INTERNATIONAL STANDARD

ISO/IEC 17826

Second edition
2016-07-15

Information technology — Cloud Data Management Interface (CDMI)

Technologies de l'information — Interface de management des données du nuage informatique (CDMI)

Reference number
ISO/IEC 17826:2016(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

ISO/IEC 17826 was prepared by SNIA and was adopted, under the PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

This second edition cancels and replaces the first edition (ISO/IEC 17826:2012), which has been technically revised.

Cloud Data Management Interface (CDMI™)

Version 1.1.1

ABSTRACT: This CDMI International Standard is intended for application developers who are implementing or using cloud storage. It documents how to access cloud storage and to manage the data stored there.

This document has been released and approved by the SNIA. The SNIA believes that the ideas, methodologies, and technologies described in this document accurately represent the SNIA goals and are appropriate for widespread distribution. Suggestion for revision should be directed to http://www.snia.org/feedback/.

SNIA Technical Position

March 19, 2015

USAGE

The SNIA hereby grants permission for individuals to use this document for personal use only, and for corporations and other business entities to use this document for internal use only (including internal copying, distribution, and display) provided that:

1 Any text, diagram, chart, table or definition reproduced shall be reproduced in its entirety with no alteration, and,

2 Any document, printed or electronic, in which material from this document (or any portion hereof) is reproduced shall acknowledge the SNIA copyright on that material, and shall credit the SNIA for granting permission for its reuse.

Other than as explicitly provided above, you may not make any commercial use of this document, sell any excerpt or this entire document, or distribute this document to third parties. All rights not explicitly granted are expressly reserved to SNIA.

Permission to use this document for purposes other than those enumerated above may be requested by emailing tcmd@snia.org. Please include the identity of the requesting individual or company and a brief description of the purpose, nature, and scope of the requested use.

All code fragments, scripts, data tables, and sample code in this SNIA document are made available under the following license:

BSD 3-Clause Software License

Copyright (c) 2014, The Storage Networking Industry Association.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of The Storage Networking Industry Association (SNIA) nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

DISCLAIMER

The information contained in this publication is subject to change without notice. The SNIA makes no warranty of any kind with regard to this specification, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The SNIA shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this specification.

Suggestions for revisions should be directed to http://www.snia.org/feedback/.

Revision History

Version	Date	Originator	Comments
1.1.1	March 19, 2015	CDMI TWG	Released as a SNIA Technical Position.

Contents

Figures

Tables

Section I

CDMI Preamble

Introduction

This Cloud Data Management Interface (CDMI™) International Standard is intended for application developers who are implementing or using cloud storage. It documents how to access cloud storage and to manage the data stored there.

This document is organized as follows:

1 - Scope	Defines the scope of this document
2 - References	Lists the normative references for this document
3 - Terms	Provides terminology used in this document
4 - Conventions	Describes the conventions used in presenting the interfaces and the typographical conventions used in this document
5 - Overview of Cloud Storage	Provides a brief overview of cloud storage and details the philosophy behind this International Standard as a model for the operations
6 - Data Object Resource Operations using HTTP	Provides the normative standard of data object resource operations using HTTP
7 - Container Object Resource Operations using HTTP	Provides the normative standard of container object resource operations using HTTP
8 - Data Object Resource Operations using CDMI	Provides the normative standard of data object resource operations using CDMI
9 - Container Object Resource Operations using CDMI	Provides the normative standard of container object resource operations using CDMI
10 - Domain Object Resource Operations using CDMI	Provides the normative standard of domain object resource operations using CDMI
11 - Queue Object Resource Operations using CDMI	Provides the normative standard of queue object resource operations using CDMI
12 - Capability Object Resource Operations using CDMI	Provides the normative standard of capability object resource operations using CDMI
13 - Exported Protocols	Discusses how virtual machines in the cloud computing environment can 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
17 - Retention and Hold Management	Describes the optional retention management disciplines to be implemented into the system management functions
18 - Scope Specification	Describes the structure of the scope specification for JSON objects
19 - Results Specification	Provides a standardized mechanism to define subsets of CDMI object contents

20 - Logging	Describes CDMI functional logging for object functions, security events, data management events, and queues
21 - Notification Queues	Describes how CDMI clients may efficiently discover what changes have occurred to the system
22 - 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 - (informative) Extensions	Provides informative vendor extensions. Each extension is added to the standard when at least two vendors implement the extension.
Bibliography	Provides informative references that may contain additional useful information

1 Scope

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

2 Normative references

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

The provisions of the referenced specifications other than ISO/IEC, IEC, ISO, and ITU documents, as identified in this clause, are valid within the context of this International Standard. The reference to such a specification within this International Standard does not give it any further status within ISO/IEC. In particular, it does not give the referenced specifications the status of an International Standard.

