = 0x00020089;    "READ_ALL"
const CDMI_ACE_WRITE                = 0x00000016;    "WRITE"
const CDMI_ACE_WRITE_ALL            = 0x00040156;    "WRITE_ALL"
const CDMI_ACE_RW                   = 0x0000001F;    "RW"
const CDMI_ACE_RW_ALL               = 0x0006006F;    "RW_ALL"
const CDMI_ACE_ALL                  = 0x1007FFFF;    "ALL_PERMS"
```

Note that CDMI_ACE_WRITE_OWNER is not included in CDMI_ACE_WRITE_ALL or CDMI_ACE_ALL. This permission, to change the owner of an object, is generally only given to admins and the object's owner.

Note that several constants are duplicated in the above list. This duplication is because several of the flags apply to both objects and containers:

Object type	constant	string form
object	0x01	"READ_OBJECT"
container	0x01	"LIST_CONTAINER"
object	0x02	"WRITE_OBJECT"
container	0x02	"ADD_OBJECT"
object	0x04	"APPEND_DATA"
container	0x04	"ADD_SUBCONTAINER"
object	0x40	"DELETE_OBJECT"
container	0x40	"DELETE_SUBCONTAINER"

Implementations should use the correct string form for display of permissions, if the object type is known. If the object type is unknown, the "object" version of the string should be used.

16.1.6 ACL Evaluation

When evaluating whether access to a particular object *O* by a principal *P* shall be granted, the server traverses the object's logical ACL (its ACL after processing inheritance from ancestor containers) in list order, using a temporary permissions bitmask *m*, initially empty (all zeroes).

- If the ACL timestamp is nonexistent or is older than that of some parent (see foregoing discussion in previous section), use `get_acl(O, P)` to set the physical ACL equal to the logical ACL.
- If the object still does not contain an ACL, the algorithm terminates and access is denied for all users and groups. This condition is not expected, as CDMI implementations should require an inheritable default ACL on all root containers.
- ACEs that do not refer to the principal *P* requesting the operation are ignored.
- If an ACE is encountered that denies access to *P* for any of the requested mask bits, access is denied and the algorithm terminates.
- If an ACE is encountered that allows access to *P*, the permissions mask *m* for the operation is XORed with the permissions mask from the ACE. If *m* is sufficient for the operation, access is granted and the algorithm terminates.

- If the end of the ACL list is reached and permission has neither been granted nor explicitly denied, access is denied and the algorithm terminates, unless the object is a container root. In this case, the server shall:
  - Allow access to the container owner, ADMINISTRATOR@, and any member of ADMINUSERS@
  - Log an event indicating what has happened

When permission for the desired access is not explicitly given, even ADMINISTRATOR@ and equivalents are denied for objects that aren't container roots. When an admin needs to access an object in such an instance, the root container shall be accessed and its inheritable ACEs changed in a way as to allow access to the original object. The resulting log entry then provides an audit trail for the access.

When a root container is created and no ACL is supplied, the server shall place an ACL containing the following ACEs on the container:

```
{
    [
        {
            "ALLOW",
            "OWNER@",
            "NO_FLAGS",
            "ALL_PERMS",
        } ,
        {
            "ALLOW",
            "AUTHENICATED@",
            "NO_FLAGS",
            "READ",
        }
    ]
}
```

As ACLs are metadata, they are PUT and GOTTEN through the *metadata* field of a PUT or GET request. The syntax is as follows:

```
ACL = { ACE [, ACE ...] }
ACE = { acetype , identifier , aceflags , acemask , acetime }
acetype = uint_t | acetypeitem
identifier = utf8string_t
aceflags = uint_t | aceflagsstring
acemask = uint_t | acemaskstring

acetypeitem = aceallowedtype |
              acedeniedtype |
              aceaudittype
aceallowedtype = "CDMI_ACE_ACCESS_ALLOWED_TYPE" | 0x0
acedeniedtype  = "CDMI_ACE_ACCESS_DENIED_TYPE" | 0x01
aceaudittype   = "CDMI_ACE_SYSTEM_AUDIT_TYPE" | 0x02

aceflagsstring = aceflagsitem [| aceflagsitem ...]
aceflagsitem = aceobinherititem |
               acecontinherititem |
               acenopropagateitem |
               aceinheritonlyitem
```

```
aceobinheritiitem   = "CDMI_ACE_OBJECT_INHERIT_ACE" | 0x01
acecontinheritiitem = "CDMI_ACE_CONTAINER_INHERIT_ACE" | 0x02
acenopropagateitem = "CDMI_ACE_NO_PROPAGATE_INHERIT_ACE" | 0x04
aceinheritonlyitem = "CDMI_ACE_INHERIT_ONLY_ACE" | 0x08

acemaskstring = acemaskitem [| acemaskitem ...]
acemaskitem = acereaditem | acewriteitem |
              aceappenditem | acereadmetaitem |
              acewritemetaitem | acedeleteitem |
              acedelselfitem | acereadaclitem |
              acewriteaclitem | acewriteowneritem
acereaditem = "CDMI_ACE_READ_OBJECT" |
              "CDMI_ACE_LIST_CONTAINER" |      0x01
acewriteitem = "CDMI_ACE_WRITE_OBJECT" |
               "CDMI_ACE_ADD_OBJECT" |         0x02
aceappenditem = "CDMI_ACE_APPEND_DATA" |
                "CDMI_ACE_ADD_SUBCONTAINER" |  0x04
acereadmetaitem = "CDMI_ACE_READ_METADATA" |   0x08
acewritemetaitem = "CDMI_ACE_WRITE_METADATA" | 0x10
acedeleteitem = "CDMI_ACE_DELETE_OBJECT" |
                "CDMI_ACE_DELETE_SUBCONTAINER" | 0x40
acedelselfitem = "CDMI_ACE_DELETE" |          0x10000
acereadaclitem = "CDMI_ACE_READ_ACL" |        0x20000
acewriteaclitem = "CDMI_ACE_WRITE_ACL" |      0x40000
acewriteowneritem = "CDMI_ACE_WRITE_OWNER" |  0x80000
```

When ACE masks are presented in numeric format, they shall, at all times, be specified in hexadecimal notation with a leading "0x". This format allows both servers and clients to quickly determine which of the two forms of a given constant is being used. When masks are presented in string format, they shall be converted to numeric format and then evaluated using standard bitwise operators.

16.1.7 Example ACE Mask Expressions

```
"READ_ALL" | 0x02
```

evaluates to 0x09 | 0x02 == 0x0B

```
0xFFFFF
```

evaluates to 0x000FFFFF == CDMI_ACE_ALL

```
"RW_ALL" | DELETE
```

evaluates to 0x0060006F | 0x00100000 == 0x0070006F

16.1.8 Canonical Format for ACE Hexadecimal Quantities

ACE mask expressions shall always be evaluated and converted to a single hexadecimal value before transmission in an HTTP protocol datagram. Applications or utilities that display them to users should convert them into a text expression prior to display and accept user input in text format as well. The C bitwise operators "|" and "&" should be used for textual representations of bitmask entities.

The following technique should be used to decompose masks into strings. A table of masks and string equivalents should be maintained and ordered from greatest to least:

```
0x1007FFFF    "ALL_PERMS""ALL_PERMS"
   0x0006006F    "RW_ALL""RW_ALL"
   0x0000001F    "RW""RW"
         ...
   0x00000002    "WRITE_OBJECT""ADD_OBJECT"
   0x00000001    "READ_OBJECT""LIST_CONTAINER"
```

Given an access mask M, the following is repeated until M == 0:

1. Select the highest mask *m* from the table such that M & *m* == *m*
2. If the object is a container, select the string from the 3rd column; otherwise select the string from the 2nd column.
3. Bitwise subtract *m* from M, i.e. set M = M xor *m*

The complete textual representation is then all the selected strings concatenated with " | " between them, e.g. "ALL_PERMS | WRITE_OWNER". The strings should appear in the order they are selected.

A similar technique should be used for all other sets of hex/string equivalents.

Note that this algorithm, properly coded, requires only one (often partial) pass through the corresponding string equivalents table.

16.1.9 JSON Format for ACLs

ACE flags and masks are members of a 32-bit quantity that is widely understood in its hexadecimal representations. The JSON data format does not support hexadecimal integers, however. For this reason, all hexadecimal integers in CDMI ACLs shall be represented as quoted strings containing a leading "0x".

ACLs containing one or more ACEs shall be represented in JSON as follows:

```
{
    "cdmi_acl" : [
        {
            "acetype" : "0xnn",
            "identifier" : "<user-or-group-name>",
            "aceflags" : "0xnn",
            "acemask" : "0xnn",
        },
        {
            "acetype" : "0xnn",
            "identifier" : "<user-or-group-name>",
            "aceflags" : "0xnn",
            "acemask" : "0xnn",
        }
    ]
}
```

ACEs in such an ACL shall be evaluated in order as they appear. An example of an ACL embedded in a response to a GET request is as follows:

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0

{
    "objectURI" : "/MyContainer/MyDataItem.txt",
    "objectID" : "0000706D0010734CE0BAEB29DD542B51",
    "objectName" : "MyDataItem.txt",
    "parentURI" : "/MyContainer/",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/DataItem/",
    "completionStatus" : "Complete",
    "mimetype" : "text/plain",
    "metadata" : {
        "cdmi_size" : "17",
        "cdmi_acl" : [
            {
                "acetype" : "0x00",
                "identifier" : "EVERYONE@",
                "aceflags" : "0x00",
                "acemask" : "0x00020089",
            }
        ]
    },
    "valuerange" : "0-17",
    "value" : "Hello CDMI World!"
}
```

16.2 Support for User Metadata

All objects that support metadata shall permit the inclusion of arbitrary user-defined metadata items, with the restriction that the name of a user-defined metadata item shall not start with the prefix "cdmi_".

- The maximum number of user-defined metadata items is specified by the capability "cdmi_metadata_maxitems".
- The maximum size of each user-defined metadata item is specified by the capability "cdmi_metadata_maxsize".

16.3 Support for Storage System Metadata

Once an object has been created, the storage system metadata, as described in Table 19, "Storage System Metadata", shall be generated by the cloud storage system and shall immediately be made

available to a CDMI client in the metadata that is returned as a result of the create operation and any subsequent retrievals.