ISO 3166, *Codes for the representation of names of countries and their subdivisions (Parts 1, 2 and 3)*

ISO 4217:2008, *Codes for the representation of currencies and funds*

ISO 8601:2004, *Data elements and interchange formats — Information interchange — Representation of dates and times*

ISO/IEC 9594-8:2008, *Information technology — Open Systems Interconnection — The Directory: Public-key and attribute certificate frameworks*

ISO/IEC 14776-414, *SCSI Architecture Model — 4 (SAM-4)*

ISO/IEC 17788:2014, *Information technology — Cloud computing — Overview and vocabulary*

ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards, 6th edition, 2011*

IEEE Std 1003.1, 2004*, POSIX ERE, The Open Group, Base Specifications Issue 6,* available at <http://www.unix.org/version3/ieee_std.html>

RFC 1867, *Form-based File Upload in HTML,* available at <http://www.ietf.org/rfc/rfc1867.txt>

RFC 2045, *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies,* available at <http://www.ietf.org/rfc/rfc2045.txt>

RFC 2046, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*, available at <http://www.ietf.org/rfc/rfc2046.txt>

RFC 2578, *Structure of Management Information Version 2 (SMIv2)*, available at <http://www.ietf.org/rfc/rfc2578.txt>

RFC 2616, *Hypertext Transfer Protocol — HTTP/1.1*, available at <http://www.ietf.org/rfc/rfc2616.txt>

RFC 2617, *HTTP Authentication: Basic and Digest Access Authentication*, available at <http://datatracker.ietf.org/doc/rfc2617/>

RFC 3530, *Network File System (NFS) Version 4 Protocol*, available at <http://www.ietf.org/rfc/rfc3530.txt>

RFC 3720, *Internet Small Computer Systems Interface (iSCSI)*, available at <http://www.ietf.org/rfc/rfc3720.txt>

RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*, available at <http://www.ietf.org/rfc/rfc3986.txt>

RFC 4627, *The Application/JSON Media Type for JavaScript Object Notation (JSON)*, available at <http://www.ietf.org/rfc/rfc4627.txt>

RFC 4648, *The Base16, Base32, and Base64 Data Encodings,* available at <http://www.ietf.org/rfc/rfc4648.txt>

RFC 4918, *HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV),* available at <http://www.ietf.org/rfc/rfc4918.txt>

RFC 6208, *Cloud Data Management Interface (CDMI) Media Types,* available at <http://www.ietf.org/rfc/rfc6208.txt>

RFC 6839, *Additional Media Type Structured Syntax Suffixes,* available at <http://www.ietf.org/rfc/rfc6839.txt>

SNIA TLS, *TLS Specification for Storage Systems, version 1.0,* available at <https://snia.org/tech_activities/standards/curr_standards/tls>

3 Terms, acronyms, and definitions

For the purposes of this document, the terms and definitions given in Rec. ITU-T Y.3500 | ISO/IEC 17788:2014 and the following apply.

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

3.2
API
Application Programming Interface

3.3
CDMI™
Cloud Data Management Interface

3.4
CDMI capability

object that describes what operations are supported for a given cloud or cloud objectNote 1 to entry: The mimetype for this object is application/cdmi-capability.

3.5
CDMI container
object that stores zero or more children objects and associated metadata

Note 1 to entry: The mimetype for this object is application/cdmi-container.

3.6
CDMI data object
object that stores an array of bytes (value) and associated metadata

Note 1 to entry: The mimetype for this object is application/cdmi-object.

3.7
CDMI domain
object that stores zero or more children domains and associated metadata describing object administrative ownership

Note 1 to entry: The mimetype for this object is application/cdmi-domain.

3.8
CDMI object
one of CDMI capabilities, CDMI container, CDMI data object, CDMI domain, or CDMI queue

3.9
CDMI queue
object that stores a first-in, first-out set of values and associated metadata

Note 1 to entry: The mimetype for this object is application/cdmi-queue.

3.10
CIFS
Common Internet File System

3.11
cloud storage
see 3.4 Data Storage as a Service

3.12
CRC
cyclic redundancy check

3.13
CRUD
create, retrieve, update, delete

3.14
Data Storage as a Service
DSaaS
delivery of appropriately configured virtual storage and related data services over a network, based on a request for a given service level

3.15
domain
shared user authorization database that contains users, groups, and their security policies and associated accounting information

Note 1 to entry: Each CDMI object belongs to a single domain, and each domain provides user mapping and accounting information.

3.16
eventual consistency
behavior of transactional systems that does not provide immediate consistency guarantees to provide enhanced system availability and tolerance to network partitioning

3.17
FC
Fibre Channel

3.18
FCoE
Fibre Channel over Ethernet

3.19
HTTP
HyperText Transfer Protocol

3.20
iSCSI
Internet Small Computer Systems Interface

3.21
JSON
JavaScript Object Notation

3.22
LDAP
Lightweight Directory Access Protocol

3.23
LUN
Logical Unit Number

3.24
metadata
data about other data (see ISO 14701:2012)

3.25
MIME
Multipurpose Internet Mail Extensions

3.26
NFS
Network File System

3.27
object
entity that has an object ID, has a unique URI, and contains state

Note 1 to entry: Types of CDMI objects include data objects, container objects, capability objects, domain objects, and queue objects.

3.28
object identifier
globally-unique value assigned at creation time to identify an object

3.29
OCCI
Open Cloud Computing Interface

3.30
POSIX
Portable Operating System Interface

3.31
Representational State Transfer
REST
specific set of principles for defining, addressing, and interacting with resources addressable by URIs

3.32
RPO
recovery point objective

3.33
RTO
recovery time objective

3.34
service level
performance targets for a service

3.35
SNMP
Simple Network Management Protocol

3.36
thin provisioning
technology that allocates the physical capacity of a volume or file system as applications write data, rather than pre-allocating all the physical capacity at the time of provisioning

3.37
Uniform Resource Identifier
URI
compact sequence of characters that identifies an abstract or physical resource

3.38
VIM
Vendor Interface Module

3.39
virtualization
presentation of resources as if they are physical, when in fact, they are decoupled from the underlying physical resources

3.40
WebDAV
Web Distributed Authoring and Versioning

3.41
XAM
eXtensible Access Method

4 Conventions

4.1 Interface format

Each interface description has nine components, as described in Table 1.

Table 1 — Interface format

Component	Description
Synopsis	The GET, PUT, POST, and DELETE semantics
Delayed completion	For long-running operations, a description of behavior when the operation does not immediately complete
Capabilities	A description of the supported operations
Request headers	The request headers, such as Accept, Authorization, Content-Length, Content-Type, X-CDMI-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 HTTP status codes
Example	An example of the operation

4.2 Typographical conventions

All code text and HTTP status codes are shown in a fixed-width font, as follows:

EXAMPLE 1

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

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

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

EXAMPLE 2 Requesting an optional field that is not present shall result in an HTTP status code of `404 Not Found`.

4.3 Request and response body requirements

In request and response body tables, the *Requirement* column contains one of the following three values.

- Mandatory. The value specified in this row shall be provided.
- Conditional. If the conditions specified in the *Description* cell of this row (to the left of the *Requirement*) are met, the value specified in this row shall be provided. Otherwise, it can be provided unless the *Description* specifically prohibits it, in which case it shall not be provided.
- Optional. The value specified in this row may be provided.

4.4 Key word requirements

In this International Standard, the key words in Table 2 shall be interpreted as described in ISO/IEC Directives, Part 2.

Table 2 — Key word requirements

Key words	Denotes	Description	Equivalent expressions (for exception cases only)
shall	requirement	An action that is unconditionally required. • Do not use *must* as an alternative to *shall*. • To express a direct instruction, for example, when referring to steps to be taken in a test method, use the imperative mood in English. EXAMPLE: *Switch on the recorder*.	• is to • is required to • it is required that • has to • only ... is permitted • it is necessary
shall not		An action that is unconditionally prohibited. Do not use *may not* instead of *shall not* to express a prohibition.	• is not allowed [permitted] [acceptable] [permissible] • is required to be not • is required that ... be not • is not to be
should	recommendation	An action that is recommended when choosing among several possibilities, or an action that is preferred but not necessarily required.	• it is recommended that • ought to
should not		An action or certain possibility or course of action that is deprecated but not prohibited.	• it is not recommended that • ought not to
may	permission	An action that indicates what is allowed within the limits of the document. Do not use *possible* or *can* in this context. *May* signifies permission expressed by the document, whereas *can* refers to the ability of a user of the document or to a possibility open to him or her.	• is permitted • is allowed • is permissible
need not		An action that indicates what is not required within the limits of the document. Do not use *impossible* in this context.	• it is not required that • no ... is required

Table 2 — Key word requirements

Key words	Denotes	Description	Equivalent expressions (for exception cases only)
can	possibility and capability	An action used for statements that indicate possibility and capability, whether material, causal, or physical. Do not use *may* instead of *can* in this context.*Can* refers to the ability of a user of the document or to a possibility open to him or her, whereas *may* signifies permission expressed by the document.	• be able to • it is possible to
cannot		An action used for statements that indicate impossibility and incapability, whether material, causal, or physical.	• be unable to • it is impossible to

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 (i.e., data paths) and are managed by management interfaces (i.e., control paths). This International Standard explores the various types of interfaces that are part of cloud services today and shows how they are related. This International Standard defines a model for the interfaces that can be mapped to the various cloud services and a model that forms the basis for cloud storage interfaces into the future.