Table 19 - Storage System Metadata

Metadata Name	Type	Description	Requirement
cdmi_size	JSON String	The number of bytes stored in the data item.	Optional
cdmi_ctime	JSON String	The time when the data item was created, in [ISO-8601] point-in-time format, as described in Section 5.14.	Optional
cdmi_atime	JSON String	The time when the data item was last accessed in [ISO-8601] point-in-time format, as described in Section 5.14. The access or modification of a child is not considered an access of a parent container (access/modify times do not propagate up the tree).	Optional
cdmi_mtime	JSON String	The time when the data item was last modified, in [ISO-8601] point-in-time format, as described in Section 5.14. The modification of a child is not considered a modification of a container (modification times do not propagate up the tree).	Optional
cdmi_acount	JSON String	The number of times that the object has been accessed since it was originally created. Accesses include all reads, writes, and lists.	Optional
cdmi_mcount	JSON String	The number of times that the object has been modified since it was originally created. Modifications include all value and metadata changes. Modifications to metadata resulting from reads (such as updates to *atime*) do not count as a modification.	Optional
cdmi_hash	JSON String	The hash of the value of the object, encoded as a base64 string. This *metadata* field shall be present when the "cdmi_value_hash" data system metadata for the object or a parent object indicates that the value of the object should be hashed.	Optional
cdmi_owner	JSON String	The *cdmi_member_name* of the principal that has owner privileges for the object.	Optional
cdmi_acl	JSON Array	Standard ACL metadata. If not specified when the object is created, this metadata shall be filled in by the system.	Optional

16.4 Support for Data System Metadata

When specified, data system metadata provides guidelines to the cloud storage system on how to provide storage data services for data managed through the CDMI interface.

As described in Table 20, data system metadata is inherited from parent objects to any children. If a child explicitly contains data system metadata, the metadata value of the child data system shall override the metadata value of the parent data system.

Table 20 - Data Systems Metadata

Metadata Name	Type	Description	Requirement
cdmi_data_redundancy	JSON String	Contains the desired number of complete copies of the data item to be maintained. This number determines the minimum number of primary copies of the data that the cloud shall maintain. Additional primary copies may be made to satisfy demand for the value.	Optional
cdmi_immediate_redundancy	JSON String	If present and set to the value "true", indicates that at least a *cdmi_data_redundancy* number of copies shall contain the newly written value before the operation completes. This metadata is used to make sure that multiple copies of the data are written to permanent storage to prevent possible data loss.	Optional
cdmi_assignedsize (only valid for a container.) See Chapter 9, "Container Object Resource Operations".	JSON String	Contains the number of bytes that are reported via exported protocols (and may be thin provisioned by the system). This number may limit *cdmi_size* for the container. This metadata is the size that shall be shown through any number of data path protocols that are used to export a container. If the container is thin provisioned, this may be greater than the actual storage consumed.	Optional
cdmi_infrastructure_redundancy	JSON String	Contains the number of desired independent storage infrastructures supporting the data. This metadata is used to convey that, of the primary copies specified in *cdmi_data_redundancy*, these copies shall be stored on this many separate infrastructures. Any two infrastructures may not share common elements, such as a network or power source.	Optional
cdmi_data_dispersion	JSON String	Contains the desired distance (km) between the infrastructures supporting the multiple copies of data. This metadata is used to separate the (*cdmi_infrastructure_redundancy* number of) infrastructures by a minimum geographic distance to prevent data loss due to site disasters.	Optional
cdmi_geographic_placement	JSON Array of JSON Strings	Contains a list of geopolitical identifiers, each specifying a region where the object is permitted or not permitted to be stored. Geopolitical boundaries are a list of ISO-3166 country codes. A "!" in front of a country code excludes that country from the previous list of geopolitical boundaries. This metadata limits where the data may be placed physically and constrains all cloud movement of the data within the cloud. It does not apply to data once it leaves the cloud. This metadata takes precedent over other metadata, such as *cdmi_data_dispersion*.	Optional

Table 20 - Data Systems Metadata

Metadata Name	Type	Description	Requirement
cdmi_retention_id	JSON String	Contains a user-specified retention identifier. This metadata is a user-specified text field that is used to tag a given object as being managed by a specific retention policy. It is not required to place an object under retention but is useful when needing to be able to perform a query to find all objects under a specific retention policy.	Optional
cdmi_retention_period	JSON String	Contains an [ISO-8601] time interval (as described in Section 5.14) during which the object is under retention. Only the end-date may be altered when updated. If an object is under retention, the object may not be deleted and its value may not be altered. After the retention date has passed, the object may be deleted.	Optional
cdmi_retention_autodelete	JSON String	This metadata is used to indicate if the object is to be automatically deleted when retention expires. The value of this metadata item shall be "true" when set.	Optional
cdmi_hold_id	JSON Array of JSON Strings	This metadata is used to indicate if the object is to be placed under a retention hold. If the array is not empty, the object is under a hold, with each string in the array containing a user-specified hold identifier. If an object is under one or more holds, the object is completely immutable.	Optional
cdmi_encryption	JSON String	This metadata is used to indicate if the object is to be encrypted and indicates the desired encryption algorithm, the mode of operation, and the key size. This metadata is the desired encryption support that the client is requesting of the cloud. All data related to the data item/container shall be encrypted when this value is set, including metadata. This metadata is the desired encryption support that the client is requesting of the cloud. Using the template, ALGORITHM_MODE_KEYLENGTH, the client is able to specify the encryption where: • "ALGORITHM" is the encryption algorithm (e.g., "AES" or "3DES"). • "MODE" is the mode of operation (e.g., "XTS", "CBC", or "CTR"). • "KEYLENGTH" is the key size (e.g., "128", "192", "256").	Optional

Table 20 - Data Systems Metadata

Metadata Name	Type	Description	Requirement
		To improve interoperability between CDMI implementations, the following designators should be used for the more common encryption combinations: • “3DES_ECB_168” for the three-key Triple DES algorithm, the Electronic Code Book (ECB) mode of operation, and a key size of 168 bits. • “3DES_CBC_168” for the three-key Triple DES algorithm, the Cipher Block Chaining (CBC) mode of operation, and a key size of 168 bits. • "AES_CBC_128" for the AES algorithm, the CBC mode of operation, and a key size of 128 bits. • "AES_CBC_256" for the AES algorithm, the CBC mode of operation, and a key size of 256 bits. • "AES_XTS_128" for the AES algorithm, the XTS mode of operation, and a key size of 128 bits. "AES_XTS_256" for the AES algorithm, the XTS mode of operation, and a key size of 256 bits.	
cdmi_value_hash	JSON String	If present, this capability lists the hash algorithm/ lengths supported. If absent, objects shall not present a hash value as system metadata. Values are in the form of "Algorithm Length”, for example, "SHA256". When a CDMI implementation supports hashing, the system-wide capability of “cdmi_security_data_integrity” specified in Table 10 of Section 12.1.1, "Cloud Storage System-Wide Capabilities" shall be set to “true”. Otherwise, it shall not be present, indicating that there is no hashing support.	Optional
cdmi_latency	JSON String	Contains the desired maximum time to first byte, in milliseconds. This metadata is the desired latency (in milliseconds) to the first byte of data in a primary copy, as measured from the edge of the cloud and factoring out any propagation latency between the client and the cloud. For example, this metadata may be used to determine, in an interoperable way, from what type of storage medium the primary copy(s) of the data may be served.	Optional
cdmi_throughput	JSON String	Contains the desired maximum data rate on retrieve, in bytes per second. This metadata is the desired bandwidth (in Mbits/sec) to the primary copy of data, as measured from the edge of the cloud and factoring out any bandwidth capability between the client and the cloud. This metadata is used to stage the primary data copies in locations where there is sufficient bandwidth to accommodate a maximum usage.	Optional
cdmi_sanitization_method	JSON String	If present, this metadata specifies the sanitization method selected from the list in the cdmi_sanitization_method capability list. If absent, objects shall not be securely sanitized.	Optional

Table 20 - Data Systems Metadata

Metadata Name	Type	Description	Requirement
cdmi_RPO	JSON String	Contains the largest acceptable duration in time between an update and when the update may be recovered, specified in seconds. This metadata is used to indicate the desired backup frequency from the primary copy(s) of the data to the secondary copy(s). It is the maximum acceptable duration between a write to the primary copy and the backup to the secondary copy during which a failure of the primary copy(s) shall result in data loss.	Optional
cdmi_RTO	JSON String	Contains the largest acceptable duration in time to restore data, specified in seconds. This metadata is used to indicate the desired maximum acceptable duration to restore the primary copy(s) of the data from a secondary backup copy(s).	Optional

16.5 Support for Data Copies

CDMI envisions a cloud storage system where multiple primary copies of data are stored and distributed geographically for both availability and performance reasons. All of these primary copies may be accessed through the URI for the object, and the system determines which copy may best satisfy the request. CDMI also envisions secondary copies that are not accessible immediately through the URI but may be used to restore the primary copies, if they were destroyed. The metadata for CDMI data copies are described in Table 20, "Data Systems Metadata".

16.6 Support for Provided Data System Metadata

For each metadata element in a data system, there is an actual value that the offering is able to achieve at this time, as shown in Table 21.

Table 21 - Provided Values of Data Systems Metadata Elements

Metadata Name	Type	Description	Requirement
cdmi_data_redundancy_provided	JSON String	Contains the current number of complete copies of the data item at this time.	Optional
cdmi_immediate_redundancy_provided	JSON String	If present and set to "true", indicates if immediate redundancy is provided for the object.	Optional
cdmi_infrastructure_redundancy_provided	JSON String	Contains the current number of independent storage infrastructures supporting the data currently operating.	Optional
cdmi_data_dispersion_provided	JSON String	Contains the current lowest distance (km) between any two infrastructures hosting the data.	Optional
cdmi_geographic_placement_provided	JSON Array of JSON Strings	Each string contains an ISO-3166 identifier that corresponds to a geopolitical region where the object is stored.	Optional

Table 21 - Provided Values of Data Systems Metadata Elements

Metadata Name	Type	Description	Requirement
cdmi_retention_period_provided	JSON String	Contains an [ISO-8601] time interval (as described in Section 5.14) specifying the period the object is protected by retention.	Optional
cdmi_retention_autodelete_provided	JSON String	Contains "true" if the object will automatically be deleted when retention expires.	Optional
cdmi_hold_id_provided	JSON Array of JSON Strings	Contains the user-specified hold identifier for active holds.	Optional
cdmi_encryption_provided	JSON String	Contains the algorithm used for encryption, the mode of operation, and the key size. (See cdmi_encryption in Table 20, "Data Systems Metadata" for the format.)	Optional
cdmi_value_hash_provided	JSON String	Contains the algorithm and length being used to hash the object value.	Optional
cdmi_latency_provided	JSON String	Contains the provided maximum time to first byte.	Optional
cdmi_throughput_provided	JSON String	Contains the provided maximum data rate on retrieve.	Optional
cdmi_sanitization_method_provided	JSON String	Contains the sanitization method used.	Optional
cdmi_RPO_provided	JSON String	Contains the provided duration, in seconds, between an update and when the update may be recovered.	Optional
cdmi_RTO_provided	JSON String	Contains the provided duration, in seconds, to restore data.	Optional

17 Logging

Data that is stored in a cloud-based storage system is governed by a similar set of policy, process, and governance regulations, as if the data were being stored on site or in a private cloud. The obligations to a user of cloud-based storage to satisfy regulating agencies, auditors, and IT oversight functions need to be part of the CDMI standard. As the data stored in clouds may span geo-political boundaries, multiple regulations often need to be satisfied to oversight agencies. To accomplish this, CDMI logging is divided into three functional areas, each with differing levels of detail. These include:

- Logging of CDMI object functions
- Logging of security events
- Logging of data management events

Information contained in these logs shall be for CDMI operations. Logging of non-CDMI operations is optional, but may be provided through a CDMI logging interface.