Another important concept in this International 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 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 use cases.

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 icon for a network. The cloud represents any-to-any network connectivity in an abstract way. In this abstraction, the network connectivity in the cloud is represented without concern for how it is made to happen.

The cloud abstraction of complexity produces a simple base on which other features can be built. The general cloud model extends this base by adding a pool of resources. An important part of the cloud model is the concept of a pool of resources that is drawn from, on demand, in small increments. A 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 is used for this is *Data storage as a Service* (*DaaS*).

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 cloud services and implementations are possible. The only type of storage that is excluded from this definition is that which is delivered in fixed-capacity increments instead of that which is based on demand.

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

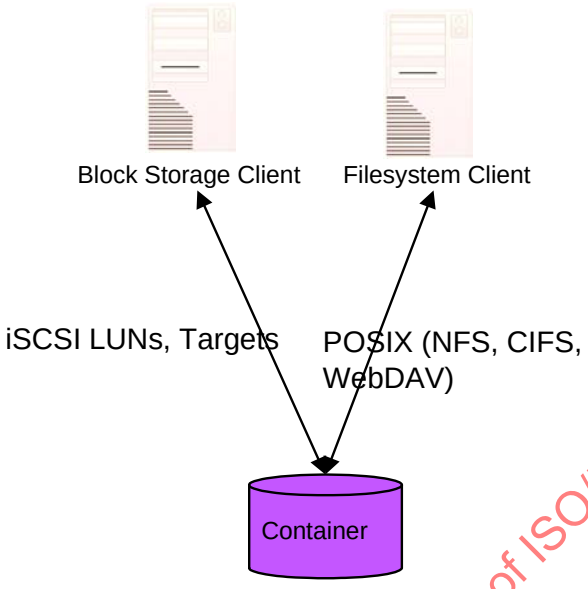

Figure 1 — Existing data storage interface standards

The difference between purchasing a dedicated appliance or purchasing cloud storage is not the functional interface, but the fact that the storage is delivered on demand. Customers pay for either what they actually use or what they have allocated for use. For 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, can be used to further reduce the actual space consumed.

Managing this storage is typically done out of band for these standard data storage interfaces, either through an API, or more commonly, through an administrative browser-based user interface. This out-of-band interface can be used to invoke other data services as well (e.g., snapshots or cloning).

In this model, the underlying storage space that has been exposed by the out-of-band interfaces is abstracted and exposed 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.

Each data object is created, retrieved, updated, and deleted 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 from being hierarchical, although any given implementation might support only a single level (see Figure 2).

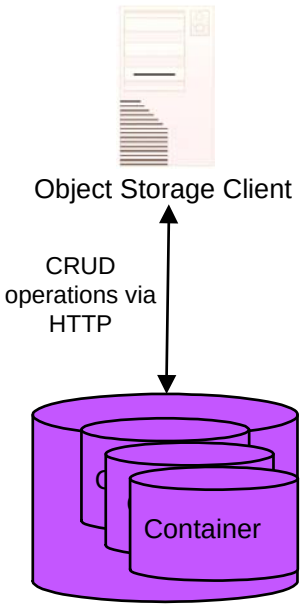

Figure 2 — Storage interfaces for object storage client data

5.4 Data management for cloud storage

Many of the initial implementations of cloud storage focused on a kind of best effort quality of storage service and ignored most other types of data services. To address the needs of enterprise applications with cloud storage, however, there is an increasing need to offer better quality of service and to deploy additional data services.

Cloud storage can lose its abstraction and simplicity benefits if new data services that require complex management are added. Cloud storage customers are likely to resist new demands on their time (e.g., setting up backup schedules through dedicated interfaces, deploying data services individually for stored objects).

By supporting metadata in a cloud storage interface 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 model can be maintained while still addressing the requirements of enterprise applications and their data.

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

Storage system metadata is produced/interpreted by the cloud service and basic storage functions (e.g., modification and access statistics, access control). For more information on support for storage system metadata, see 16.3.

Data system metadata is interpreted by the cloud service as data requirements that control the operation of underlying data services for that data. Depending on the level of granularity supported by the cloud, data system metadata may apply to an aggregation of data objects in a container or to individual data objects. For more information on support for data system metadata, see 16.4.

5.5 Data and container management

There is no reason that managing data and managing containers should involve different interfaces. Therefore, the use of metadata is extended from applying to individual objects to applying to containers of objects as well. Thus, any data placed into a container 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 metadata. After an object is created, the data system metadata can be overridden at the container or individual object level, as desired.

Even if the provided interface does not support setting metadata on individual objects, metadata can still be applied to the containers. In such a case, the interface does not provide a mechanism to override metadata that an individual object inherits from its parent container. For file-based interfaces that support extended attributes (e.g., CIFS, NFSv4), these extended attributes can be used to specify the data system metadata to override that specified for the container.

5.6 Reference model for cloud storage interfaces

The cloud storage reference model is shown in Figure 3.

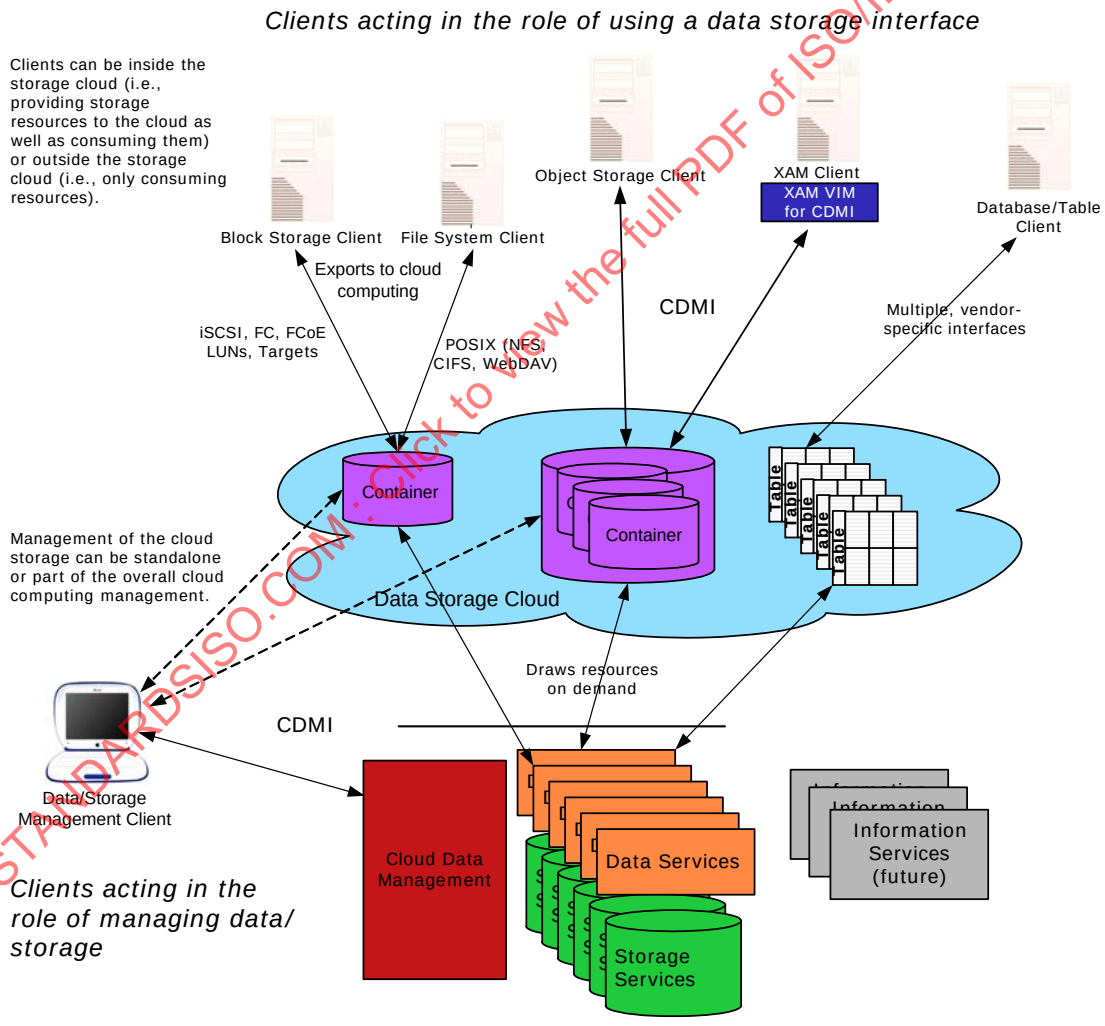

Figure 3 — Cloud storage reference model

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 storage capacity is drawn from a pool of storage capacity provided by storage services.

The data services are applied to individual objects, as determined by the data system metadata. Metadata specifies the data requirements on the basis of individual objects or for groups of objects (containers).