17.1 Access to Log Data

A CDMI client may access log data by creating a logging queue that indicates the scope of log messages that they wish to receive, as described in Section 17.5, "Logging Queues". If the user has sufficient permissions to create a log queue, all log messages that he or she has subscribed to shall be enqueued into the queue, which may be accessed for processing and archival storage.

If multiple logging queues are defined, each shall get the log entry for a subscribed event. If there are no logging queues defined that subscribe to a given log message or class of log messages, these messages do not have to be retained by the system.

17.2 Object Logging

If logging is supported by the storage system, all operations performed on CDMI objects (data objects, containers, domains, queues, and capabilities) shall be persistently stored into all defined logging queues.

Log messages shall contain a minimum of the following information:

- A timestamp in [ISO-8601] format (see Section 5.14)
- The domain in which the operation was performed
- The operation being performed
- The URI of the object against which the operation was performed
- The principal of the entity by which the operation was performed
- The result of the operation

It is anticipated that a standardized format and set of log messages for CDMI operations will be added to a future release of the standard.

Operations logged shall include serializations and de-serializations, snapshots, export actions, and mappings of exports to file systems. Operations logged should include operations performed to a CDMI-exported file system.

Any storage consumed by the logging queues should count as part of the overall storage consumed by a domain.

CDMI object logging is a superset of object notifications. Notifications are intended to provide high-level information about changes to objects, to provide less detail than object logging, and to be available to a wider group of end users.

17.3 Security Logging

All security-sensitive events, including session establishment, authentication and authorization, and domain modifications and delegation shall be logged as security events. Security logging includes user and domain management, credential-related actions (i.e., revocation list validation) and should include out-of-band operations that affect the security of a CDMI system (such as modifications of security properties of a CDMI domain via an administrative GUI).

If the cloud storage system supports a queue type of "cdmi_logging_queue" and a “cdmi_logging_class” of “cdmi_security_logging” as shown in Section 17.5, "Logging Queues", this indicates that the system supports audit logging. Consequently, the system-wide capability of “cdmi_security_audit” specified in Table 10 of Section 12.1.3, "Data System Metadata Capabilities" shall be set to “true”. Otherwise, it shall not be present.

17.4 Data Management Logging

In addition to log messages associated with the alteration of metadata when changing data system metadata, logging should also include all conditions where the specified or actual data system metadata for objects change. For example, if the number of requested replicas was changed by a client, this change shall generate a log message indicating this change. A corresponding change in the actual number of replicas by the system shall also generate a log message.

This class of logging shall also contain object holds and retention policy log messages.

17.5 Logging Queues

Logging queues allow CDMI clients to obtain detailed logging information about the actions related to the operation of a CDMI storage system. As queue data is persistent, no session state needs to be retained by the client, and clients can operate in a disconnected and casual manner. For example, a logging analysis application might use logging queues to obtain information about actions performed to a cloud, and it may receive only logging messages that are relevant to a domain or set of objects.

Logging queues differ from notification queues in that the information provided is at a much more detailed level than notifications and is typically restricted to a smaller, privileged subset of clients.

When a client wishes to receive logging information, it may first check if the system is capable of providing logging by checking for the presence of the "cdmi_logging" capability in the root capabilities container. If this capability is not present, creating a logging queue shall be successful, but no logging entries shall be enqueued into the logging queue.

When creating a logging queue, the metadata described in Table 22 shall be provided.

Table 22 - Required Metadata for a Logging Queue

Metadata Name	Type	Description	Requirement
cdmi_queue_type	JSON String	The queue type indicates how the cloud storage system shall manage the queue object. The type of "cdmi_logging_queue" is defined for logging queues.	Mandatory
cdmi_logging_class	JSON Array of JSON Strings	Contains a JSON array that indicates which log messages are to be enqueued. Defined values are: • cdmi_object_logging - Receive logging messages related to object operations. • cdmi_datasystem_logging - Receive logging messages related to data system metadata state changes. • cdmi_security_logging - Receive logging messages related to security events.	Mandatory
cdmi_scope_specification	JSON Array of JSON Objects	The scope specification determines the set of objects that operations trigger the generation of log messages. If logging is desired for all objects, include an empty JSON array. For security logging, the scope specification is ignored. See Section 20.1 for how to construct a scope specification.	Mandatory

An example of the metadata associated with a logging queue is as follows:

```
{
    "metadata" : {
        "cdmi_queue_type" : "cdmi_logging_queue",
        "cdmi_logging_class" : [
            "cdmi_object_logging",
            "cdmi_security_logging"
        ],
        "cdmi_scope_specification" : [
            {
                "domainURI" : "== /cdmi_domains/MyDomain/"
            }
        ]
    }
}
```

When logging messages are enqueued into a logging queue, each enqueued value consists of a JSON array of JSON objects, with each JSON object representing a single log message.

Log messages are only included in a logging queue if the user who created the logging queue is able to access the object associated with the log message. (User has any ACE from Section 16.1.5, "ACE Mask Bits".)

Example: If the logging queue was created by the administrator, then all matching objects that the administrator is allowed to read are included in the results. If the notification queue was created by user "jdoe", then only logging messages for objects that "jdoe" is allowed to read are included in the results.

Table 21, "Logging Status Metadata" describes the system-created metadata that provides details on the completion status of the logging queue.

Table 23 - Logging Status Metadata

Metadata Name	Type	Description	Requirement
cdmi_logging_status	JSON String	Indicates if the query is in progress or complete. The three values defined are "`Error`", "`Processing`", and "`Complete`".	Mandatory

When logging results are stored in a logging queue, each enqueued value shall consist of a JSON object of MIME-type "application/json".

17.6 Logging Security Considerations

The accuracy and integrity of the log entries depend on the accuracy and integrity of the clock that is used to set their timestamp values. Accurate timestamps are essential to troubleshooting, forensic analysis of distributed attacks, dispute resolution, and proof of time-sensitive transactions. In essence, debugging, security, audit, and authentication are founded on the basis of event correlation (knowing exactly what happened in what order and on which side), and these security considerations depend on good time synchronization.

While specifying the accuracy and integrity of time keeping is not within the scope of the CDMI standard, to demonstrate that log timestamps are trustworthy, timestamps should be traceable to NIST standard time, and it should be demonstrated that system time may not be arbitrarily changed.

To ensure that log messages have not been changed, log messages should be signed, such that any changes may be detected and the log integrity may be verified. Log verification also requires the demonstration that messages have not been inserted or removed, requiring the presence of sequence counts or similar methods that demonstrate continuity.

18 Retention and Hold Management

A CDMI system may optionally implement retention management disciplines into the system management functionality of the cloud-based storage system. The implementation of retention and hold capabilities is governed by the presence of the CDMI system-wide capabilities for retention and hold capabilities.

Retention management includes implementing a retention policy, defining a hold policy to enable objects to be held for specific purposes (typically litigation), and defining how the rules for deleting objects are affected by placing either a retention policy and/or a hold on an object. CDMI object deletion is not a capability of retention management, per se, but rather is a general system capability. However, this section describes what happens when placing either a retention policy and/or a hold on an object

Retention management may be applied to the following object types:

- Data items
- Queues
- Containers

18.1 Retention Management Disciplines

CDMI retention, deletion, and hold management apply to any CDMI application that creates or deletes CDMI objects, as these disciplines mandate how a CDMI system manages CDMI objects when they are created and until they are deleted.

CDMI retention management is comprised of three management disciplines: retention, hold, and deletion:

- CDMI retention uses retention time criteria to determine the time period during which object deletion from the CDMI-based system is prohibited. No changes to the object are allowed; however, extensions of the object metadata are allowed.
- A CDMI-based system shall not allow the deletion of a CDMI object before the CDMI retention time criteria are met, and any deletion tries (e.g., by a CDMI application) shall generate non-fatal errors.
- After the CDMI retention time criteria have been met, CDMI retention shall no longer be a reason to prohibit object deletion.

18.2 CDMI Retention

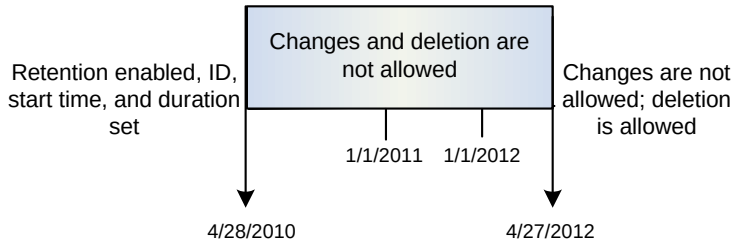
CDMI retention only allows one concurrent retention policy to be applied to an object at a time.

Retention management uses time criteria to determine the time period during which CDMI object deletion from the CDMI-based system shall be prohibited. CDMI retention criteria shall be specified by:

- A retention criteria identifier: a CDMI application-specified string that shall identify the retention records class (cdmi_retention_id)
- A retention start time and retention duration time: the start time, when used together with duration, indicating when retention shall no longer be enforced (cdmi_retention_duration)

When a CDMI application tries to delete an object, the CDMI system shall evaluate all such retention criteria and return a non-fatal error, if any retention criteria have not been met.

Figure 11, "Object Retention" shows how to establish time-based retention with a retention identifier. Object metadata may be extended but not modified.



Example: Retention start date of 4/18/2010 with a duration of 730 days. No holds.

Figure 11 - Object Retention

A specific HTTP error code (403) shall be returned on operations to objects that are under retention period when the CDMI system tries to change or delete the object before the retention duration criteria are met. This failure should be a non-fatal error to the application.

It is not the responsibility of a CDMI system to enforce value changes to the retention duration, as there are valid business reasons to change a retention duration for an object.

18.3 CDMI Hold

CDMI hold enforces read-only data item access and prohibition of object deletion. A CDMI system shall allow multiple holds to be applied to a single object to satisfy multiple hold orders.

While an object is on hold, a CDMI system shall:

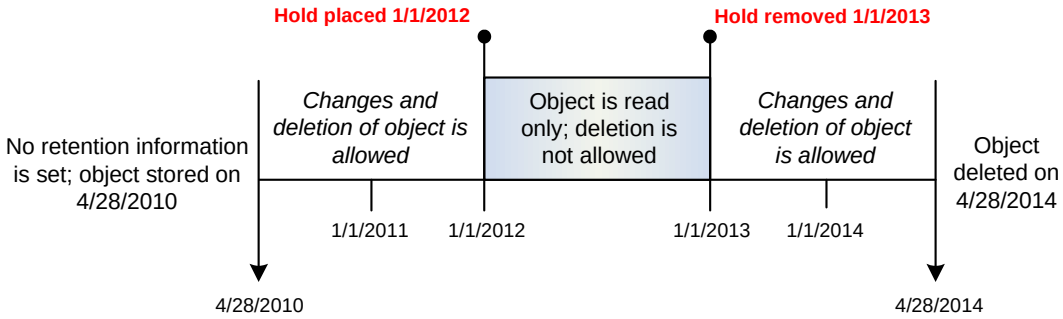
- Strictly enforce read-only access to the object
- Prohibit object deletion

When copying objects that are on hold, hold properties shall not be transferred from the existing CDMI object to the new object, and the new object shall not be on hold.