5.7 Cloud Data Management Interface

The Cloud Data Management Interface (CDMI™) shown in Figure 3 can be used to create, retrieve, update, and delete objects in a cloud. The features of the CDMI include functions that:

- allow clients to discover the capabilities available by the cloud provider,
- manage containers and the data that is placed in them, and
- allow metadata to be associated with containers and the objects they contain.

This International Standard divides operations into two types: those that use a CDMI content type in the HTTP body and those that do not. While much of the same data is available via both types, providing both allows for CDMI-aware clients and non-CDMI-aware clients to interact with a CDMI provider.

CDMI can 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 can understand what services the cloud provides.

Conformant cloud services can support a subset of the CDMI, as long as they expose the limitations in the capabilities reported via the interface.

This International Standard uses RESTful principles in the interface design where possible (see REST).

CDMI defines both a means to manage the data as well as a means to store and retrieve the data. The means by which the storage and retrieval of data is achieved is termed a *data path*. The means by which the data is managed is termed a *control path*. CDMI specifies both a data path and control path interface.

CDMI does not need to be used as the only data path and is able to manage cloud storage properties for any data path interface (e.g., standardized or vendor specific).

Container metadata is used to configure the data requirements of the storage provided through the exported protocol (e.g., block protocol or file protocol) that the container exposes. When an implementation is based on an underlying file system to store data for a block protocol (e.g., iSCSI), the CDMI container provides a useful abstraction for representing the data system metadata for the data and the structures that govern the exported protocols.

A cloud service may also support domains that allow administrative ownership to be associated with stored objects. Domains allow this International Standard to (among other things):

- determine how user credentials are mapped to principals used in an Access Control List (ACL),
- allow granting of special cloud-related privileges, and
- allow delegation to external user authorization systems (e.g., 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, capabilities allow a client to discover the capabilities of a CDMI implementation. Requirements throughout this International Standard shall be understood in the context of CDMI capabilities. Mandatory requirements on functionality that is conditioned on a CDMI capability shall not be interpreted to require implementation of that capability, but rather shall be interpreted to apply only to implementations that support the functionality required by that capability.

For example, in 5.10, this International Standard states, "Every cloud storage system shall allow object ID-based access to stored objects." This requirement shall be understood in the context that access by object ID is predicated on the presence of the cdmi_object_access_by_ID capability.

5.8 Object model for CDMI

The model for CDMI is shown in Figure 4.

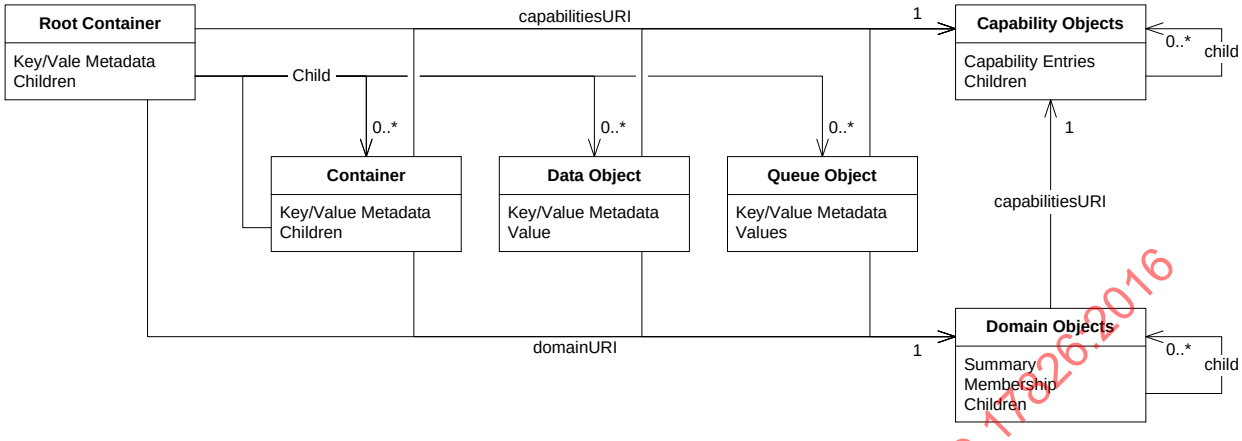

Figure 4 — CDMI object model

The five types of resources defined are shown in Table 3. The content type in any given operation is specific to each type of resource.

Table 3 — Types of resources in the model

Resource type	Description	Reference
Data objects	Data objects are used to store values and provide functionality similar to files in a file system.	See Clause 8.
Container objects	Container objects have zero or more children, but do not store values. They provide functionality similar to directories in a file system.	See Clause 9.
Domain objects	Domain objects represent administrative groupings for user authentication and accounting purposes.	See Clause 10.
Queue objects	Queue objects store zero or move values and are accessed in a first-in-first-out manner.	See Clause 11.
Capability objects	Capability objects describe the functionality implemented by a CDMI server and are used by a client to discover supported functionality.	See Clause 12.