Hold management uses a hold indicator to determine the time period(s) during which CDMI object revision (data and metadata) and deletion from the CDMI-based system shall be prohibited. CDMI hold criteria shall be specified by data system metadata, specifically, a hold criteria identifier that is an application-specified string that shall identify the holds and their order.

A CDMI application may place an object on hold by adding a hold id to the cdmi_retention_hold data system metadata item. When an object is on hold, CDMI applications shall be subject to failures or unexpected state changes on operations, which would otherwise be successful if the object was not on hold.

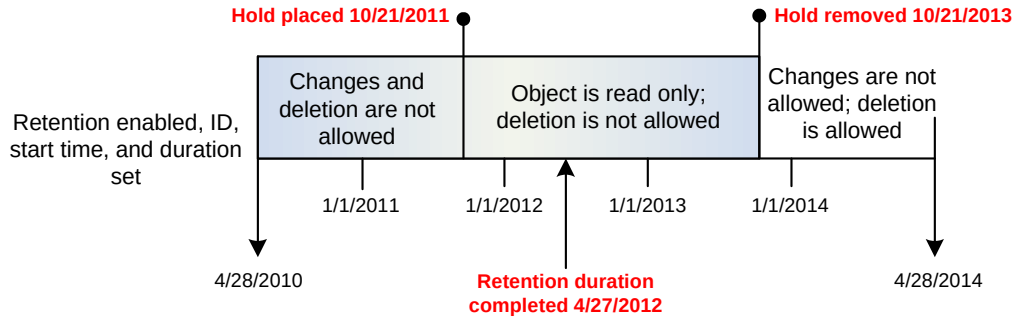
Figure 12, "Object Hold" shows how placing a hold on an object affects its read-only and deletion capability.



Example: Hold placed on the object on 1/1/2012 and removed on 1/1/2013

Figure 12 - Object Hold

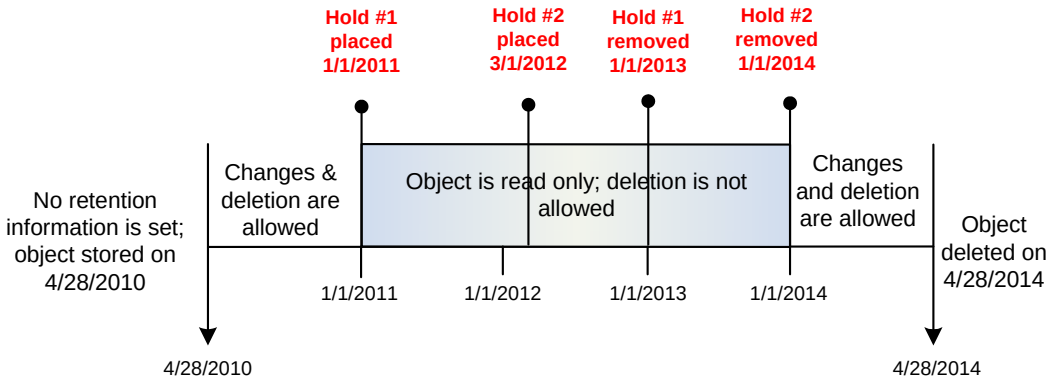
Figure 13, "Object Hold on Object with Retention" shows how to establish time-based retention with a retention identifier that has a hold placed on the object. Object metadata for the retention duration may be extended but not modified.



Example: Start date of 4/28/2010 with a duration of 730 days; hold placed on the object

Figure 13 - Object Hold on Object with Retention

Figure 14, "Object with Multiple Holds" shows how placing multiple holds on an object affects its read-only and deletion capability.



Example: Object created on 4/28/2010.
Hold #1 is placed on 1/1/2011 and removed on 1/1/2013.
Hold #2 is placed on 3/1/2012 and removed on 1/1/2014.

Figure 14 - Object with Multiple Holds

A CDMI system shall maintain an on-hold object in read-only mode with respect to the application access to data and metadata and shall prohibit deletion, either automated or explicit.

- CDMI applications shall tolerate these object on-hold failures or state changes.
- Releases from hold are performed out-of-band or by vendor extension, and are not part of the CDMI standard.

A specific HTTP error code (403) shall be returned on operations to objects that are under a hold when the system tries to change the object or tries to delete the object before the hold is removed. This failure should be a non-fatal error to the application.

18.4 CDMI Deletion

CDMI deletion controls CDMI system actions with respect to object deletion. A CDMI system may automatically delete a CDMI object once the retention time and hold criteria have been met. (See *cdmi_retention_autodelete* in Table 20, "Data Systems Metadata".

CDMI objects shall be automatically deleted by the system by setting the data system metadata flag, *cdmi_retention_autodelete*, at the retention duration expiration. The *cdmi_retention_autodelete* flag indicates to the system that the object shall be made unavailable for access once the retention criteria have been satisfied. The system shall ensure that the object is no longer available through the CDMI interface. If the system has satisfied the retention requirement and a hold is established for the object, the object shall not be made unavailable or deleted. When a hold and retention have been applied to an object, both need to be satisfied (retention duration expired and no holds existing) for objects to be automatically deleted from the system.

The autodeletion flag is used with the retention/hold capability of a CDMI system.

18.5 Retention Security Considerations

The accuracy and integrity of the retention start and elapsed times depend on the accuracy and integrity of the clock that is used to set their values. Equally important is the relative accuracy and integrity of the clock, which determines if retention duration has elapsed, to the clock, which sets the start time property. Relative time differences between these two clocks may lead to undesirable retention and deletion management behavior.

It is important to have a reliable source from which the system clock is set. A stratum 1 time is directly connected to a reference clock and is at the top of the time server hierarchy. Relative time differences between the system clock and the reference clock may lead to undesirable retention timestamps and difficulties with time action events. For example, an object is created in an CDMI system at time 0 with duration of 8 years and autodelete of TRUE. At time 1 year, the system clock is adjusted forward to 9 years. Now, because the system time is 9 years, the retention time criterion is satisfied, even though only 1 year has actually elapsed. And, since autodelete is TRUE, the system automatically deletes the object.

The specification for accuracy and integrity of timekeeping is not within the scope of CDMI. However, to prevent undesirable retention and deletion management consequences, systems are strongly encouraged to maintain accurate clock time, with zero or minimal deviation to clock integrity.

19 Notification Queues

A CDMI system may optionally implement notification functionality. The implementation of notification is indicated by the presence of the CDMI system-wide capabilities for notification and requires support for CDMI queues.

Notification queues allow CDMI clients to efficiently discover what changes have occurred to the system. As queue data is persistent, no session state needs to be retained by the client, and clients may operate in a disconnected and casual manner. For example, an application might use notification queues to keep its database current without having to do full scans of a container to discover what objects have been added, modified, or removed.

When a client wishes to receive notifications, it may first check if the system is capable of providing notifications by checking for the presence of the "cdmi_notifications" capability in the root capabilities container. If this capability is not present, creating a notification queue shall be successful, but no notifications shall be enqueued into the notification queue.

To create a notification queue, the client creates a regular CDMI queue and adds metadata instructing the storage system to treat the queue as a notification queue. This added metadata also instructs the system about what types of notifications shall be generated and what information shall be included with each notification.

Once the notification queue is created, all subsequent matching events after the queue creation time shall result in notification results being enqueued into the queue. CDMI does not mandate any specific ordering of events, and clients must be able to handle events that arrive out of order.

When creating a notification queue, the metadata described in Table 24 shall be provided. Attempts to alter metadata in this table will result in an `HTTP 403 Forbidden` HTTP status code. Once a notification queue has been created, with the exception of *cdmi_queue_type*, the metadata items in this table cannot be altered. *cdmi_queue_type* can only be removed, indicating to the system that the notification queue shall no longer receive notifications and shall be treated as a regular CDMI queue object.

Table 24 - Required Data for a Notification Queue

Metadata Name	Type	Description	Requirement
cdmi_queue_type	JSON String	The queue type indicates how the cloud storage system shall manage the queue object. The type of "cdmi_notification_queue" is defined for notification queues.	Mandatory
cdmi_notification_events	JSON Array of JSON Strings	Contains a JSON array that indicates which events generate notifications. Defined values are: • *cdmi_create_processing* - Notifications are generated when a new object is created, but is still in the processing completion status. • *cdmi_create_complete* - Notifications are generated when a new object is created and is in the complete completion status. This notification is also generated when a new object being created transitions from “`Processing`” to “`Complete`”. • *cdmi_read* - Notifications are generated when an object is read. • *cdmi_modify_processing* - Notifications are generated when an existing object is modified, but is still in the processing completion status.	Mandatory

Table 24 - Required Data for a Notification Queue

Metadata Name	Type	Description	Requirement
		• *cdmi_modify_complete* - Notifications are generated when an existing object is modified and is in the complete completion status. This notification is also generated when an existing object being modified transitions from "`Processing`" to "`Complete`". • *cdmi_rename* - Notifications are generated when an object is renamed. • *cdmi_copy* - Notifications are generated when an object is copied. • *cdmi_reference* - Notifications are generated when an object is referenced. • *cdmi_delete* - Notifications are generated when an object is deleted. • *cdmi_export* - Notifications are generated when an container is exported. • *cdmi_snapshot* - Notifications are generated when an container is snapshotted. • <implementor-specific events>	
cdmi_scope_specification	JSON Array of JSON Objects	The scope specification determines the set of objects that on which operations trigger the generation of notifications. If notifications are desired for all objects, include an empty JSON array See Section 20.1 for how to construct a scope specification.	Mandatory
cdmi_results_specification	JSON Object	Contains the JSON fields to be returned for each object that matches the notification scope specification. See Section for how to construct a results specification. In addition to the fields defined in Section , for notifications, two additional fields are defined: • *cdmi_event* - Indicates the event as specified in the *cdmi_notification_events* field that triggered the notification. • *cdmi_event_result* - Indicates the status result of the event that triggered the notification. The status is the same as the status that was returned over the HTTP request, i.e., `200 OK`, `404 Not Found`, etc. • *cdmi_event_time* - Indicates the time of the event that triggered the notification. The time will be formatted in [ISO-8601] time (see Section 5.14). • *cdmi_event_user* - Indicates the principal (*acl name*) of the user that caused the event that triggered the notification. If the system triggered the event, the name will be left as an empty string.	Mandatory

An example of the metadata associated with a notification queue is as follows:

```
{
    "metadata" : {
        "cdmi_queue_type" : "cdmi_notification_queue",
        "cdmi_notification_events" : [
            "cdmi_create_complete",
            "cdmi_read",
            "cdmi_modify",
            "cdmi_delete"
        ],
        "cdmi_scope_specification" : [
            {
                "domainURI" : "== /cdmi_domains/MyDomain/",
                "objectURI" : "starts /sandbox",
                "metadata" : {
                    "cdmi_size" : "> 100000"
                }
            }
        ],
        "cdmi_results_specification" : {
            "cdmi_event" : "",
            "cdmi_event_result" : "",
            "cdmi_event_time" : "",
            "objectID" : "",
            "metadata" : {
                "cdmi_size" : ""
            }
        }
    }
}
```

When notification results are stored in a notification queue, each enqueued value shall consist of a JSON object of MIME-type "application/json". This JSON object contains the specified values requested in the *cdmi_results_specification* of the notification queue metadata.

An example of a notification result JSON object is as follows:

```
{
    "cdmi_event" : "cdmi_read",
    "cdmi_event_result" : "200 OK",
    "cdmi_event_time" : "2010-11-15T13:12:52.342324",
    "objectID" : "0000706D0010B84FAD185C425D8B537E",
    "metadata" : {
        "cdmi_size" : "108263"
    }
}
```

Objects shall only be included in the notification results if the user who created the notification queue is able to read the matching object.

Example: If the notification queue was created by the administrator, then all matching objects that the administrator is allowed to read are included in the results. If the notification queue was created by user "jdoe", then only matching objects that "jdoe" is allowed to read are included in the results.

Table 25, "Notification Status Metadata" describes the system-created metadata that provides details on the completion status of the notification queue.

Table 25 - Notification Status Metadata

Metadata Name	Type	Description	Requirement
cdmi_notification_status	JSON String	Indicates if the query is in progress or complete. The two values defined are "`Error`", "`Processing`", and "`Complete`".	Mandatory

20 Query Queues

A CDMI system may optionally implement metadata and/or full-text query functionality. The implementation of query is indicated by the presence of the CDMI system-wide capabilities for query and requires support for CDMI queues.

Query queues allow CDMI clients to efficiently discover what content matches a given set of metadata query criteria or full-content search criteria. Clients create or update a query queue by specifying metadata that defines the matching criteria (known as the query scope), along with what results should be returned for matching objects (known as the query results). The CDMI offering shall then perform the query, storing the query results in the query queue. As query results are found, they are added to the queue, and when the query is complete, the *query_status* metadata of the queue is changed to indicate that the query has completed.

When a client wishes to perform queries, it may first check if the system is capable of providing query functionality by checking for the presence of the "cdmi_query" capability in the root capabilities container. If this capability is not present, creating a query queue shall be successful, but no query results shall be enqueued into the query queue.

When creating a query queue, the metadata described in Table 26, "Required Metadata for a Query Queue" shall be provided. Attempts to alter metadata in this table will result in an `HTTP 403 Forbidden` HTTP status code. Once a query queue has been created, with the exception of *cdmi_queue_type*, the metadata items in this table cannot be altered. *cdmi_queue_type* can only be removed, indicating to the system that the query queue shall no longer receive query results and shall be treated as a regular CDMI queue object.

Table 26 - Required Metadata for a Query Queue

Metadata Name	Type	Description	Requirement
cdmi_queue_type	JSON String	The queue type indicates how the cloud storage system shall manage the queue object. The type of "cdmi_query_queue" is defined for query queues.	Mandatory
cdmi_scope_specification	JSON Array of JSON Objects	The scope specification determines which objects are included in the query results. This is equivalent to a "WHERE" clause in SQL-like languages. See Section 20.1 for how to construct a scope specification.	Mandatory
cdmi_results_specification	JSON Object	Contains the JSON fields to be returned for each object that matches the query. This is equivalent to a "SELECT" clause in SQL-like languages . See Section for how to construct a results specification.	Mandatory

An example of the metadata associated with a query queue is as follows:

```
{
    "metadata" : {
        "cdmi_queue_type" : "cdmi_query_queue"
        "cdmi_scope_specification" : [
            {
                "domainURI" : "== /cdmi_domains/MyDomain/",
                "objectURI" : "starts /sandbox",
                "metadata" : {
                    "cdmi_size" : "> +100000"
                }
            }
        ],
```

```
        "cdmi_results_specification" : {
            "objectID" : "",
            "metadata" : {
                "cdmi_size" : ""
            }
        }
    }
}
```

When results are stored in a query queue, each enqueued value shall consist of a JSON object of MIME-type "application/json". This JSON object contains the specified values requested in the *cdmi_results_specification* of the query queue metadata.

An example of a query result JSON object is as follows:

```
{
    "objectID" : "0000706D0010B84FAD185C425D8B537E",
    "metadata" : {
        "cdmi_size" : "108263"
    }
}
```

Table 27, "Query Status Metadata" describes the system-created metadata that provides details on the completion status of the query queue.

Table 27 - Query Status Metadata

Metadata Name	Type	Description	Requirements
cdmi_query_status	JSON String	A string indicating if the query is in progress or has completed. The value shall be the string "`Processing`", the string "`Complete`", or an error string starting with the value "`Error`".	Mandatory

Objects shall only be included in the query results if the user who created the query queue is able to read the matching objects or metadata.

Example: If the query queue was created by the administrator, then all matching objects that the administrator is allowed to read are included in the results. If the query queue was created by user "jdoe", then only matching objects that "jdoe" is allowed to read are included in the results.

20.1 Scope Specification

Each JSON object within the scope specification represents a set of conditions that shall all be true in order for the query to return an object as part of the query results (a logical AND relationship). Multiple JSON objects are used to express logical OR relationships.

Each JSON object is constructed using the same structure that CDMI objects use. To show this, assume the following result from a CDMI GET for a data object:

```
HTTP/1.1 200 OK
X-CDMI-Specification-Version: 1.0

{
     "objectURI" : "/MyContainer/MyDataObject.txt",
     "objectID" : "0000706D0010B84FAD185C425D8B537E",
     "objectName" : "MyDataObject.txt",
     "parentURI" : "/MyContainer/",
     "domainURI" : "/cdmi_domains/MyDomain/",
     "capabilitiesURI" : "/cdmi_capabilities/DataObject/",
     "completionStatus" : "Complete",
     "mimetype" : "text/plain",
     "metadata" : {
       "cdmi_size" : "108263"
     },
     "valuerange" : "0-108262",
     "value" : "..."
}
```

Each field inside a scope specification JSON object represents a condition that shall be met for a field. For example, a query to find all objects belonging to the domain `/cdmi_domains/MyDomain/` is structured as follows:

```
[
    {
        "domainURI" : "== /cdmi_domains/MyDomain/"
    }
]
```

To query for all objects belonging to the domain `/cdmi_domains/MyDomain/` AND are also located within the container `MyContainer`, the scope specification is structured as follows:

```
[
    {
        "parentURI" : "== /MyContainer/",
        "domainURI" : "== /cdmi_domains/MyDomain/"
    }
]
```

To query for all objects that belong to the domain `MyDomain` OR are located within the container `MyContainer`, the query is structured as follows:

```
[
    {
        "parentURI" : "== /MyContainer/",
    },
    {
        "domainURI" : "== /cdmi_domains/MyDomain/"
    }
]
```

Queries may match on any field within an object that a CDMI system is capable of returning as a result of an object GET.

To query metadata items, the metadata object is included as an object within the query request. This query is shown as follows:

```
[
    {
        "metadata" : {
            "colour" : "== blue"
        }
    }
]
```

This approach allows the matching against arbitrarily nested metadata structures.

Table 28 defines the query matching expressions.

Table 28 - Query Matching Expression

Matching Expression	Description
"field" : "*"	The exists matching expression tests for the existence of the field. If the field is present, even if empty, the condition is considered to be met.
"field" : "!*"	The not exists matching expression tests for the non-existence of the field. If the field is absent, the condition is considered to be met.
"field" : "== constant"	The equals matching expression tests for the equality of the value of the field and a specified constant value. The equality test is case sensitive. The leading space after the "==" and before the constant value is not included in the comparison. If the constant value matches the value of the field, the condition is considered to be met.
"field" : "!= constant"	The not equals matching expression tests for the non-equality of the value of the field and a specified constant value. The not-equals test is case sensitive. The leading space character after the "!=" and before the constant value is not included in the comparison. If the constant value does not match the value of the field, the condition is considered to be met.
"field" : "> constant"	The greater than matching expression tests if the value of the field is lexicographically greater than a specified constant value. The greater than test is case sensitive. The leading space character after the ">" and before the constant value is not included in the comparison. If the constant value is greater than the value of the field, the condition is considered to be met. If the constant starts with a "+" or "-" sign, the value of the field is considered to be numeric for the purposes of comparison.
"field" : ">= constant"	The greater than or equals to matching expression tests if the value of the field is lexicographically greater than or equal to a specified constant value. The greater than or equals to test is case sensitive. The leading space character after the ">=" and before the constant value is not included in the comparison. If the constant value is greater than or equal to the value of the field, the condition is considered to be met. If the constant starts with a "+" or "-" sign, the value of the field is considered to be numeric for the purposes of comparison.

Table 28 - Query Matching Expression

Matching Expression	Description
"field" : "< constant”	The less than operator tests if the value of the field is lexicographically less than a specified constant value. The less than test is case sensitive. The leading space character after the "<" and before the constant value is not included in the comparison. If the constant value is less than the value of the field, the condition is considered to be met. If the constant starts with a "+" or "-" sign, the value of the field is considered to be numeric for the purposes of comparison.
"field" : "<= constant”	The less than or equals to matching expression tests if the value of the field is lexicographically less than or equal to a specified constant value. The less than or equal test is case sensitive. The leading space character after the "<=" and before the constant value is not included in the comparison. If the constant value is less than or equal to the value of the field, the condition is considered to be met. If the constant starts with a "+" or "-" sign, the value of the field is considered to be numeric for the purposes of comparison.
"field" : "starts constant”	The starts with matching expression tests if the field value starts with a specified constant value. The leading space character after the "starts" and before the constant value is not included in the comparison. The starts with test is case sensitive. If the constant value is equal to the start of the value of the field, the condition is considered to be met.
"field" : "!starts constant”	The not starts with matching expression tests if the field value does not start with a specified constant value. The leading space character after the "!starts" and before the constant value is not included in the comparison.The not starts with test is case sensitive. If the constant value is not equal to the start of the value of the field, the condition is considered to be met.
"field" : "ends constant”	The ends with matching expression tests if the field value ends with a specified constant value. The leading space character after the "ends" and before the constant value is not included in the comparison. The ends with test is case sensitive. If the constant value is equal to the end of the value of the field, the condition is considered to be met
"field" : "!ends constant”	The not ends with matching expression tests if the field value does not end with a specified constant value. The leading space character after the "!ends" and before the constant value is not included in the comparison. The not ends with test is case sensitive. If the constant value is not equal to the end of the value of the field, the condition is considered to be met.
"field" : "contains constant”	The contains matching expression tests if the field value contains a specified constant value. The leading space character after the "contains" and before the constant value is not included in the comparison. The contains test is case sensitive. If the constant value is found as a substring within the value of the field, the condition is considered to be met. The contains operator is only supported if the "cdmi_query_contains" capability is present.