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 (see 5.5). Using the CDMI object model (see Figure 4), the client can send a PUT via CDMI (see 5.6) to the new container URI and create a new container with the specified name. Container metadata are optional and are expressed as a series of name-value pairs. After a container is created, a client can send a PUT to create a data object within the newly created container. A subsequent GET will fetch the data object, including the value field.

Queue objects are also defined (see Figure 4) and provide in-order, first-in-first-out access to enqueued objects. More information on queues can be found in Clause 11.

CDMI defines two namespaces that can be used to access stored objects, a flat object ID namespace and a hierarchical path-based namespace. Support for objects accessed by object ID is indicated by the system-wide capability cdmi_object_access_by_ID, and support for objects accessed by hierarchical path is indicated by the container capability cdmi_create_dataobject found on the root container (and any subcontainers).

Objects are created by ID by performing an HTTP POST against a special URI, designated as /cdmi_objectid/ (see 9.6). Subsequent to creation, objects are modified by performing PUTs using the

object ID assigned by the CDMI server, using the /cdmi_objectid/ URI (see 8.4 "Update a data object using CDMI"). The same URI is used to retrieve and delete objects by ID.

Objects are created by name by performing an HTTP PUT to the desired path URI (see 8.2 "Create a data object using CDMI"). Subsequent to creation, objects are modified by performing PUTs using the object path specified by the client (see 8.4). The same URI is used to retrieve and delete objects by path.

CDMI defines mechanisms so that objects having only an object ID can be assigned a path location within the hierarchical namespace, and so that objects having both an object ID and path can have their path dropped, such that the object only has an object ID. This function is accomplished by using a "move" modifier to a PUT or POST operation, as shown in Figure 5.

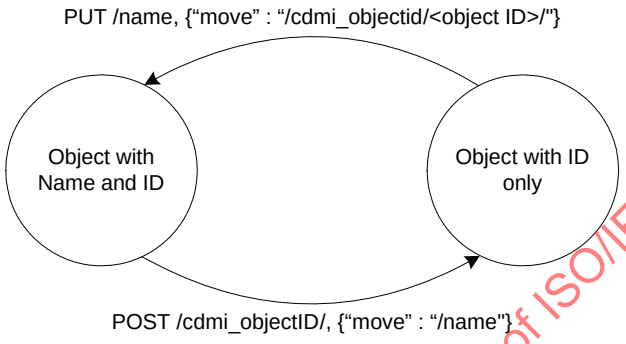

Figure 5 — Object transitions between named and ID-only

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 (e.g., Content-Length, Content-Type, etc.). HTTP metadata is not specifically related to this International Standard but needs to be discussed to explain how CDMI uses the HTTP standard.

CDMI data system metadata, user metadata, and storage system metadata is defined in the form of name-value pairs. Vendor-defined data system metadata and storage system metadata names shall begin with the reverse domain name of the vendor.

Data system metadata is metadata that is specified by a CDMI client and is a component of objects. Data system metadata abstractly specifies the data requirements associated with data services that are deployed in the cloud storage system.

User metadata consists of client-defined JSON strings, arrays, and objects that are stored in the metadata field. The namespace used for user metadata names is self-administered (e.g., using the reverse domain name), and user metadata names shall not begin with the prefix "cdmi_."

Storage system metadata is metadata that is generated by the storage services in the system (e.g., creation time, size) to provide useful information to a CDMI client.

The matrix of the creation and consumption of storage system metadata is shown in Table 4.

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 shall have a globally unique object identifier (ID) assigned at creation time. The CDMI object ID is a string with requirements for how it is generated and how it obtains its uniqueness. Each cloud service that implements CDMI shall generate these identifiers such that the probability of conflicting with identifiers generated by other cloud services and the probability of generating an identifier that has already been used is effectively zero.

Every cloud storage system shall allow object ID-based access to stored objects by allowing the object's ID to be appended to the root container URI. If the data object "MyDataObject.txt", located in the root container, has an object ID of "00006FFD001001CCE3B2B4F602032653", the following pair of URIs access the same data object:

http://cloud.example.com/root/MyDataObject.txt

http://cloud.example.com/root/cdmi_objectid/00006FFD001001CCE3B2B4F602032653

If containers are supported, they shall also be accessible by object ID. If the container "MyContainer", located in the root container, has an object ID of "00006FFD0010AA33D8CEF9711E0835CA", the following pairs of URIs access the same object:

http://cloud.example.com/root/MyContainer/

http://cloud.example.com/root/cdmi_objectid/00006FFD0010AA33D8CEF9711E0835CA/

http://cloud.example.com/root/MyContainer/MyDataObject.txt

http://cloud.example.com/root/cdmi_objectid/00006FFD0010AA33D8CEF9711E0835CA/MyDataObject.txt

5.11 CDMI object ID format

The cloud service 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 6. The native format of an object ID is a variable-length byte sequence and shall be a maximum length of 40 bytes. A client should treat object IDs as opaque byte strings.