Table 28 - Query Matching Expression

Matching Expression	Description
"field" : "!contains constant”	The not contains matching expression tests if the field value does not contain a specified constant value. The leading space character after the "!contains" and before the constant value is not included in the comparison. The not contains test is case sensitive. If the constant value is not found as a substring within the value of the field, the condition is considered to be met. The not contains operator is only supported if the "cdmi_query_contains" capability is present.
"field" : "tag constant"	The tag matching expression tests if the field value contains a specified constant tag value. The leading space character after the "tag" and before the constant value is not included in the comparison. The tag test is not case sensitive. If the constant value is found as a tag substring within the value of the field, the condition is considered to be met. Tag substrings start at the beginning of the value or a ",", and end at the next "," or the end of the string. Whitespace before and after "," characters shall be stripped for the purpose of comparisons. Tag matching expressions are only supported if the "cdmi_query_tags" capability is present.
"field" : "!tag constant"	The not tag matching expression tests if the field value does not contain a specified constant tag value. The leading space character after the "!tag" and before the constant value is not included in the comparison. The not tag test is not case sensitive. If the constant value is not found as a tag substring within the value of the field, the condition is considered to be met. Tag substrings start at the beginning of the value or a ",", and end at the next "," or the end of the string. Whitespace before and after "," characters shall be stripped for the purpose of comparisons. Tag matching expressions are only supported if the "cdmi_query_tags" capability is present.
"field" : "=~ constant"	The regular expression matching expression tests if the field value matches a specified constant regular expression value. The leading space character after the "=~" and before the constant value is not included in the comparison. If the regular expression evaluates to true against the value, the condition is considered to be met. Regular expression strings shall be processed according to the POSIX Extended Regular Expression (ERE) standard, as specified in The Open Group Base Specifications Issue 6, IEEE Std 1003.1, 2004 Edition. Regex matching expressions are only supported if the "cdmi_query_regex" capability is present.
"field" : "!~ constant"	The not regular expression matching expression tests if the field value does not match a specified constant regular expression value. The leading space character after the "!~" and before the constant value is not included in the comparison. If the regular expression evaluates to false against the value, the condition is considered to be met. Regular expression strings shall be processed according to the POSIX Extended Regular Expression (ERE) standard, as specified in The Open Group Base Specifications Issue 6, IEEE Std 1003.1, 2004 Edition. Regex matching expressions are only supported if the "cdmi_query_regex" capability is present.

All fields in objects that are not included in the scope specification shall be ignored for the purpose of matching objects.

When a URI is used as the constant for the equals and not equals operators against *objectURI*, *parentURI*, *domainURI*, and *capabilitiesURI*, either a URI by path or URI by Object ID can be specified and are considered interchangeable. For example, in a query to find all objects belonging to a specific domain, the following two query scopes are considered identical:

```
[
    {
        "domainURI" : "== /cdmi_domains/MyDomain/"
    }
]
```

and

```
[
    {
        "domainURI" : "== /cdmi_objectid/0000706D0010171EADF15DE7BC0917D3/"
    }
]
```

Likewise, a query to find all objects with a given parent container would have two equivalent forms:

```
[
    {
        "parentURI" : "== /myContainer/"
    }
]
```

and

```
[
    {
        "parentURI" : "== /cdmi_objectid/0000706D0010E0981215538EE7D19E5E/"
    }
]
```

If an Object ID is used in a query scope, such as in the *objectID* field, in the *objectName* field, when the objectPath is set to `"/cdmi_objectid/"`, and as part of a URI compared against the *objectURI*, *parentURI*, *capabiliesURI* and *domainURI* fields, all object IDs shall be processed such that they are case insensitive.

20.2 Results Specification

Each JSON object within the results specification represents a set of fields that are returned for each matching object.

The results JSON object shall be constructed using the same structure as is used for CDMI objects. To show this, assume the following result from a CDMI GET for a data object:

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.0

{
     "objectURI" : "/MyContainer/MyDataObject.txt",
     "objectID" : "0000706D0010B84FAD185C425D8B537E",
     "objectName" : "MyDataObject.txt",
     "parentURI" : "/MyContainer/",
     "domainURI" : "/cdmi_domains/MyDomain/",
     "capabilitiesURI" : "/cdmi_capabilities/DataObject/",
     "completionStatus" : "Complete",
     "mimetype" : "text/plain",
     "metadata" : {
       "cdmi_size" : "108263"
     },
     "valuerange" : "0-108262",
     "value" : "..."
}
```

Each field inside a results specification JSON object indicates that the field shall be included in the results. For example, the following results specification requests that the *objectID* and *cdmi_size* metadata fields to be returned in the results:

```
{
    "cdmi_results_specification" : {
        "objectID" : "",
        "metadata" : {
            "cdmi_size" : ""
        }
    }
}
```

If an object matched, the result JSON enqueued is as follows:

```
{
    "objectID" : "0000706D0010B84FAD185C425D8B537E",
    "metadata" : {
        "cdmi_size" : "108263"
    }
}
```

For most common use cases, either the Object ID or Object URI will be requested in the *cdmi_results_specification*. If the Object URI is included, it is up to the implementation to choose when a Object URI by path or an Object URI by ID should be returned, and both are equally valid. If the Object Name and/or parent URI are included, and paths are supported, the implementation shall return the object name and object path, respectively. If the Object URI is included in the results and the object has a path, then the path shall be returned in this field.

If a client wants to have all metadata fields returned for each matching object, the following *cdmi_results_specification* shall be used:

```
{
    "cdmi_results_specification" : {
        "metadata" : ""
    }
}
```

If a client wants to have all fields and metadata returned for each matching object, the following *cdmi_results_specification* shall be used:

```
{
    "cdmi_results_specification" : ""
}
```

20.3 Extending CDMI Query

An implementor of a CDMI server may extend CDMI query by adding vendor-specific matching expressions. When an implementor adds vendor-specific metadata fields, these fields shall be queried using the standard query queue functionality.

An implementor of a CDMI server may extend CDMI query by allowing the creation of vendor-specific query queues with a type other than "cdmi_query_queue".

Annex A
(normative)
Transport Security

For most CDMI implementations, the Hypertext Transfer Protocol (HTTP) is the underlying communications protocol used to transfer CDMI messages. This appendix identifies the details associated with securing this underlying transport.

A.1 General Requirements for HTTP Implementations

The security requirements for HTTP implementations apply to both CDMI servers and clients. A CDMI client shall comply with all security requirements for HTTP that apply to clients. The following general requirements support security when using HTTP.

- Either HTTP basic authentication or HTTP digest authentication should be implemented.
- To minimize compromising user identities and credentials, such as passwords, implementations should use HTTP basic authentication ONLY in conjunction with Transport Layer Security (TLS).
- A user identity and credential used with one type of HTTP authentication (i.e., basic or digest) should never be subsequently used with the other type of HTTP authentication. To avoid compromising the integrity of a stronger scheme, established good security practices avoids the reuse of identity and credential information across schemes of different strengths.
- TLS 1.0 shall be implemented by CDMI entities and a more current version of TLS (for example, v1.1 and v1.2) is strongly encouraged. The use of TLS by CDMI entities is optional, but should be used to protect sensitive data.
- Although HTTP shall be implemented by all CDMI entities, its use is optional.

The following requirements for implementations and optional use of HTTP over TLS (HTTPS) apply:

- The following cipher suites shall be supported to ensure a minimum level of security and interoperability between implementations:
  - TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA (mandatory for TLS 1.0)
  - TLS_RSA_WITH_AES_128_CBC_SHA (mandatory for TLS 1.1/1.2)
  - TLS_RSA_WITH_NULL_SHA (for TLS without encryption)

  Note: Implementors are free to include additional cipher suites, but there is no guarantee of interoperability when they are used.
- For clients and servers to communicate, they need to be using a consistent approach to security. Properly configured clients and servers may fail to communicate, if one is relying on port 80 and the other on port 443. Clients that fail to connect to a CDMI server via HTTP over TLS on TCP port 443 should retry with HTTP on TCP port 80 if their security policy allows it.
- Servers may accelerate discovery that a secure channel is needed by responding to HTTP contacts on TCP port 80 with a HTTP REDIRECT to the appropriate HTTPS: URL (HTTP over

TLS on TCP port 443) to avoid the need for clients to timeout the HTTP contact attempt. Clients should honor such redirects in this situation.

— All certificates, including CA Root Certificates used by clients for certificate validation, shall be replaceable.

— The DER encoded X.509, Base64 encoded X.509, and PKCS#12 certificate formats shall be supported.

— Certificate Revocation Lists shall be supported in the DER encoded X.509 and Base64 encoded X.509 formats.

Note: Since there are no absolutes when it comes to security, when specified versions are found to be vulnerable and/or inadequate, CDMI implementations should move to a newer version of TLS and stronger cipher suites as soon as possible.

A.2 Basic HTTP Security

HTTP is the mandatory transport mechanism for this version of CDMI. It is important to note that HTTP, by itself, offers no confidentiality or integrity protections.

CDMI clients may be responsible for initiating user authentication for each CDMI server that a user accesses. The CDMI server functions as the authenticator, and it receives the user credentials from the HTTP authentication operations.

IETF RFC 2616 and IETF RFC 2617 define requirements for HTTP authentication, which generally starts with an HTTP client request, such as `GET Request-URI` (where `Request-URI` is the resource requested). If the client request does not include an "Authorization" header line and authentication is required, the server responds with a `401 Unauthorized` status code, and a `WWW-Authenticate` header line. The HTTP client shall then respond with the appropriate `Authorization` header line in a subsequent request. The format of the `WWW-Authenticate` and `Authorization` header lines varies depending on the type of authentication required, basic authentication, or digest authentication. If the authentication is successful, the HTTP server shall respond with a status code of `200 OK`.

Basic authentication involves sending the user name and password in the clear, and it should only be used on a secure network or in conjunction with a mechanism that ensures confidentiality, such as Transport Layer Security (TLS). (See Section A.3, "HTTP over TLS (HTTPS)"). Digest authentication sends a secure digest of the user name and password (and other information including a nonce value), so that the password is not revealed. `401 Unauthorized` responses should not include a choice of authentication.

Client authentication to the CDMI server is based on an authentication service (local and/or external). Differing authentication schemes may be supported, including host-based authentication, Kerberos, PKI, or other; the authentication service is out scope of this standard.

A.3 HTTP over TLS (HTTPS)

CDMI may also include a mechanism to secure HTTP communications, such that data sent between the clients and servers are encrypted before being sent over the network. This security is achieved by transmitting HTTP over TLS (also known as HTTPS); the URL of a secure connection shall begin with `https://` instead of `http://`. It is also important to note that a CDMI client communicates with a CDMI server via HTTPS on TCP port 443 (TCP port 80 is used for HTTP). Section A.3.1, "Transport Layer Security (TLS)" provides important details on TLS.

When TLS is used to secure HTTP, the client and server typically perform some form of entity authentication. However, the specific nature of this entity authentication depends on the cipher suite negotiated; a cipher suite specifies the encryption algorithm and digest algorithm to use on a TLS

connection. A very common scenario involves the use of server-side certificates, which the client trusts, as the basis for unidirectional entity authentication. It is possible that no authentication will occur (e.g., anonymous authentication) or on the other extreme, mutual authentication involving both client-side and server-side certificates may be required.

A.3.1 Transport Layer Security (TLS)

CDMI servers shall implement the TLS protocol; however, its use by clients is optional. TLS 1.0, which shall be implemented, is specified in [RFC2246], and the TLS 1.1 and TLS 1.2 should be implemented as specified in [RFC4346] and [RFC5246], respectively.

The primary goal of the TLS protocol is to provide privacy and data integrity between two communicating applications. TLS allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery. TLS is layered on top of some reliable transport protocol (e.g., TCP) and is used for encapsulating various higher-level protocols (e.g., HTTP).

TLS provides endpoint authentication and communications privacy over the network using cryptography. Typically, only the server is authenticated (i.e., its identity is ensured), while the client remains unauthenticated; this means that the end user (whether an individual or an application) has a measure of assurance with whom they are communicating. Mutual authentication (the identities of both endpoints are verified) requires, with few exceptions, the deployment of digital certificates on the client.

TLS involves three basic phases:

- Peer negotiation for algorithm support
- Key exchange and authentication
- Symmetric cipher encryption and message authentication

During the first phase, the client and server negotiate cipher suites (see Section A.3.1.1, "Cipher Suites"), which determine the ciphers to be used, the key exchange, authentication algorithms, and the message authentication codes (MACs). The key exchange and authentication algorithms are typically public key algorithms. The MACs are made up from a keyed-Hash Message Authentication Code, or HMAC.

A.3.1.1 Cipher Suites

TLS packages one key establishment, confidentiality, signature and hash algorithm into a "cipher suite." A registered 16-bit (4 hexadecimal digit) number, called the cipher suite index, is assigned for each defined cipher suite. For example, RSA key agreement, RSA signature, Advanced Encryption Standard (AES) using Cipher Block Chaining (CBC) confidentiality, and the Secure Hash Algorithm (SHA-1) hash are assigned the hexadecimal value {0x000F} for TLS.

The client always initiates the TLS session and starts cipher suite negotiation by transmitting a handshake message that lists the cipher suites (by index value) that it will accept. The server responds with a handshake message indicating which cipher suite it selected from the list or an "abort" as described below. Although the client is required to order its list by increasing "strength" of cipher suite, the server may choose ANY of the cipher suites proposed by the client. Therefore, there is NO guarantee that the negotiation will select the strongest suite. If no cipher suites are mutually supported, the connection is aborted. When the negotiated options, including optional public key certificates and random data for developing keying material to be used by the cryptographic algorithms, are complete, messages are exchanged to place the communications channel in a secure mode.

To ensure a minimum level of security and interoperability between implementations:

- All CDMI clients and servers shall support TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA cipher suite (hexadecimal value {0x0013}), which is also the mandatory cipher suite for TLS 1.0 (see [RFC2246] Section 9, Mandatory Cipher Suites).

- TLS_RSA_WITH_AES_128_CBC_SHA cipher suite (hexadecimal value {0x002F}) shall be implemented, which is the mandatory cipher suite for both TLS 1.1 and TLS 1.2.
- TLS_RSA_WITH_NULL_SHA cipher suite (hexadecimal value {0x0002}) shall be supported by both CDMI clients and servers to implement authenticated, non-encrypted communications.
- TLS_RSA_WITH_AES_128_CBC_SHA256 cipher suite (hexadecimal value {0x003C}) should be included with all recommended TLS 1.2 implementations to meet the transition to a security strength of 112 bits (guidance is provided in NIST Special Publication 800-57 and NIST Special Publication 800-131A).

Implementors are free to include additional cipher suites, but must prefer the mandatory ones in negotiation.

A.3.1.2 Digital Certificates

CDMI clients and servers may be attacked by setting up a false CDMI server to capture userids and passwords or to insert itself as an undetected proxy between a CDMI client and server. The most effective countermeasure for this attack is the controlled use of server certificates with TLS, matched by client controls on certificate acceptance on the assumption that the false server will be unable to obtain an acceptable certificate. Specifically, this may be accomplished by configuring clients to always use TLS underneath HTTP authentication, and only accept certificates from a specific local certificate authority.

When used by CDMI, TLS shall use X.509 version 3 public key certificates that conform to the Certificate and Certificate Extension Profile defined in Section 4 of [RFC3280] (X.509v3 Certificate and CRL). This certificate and certificate revocation list (CRL) profile specifies the mandatory fields that shall be included in the certificate, as well as optional fields and extensions that may be included in the certificate.

Server certificates shall be supported by all CDMI servers, and client certificates may be supported by CDMI clients. The server presents a server certificate to authenticate the server to the client; likewise, the client presents a client certificate to authenticate itself to the server. For public web sites offering secure communications via TLS, server certificate usage is quite common, but client certificates are rarely used, because the client is typically authenticated by other means. For example, an e-commerce site will authenticate a client by a credit card number, user name/password, etc., when a purchase is made. It is much more of a trust issue that the client (purchaser) be assured of the identity of the e-commerce site, and for this reason, server certificates are much more commonly encountered in practice.

These X.509 certificates use a digital signature to bind together a public key with an identity. These signatures will often be issued by a certification authority (CA) that is associated with an internal or external public key infrastructure (PKI); however, an alternate approach uses self-signed certificates (the certificate is digitally signed by the very same key-pair whose public part appears in the certificate data). The trust models associated with these two approaches are very different. In the case of PKI certificates, a hierarchy of trust and a trusted third party may be consulted in the certificate validation process, which enhances security at the expense of increased complexity. The self-signed certificates may be used to form a web of trust (trust decisions are in the hands of individual users/administrators), but is considered less secure, as there is no central authority for trust (e.g., no identity assurance or revocation). This reduction in overall security, which may still offer adequate protections for some environments, is accompanied by an easing of the overall complexity of implementation.

With PKI certificates, it is often necessary to traverse the hierarchy or chain of trust in search of a root of trust or trust anchor (a trusted CA). This trust anchor may be an internal CA, which has a certificate signed by a higher ranking CA, or it may be the end of a certificate chain as the highest ranking CA. This highest ranking CA is the ultimate attestation authority in a particular PKI scheme, and its certificate, known as a root certificate, may only be self-signed. Establishing a trust anchor at the root certificate level, especially for commercial CAs, may have undesirable side effects resulting from the implicit trust afforded all certificates issued by that commercial CA. Ideally the trust anchor should be established with the lowest ranking CA that is practical.

A.3.1.2.1 Certificate Validation

CDMI clients and servers shall perform basic path validation, extension path validation, and Certificate Revocation List (CRL) validation as specified in Section 6 of [RFC3280] for all presented certificates. These validations include, but are not limited to, the following:

- The certificate is a validly constructed certificate.
- The signature is correct for the certificate.
- The date of its use is within the validity period (i.e., it has not expired).
- The certificate has not been revoked (applies only to PKI certificates).
- The certificate chain is validly constructed (considering the peer certificate plus valid issuer certificates up to the maximum allowed chain depth (applies only to PKI certificates).

When CDMI clients and servers use CRLs, they shall use X.509 version 2 CRLs that conform to the CRL and CRL Extension Profile defined in Section 5 of [RFC3280]. (This requirement also only applies to PKI certificates.)

When PKI certificates and self-signed certificates are used together in a single management domain, it is important to recognize that the level of security is lowered to that afforded by self-signed certificates. Self-signed certificates by themselves only offer the keying materials to allow confidentiality and integrity in communications. The only identity assurances for self-signed certificates lie in the processes governing their acceptance as described below.

A.3.1.2.2 Certificate Formats

All interfaces for certificate configuration (import in particular) shall support the following certificate formats:

- DER encoded X.509. See [ITU-T509] for specification and technical corrigenda.

  International Telecommunications Union Telecommunication Standardization Sector (ITU-T), Recommendation X.509: Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks, May 2000. Specification and technical corrigenda may be obtained from: http://www.itu.int/ITU-T/publications/recs.html

- Base64 encoded X.509 (often called PEM). See Section 6.8 of [RFC2045].

  N. Freed and N. Borenstein, Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, IETF RFC 2045, November 1996, Section 6.8. Available at: http://www.ietf.org/rfc/rfc2045.txt

- PKCS#12. See [PKS12] for specification and technical corrigenda.

All certificate validation software shall support local certificate revocation lists, and at least one list per CA root certificate supported. Support is required for both DER encoded X.509 and Base64 encoded X.509 formats, but this support may be provided by using one format in the software and providing a tool to convert lists from the other format. OCSP and other means of immediate online verification of certificate validity are optional, as connectivity to the issuing Certificate Authority may not be assured.

A.3.1.2.3 Certificate Management

All certificates identifying CDMI entities and their associated private keys shall be replaceable. CDMI clients and servers shall either 1) have the ability to import an externally generated certificate and corresponding private key or 2) have the ability to generate and install a new self-signed certificate along with its corresponding private key.

When CDMI clients and servers use PKI certificates, the implementations shall include the ability to import, install/store, and remove the CA root certificates; support for multiple trusted issuing CAs shall be included. CA certificates are used to verify that a certificate has been signed by a key from an acceptable certification authority.

All certificate interfaces required above shall support access restrictions that permit access only by suitably privileged administrators. A suitably privileged security administrator shall be able to disable functionality for acceptance of unrecognized certificates described in Section A.3.1.2.1, "Certificate Validation" and Section A.3.1.2.2, "Certificate Formats".

The above requirements may be satisfied via appropriate use of the readily-available OpenSSL toolkit software (www.openssl.org). Support for PKCS#7 certificate format was deliberately omitted from the requirements. This format is primarily used for online interaction with certificate authorities; such functionality is not appropriate to require of all CDMI software, and tools are readily available to convert PKCS#7 certificates to or from other certificate formats.

A.3.1.2.4 Digital Certificate Guidance for TLS

To facilitate the use of certificates, CDMI implementations should include configurable mechanisms that allow for one of the following mutually exclusive operating modes to be in force at any time for end-entity certificates (i.e., not CA certificates):

- Unverifiable end-entity (self-signed) certificates are automatically installed as trust anchors when they are presented; such certificates shall be determined to not be CA root certificates prior to being installed as trust anchors and shall not serve as trust anchors to verify any other certificates. If a CA certificate is presented as an end-entity certificate in this mode, it shall be rejected. For CDMI clients, a variant of this option, which consults the user before taking action, should be implemented and used when possible.

  Note: The use of this operating mode should be limited to a learning or enrollment period during which communication is established with all other CDMI systems with which security communication is desired. Use of a timeout to force automatic exit from this mode is recommended.

- Unverifiable end-entity (self-signed) certificates may be manually imported and installed as trust anchors (in a fashion similar to manually importing and installing a CA root certificate), but they are not automatically added when initially encountered. Administrative privilege may be required to import and install an end-entity certificate as a trust anchor.

  This is considered the normal operating mode. All certificate acceptance policies for CDMI clients and servers shall be configurable. The configurable mechanisms determine how the CDMI implementation handles presented certificates. Under normal operating mode, CDMI servers should not accept certificates from unknown trust authorities (i.e., the CA root certificate has not been installed).

Interactive clients should provide a means to query the user about acceptance of a certificate from an unrecognized certificate authority (no corresponding CA root certificate installed in client), and accept responses allowing use of the certificate presented, or all certificates from the issuing CA. Servers should not support acceptance of unrecognized certificates; it is expected that a limited number of CAs will be acceptable for client certificates in any site that uses them.

Pre-configuring root certificates from widely used CAs is optional, but simplifies initial configuration of certificate-based security, as certificates from those CAs will be accepted. These CA root certificates may be exported from widely available web browsers.

Annex B
(informative)
Extending the Interface

The CDMI standard is designed to allow implementor-specific functions to be added. Vendor extensions are added by providing capabilities, where the capability name starts with an implementor name, instead of the "cdmi" value reserved for standardized capabilities.

The vendor extension for data object versioning, as described in this section, is written with the intention that it will be added as part of a future version of the CDMI standard, if so approved by the technical working group. Such a vendor extension should be written so that the implementor name may be replaced with "cdmi_" when added to the standard.

B.1 Data Object Versioning

Data object versioning provides a standard method for requesting versioning to be enabled, thereby retaining previous versions of data objects and providing a standardized method for accessing previous versions of data objects. How data object versioning is enabled and the interactions with other CDMI features, such as retention and serialization, is discussed in this section.

The primary goal of data object versioning is to allow CDMI clients to specify that some data objects shall have their previous contents retained when changed and to enable CDMI clients to access the previous contents of versioned data objects. The secondary goal is to ensure that a non-version-aware CDMI client is unaffected when versioning is enabled.

B.2 Terms

- Versioned object - A CDMI data object with versioning enabled.
- Versions container - A CDMI container that stores the versions of a versioned object.
- Object version - A CDMI data object that represents a specific version of a versioned object.

B.3 Versioning System Capabilities

The following system-wide capability is defined (as per Section 12.1.1, "Cloud Storage System-Wide Capabilities"):

Capability	Definition
com.netapp.versioning	If present and true, indicates that the cloud storage system supports versioning of data objects.

The following data system metadata capability is defined (as per Section 12.1.3, "Data System Metadata Capabilities"):

Capability	Definition
com.netapp.versions	If present, this capability defines the maximum number of versions that may be requested for a given data object.

B.4 Versioning Data System Metadataw

The following data system metadata is defined (as per Section 16.4, "Support for Data System Metadata"):

Metadata Name	Description
com.netapp.versions	JSON string containing the number of previous versions of the data object to be stored.

The following provided data system metadata item is defined (as per Section 16.6, "Support for Provided Data System Metadata"):

Metadata Actual Value	Description
com.netapp.versions_provided	JSON string containing the number of previous versions of the data object that are being stored.

B.5 Conditions for Versioning

Two conditions shall be met for previous versions to be retained for a given data object:

- Versioning shall be supported by the system, as indicated by the presence of the "com.netapp.versioning" system capability.
- Versioning shall be enabled for the data object, as indicated by the presence of the "com.netapp.versions" data system metadata in the data object or as inherited from a parent container.

If these conditions are met, the storage system shall store versions of the data object and provide access to these versions. Every change to a versioned object (data or metadata) shall result in a new version being retained, up to the specified or supported maximum number of versions.

Version objects created by the system are immutable, and versions shall not have nested versions.

B.6 Access to Versions

Previous versions of data objects are accessed in the same manner as snapshots are accessed for containers.

If versioning is enabled for a data object, the data object shall contain a JSON field named *versions*. This field shall contain a string that specifies a URI to a versions container. For example, if a data object with the URI “`/document.txt`” has versioning enabled, the contents of the *versions* field may be “`/documents.txt/versions/`”.

The children of the versions container are data objects that correspond to the versions of the data object. Each version data object has the name set to its object ID. This relationship between the versioned object, the versions container, and the object versions is shown in Figure 15.

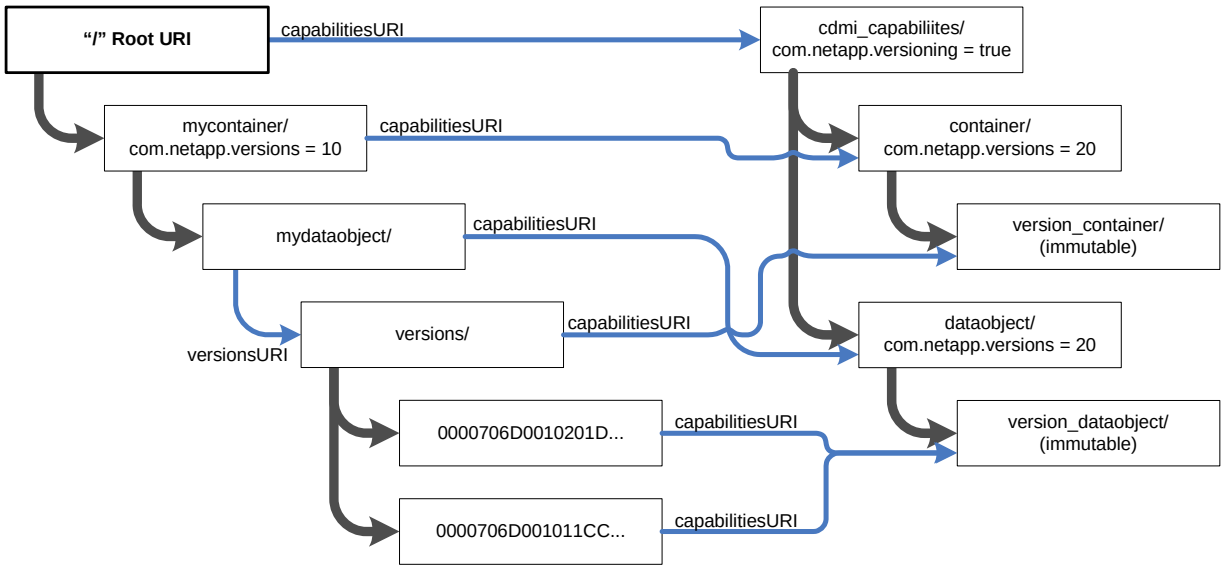


Figure 15 - Relationship Between Versioned Object, Versions Container, and Object Version

For example, if a data object "document.txt" has versioning enabled and two subsequent changes are made to the data object, the following children shall be present:

```
/document.txt/versions/
/document.txt/versions/0000706D0010201DA76AAE16879EB847
/document.txt/versions/0000706D001011CCC130940D93F8D97B
/document.txt/versions/0000706D00104E935A9B4F95242538FE
```

When present, the versions container shall always have a minimum of one version child data object, which represents the current version. As changes are made, additional versions are created, incorporating all changes.

In the above example, the 8FE object is the original version, the 97B object is the first revision, and the 847 object is the second and current revision. The list of the children objects of the versions container shall be returned and ordered from the newest to the oldest version.

If the number of versions supported by the cloud is limited, or if the value specified in the "com.netapp.versions" data system metadata has been reached, the oldest version is automatically removed by the system.

For example, if the "com.netapp.versions" data system metadata is set to the value "1", then the same operations as performed in the above example shall result in the following:

```
/document.txt/versions/
/document.txt/versions/0000706D0010201DA76AAE16879EB847
/document.txt/versions/0000706D001011CCC130940D93F8D97B
```

If the "com.netapp.versions" data system metadata is set to "0", then only the current (newest) version is accessible through the versions container.

Only if the "com.netapp.versions" data system metadata item is absent is the versions container absent.

Individual versions of an object are subject to the same ACL and retention characteristics of the original object; therefore, clients shall only be permitted to delete a version if they are permitted to delete the original data object.

B.6.1 The Current Version

If an object is modified and versioning is enabled for that object, a new version of the object shall be created. If the X-CDMI-PARTIAL header is used, the new version shall only be created when the object is complete.

The newest version object (the first child returned) shall have the same contents as the versioned object, except when partial updates are being performed. In this scenario, the contents of the newest version object shall differ from the object being versioned.

To illustrate this, assume a newly created versioned object with the value "`This`". The versioned object shall have the following versions:

```
/document.txt "This"
/document.txt/versions/0000706D001006A1D4534CF0DFDC1289"This"
```

This version is the newest version and has the value "`This`". Note that the object ID of the newest version is different from the object ID of the versioned object.

If the value "` is`" is appended to the object, the contents of the versions container is updated as a new version is added:

```
/document.txt"This is"
/document.txt/versions/0000706D001089F999BB0B087D30B340"This is"
/document.txt/versions/0000706D001006A1D4534CF0DFDC1289"This"
```

If the value " a" is appended to the object with the X-CDMI-PARTIAL flag, no new versions are added:

```
/document.txt     "This is a"
/document.txt/versions/0000706D001089F999BB0B087D30B340"This is"
/document.txt/versions/0000706D001006A1D4534CF0DFDC1289"This"
```

If the value "` test`" is appended to the object without the X-CDMI-PARTIAL flag, to complete the update, new versions are added:

```
/document.txt     "This is a test"
/document.txt/versions/0000706D0010CCB1FCDD87D9CA762065"This is a test"
/document.txt/versions/0000706D001089F999BB0B087D30B340"This is"
/document.txt/versions/0000706D001006A1D4534CF0DFDC1289"This"
```

B.6.2 Eventual Consistency

Given eventual consistency, at any time, the version list may be incomplete. Versions may appear between versions as consistency is restored within a system.

B.6.3 Audit of Versions

No new log messages or notifications are defined for versioning. Access of versions is logged and notified using standard messages. Attempts to delete retention-controlled versions are logged using standard messages.

If a limited number of versions are requested in the "com.netapp.versions" data system metadata, a modification of a versioned object that results in the deletion of a historical version results in an additional deletion audit message for the version deleted.

B.6.4 Serialization

Versions are serialized as children of the data object. If a serialized data object containing versions is attempted to be deserialized to a system that does not support versioning, the operation shall ignore the versions.

B.6.5 Exports

The versions container and object versions are not visible as part of the file hierarchy in file system exports. Implementors may use protocol-specific features, such as the "Previous Versions" feature in Windows/SMB to provide access to object versions.

B.6.6 Copy and Move

A copy operation for a versioned data object shall copy the object and all versions, unless the copy is to a system that does not support versions, in which case it shall only copy the data object. A copy operation with the source being a version of a data object shall copy that version to the specified destination as a data object.

A move operation for a versioned object shall move the object and all versions, unless the move is to a system that does not support versions, in which case it shall only move the data object. A move operation for a version shall fail, as the version is immutable and may not be deleted.

If a version of a data object is copied out to become a normal data object, it is no longer immutable and may have versions maintained for all subsequent modifications.