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

Figure 6 — Object ID format

The fields shown in Figure 6 are defined as follows.

- The reserved bytes shall be set to zero.
- The Enterprise Number field shall be the SNMP enterprise number of the offering organization that developed the system that created the object ID, in network byte order. See RFC 2578 and http://www.iana.org/assignments/enterprise-numbers. 0 is a reserved value.
- The byte at offset 5 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 (see ISO 14701:2012) 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 fields:

— Name : "CRC-16",
— Width : 16,
— Poly : 0x8005,
— Init : 0x0000,
— RefIn : True,
— RefOut : True,
— XorOut : 0x0000, and
— Check : 0xBB3D.

This function defines a 16-bit CRC with polynomial 0x8005, reflected input, and reflected output. This CRC-16 is specified in ISO 14701:2012.

- 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 URIs and JSON strings, the object ID textual representation shall be encoded using Base16 encoding rules described in RFC 4648 and shall be case insensitive.

5.12 Security

5.12.1 Security objectives

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 providing a mechanism that:

- assures that the communications between a CDMI client and server cannot be read or modified by a third party;
- allows CDMI clients and servers to assure their identity;
- allows control of the actions a CDMI client is permitted to perform on a CDMI server;
- allows records to be generated for actions performed by a CDMI client on a CDMI server;
- protects data at rest;
- eliminates data in a controlled manner; and
- discovers the security capabilities of of a particular implementation.

Security measures within CDMI are 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 capabilities.

With the exception of both the transport security and the security capabilities, which are mandatory to implement, the security measures can vary significantly from implementation to implementation.

When security is a concern, the CDMI client should begin with a series of security capability lookups (see 12.1.1) to determine the exact nature of the security features that are available. Based on the values of these capabilities, a risk-based decision should be made as to whether the CDMI server should be used.

This is particularly true when the data to be stored in the cloud storage is sensitive or regulated in a way that requires stored data to be protected (e.g., encrypted) or handled in a particular manner (e.g., full accountability and traceability of management and access).

5.12.2 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. As CDMI is built on top of HTTP, HTTP over Transport Layer Security (TLS) (i.e., HTTPS) is the mechanism that is used to secure the communications between CDMI clients and servers.

To ensure both security and interoperability, all CDMI implementations shall:

- implement the TLS protocol as described in the latest version of the "SNIA TLS Specification for Storage Systems," with a six-month transition period for implementations. The TLS specification is updated when new vulnerabilities are found, and CDMI implementations shall support the latest specification within six months of its publication announcement;
- support both HTTP over TLS and HTTP without TLS; and
- allow HTTP without TLS to be disabled.

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 using server-side certificates, which the client trusts, as the basis for unidirectional entity authentication. It is possible that mutual authentication involving both client-side and server-side certificates are required.

5.12.3 Client authentication

A CDMI client shall comply with all security requirements for HTTP that apply to clients.

CDMI clients can be responsible for initiating user authentication for each CDMI operation that is performed. The CDMI server functions as the authenticator and receives and validates authentication credentials from the client.

RFC 2616 and RFC 2617 define requirements for HTTP authentication, which generally starts with an HTTP client request. If the client request does not include an "Authorization" header and authentication is required, the server responds with an HTTP status code of `401 Unauthorized` and a WWW-Authenticate response header. The HTTP client shall then respond with the appropriate Authorization header in a subsequent request.

The format of the WWW-Authenticate and Authorization headers varies depending on the type of authentication required.

- HTTP 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 TLS.
- HTTP digest authentication sends a secure digest of the user name and password (and other information such as a nonce value), and can be used on an insecure network without TLS.
- HTTP status codes of `401 Unauthorized` should not include a choice of authentication.
- HTTP basic authentication or HTTP digest authentication should be implemented.
- Authentication credentials used with one type of HTTP authentication (i.e., basic or digest) should never be subsequently used with the other type of HTTP authentication.

Once a user is authenticated, the provided principal name shall be mapped by the CDMI domain to a domain user (or used directly as the ACE "who" if domains are not supported). This mapping is then used to determine authorization.

A CDMI server typically relies on an authentication service (local or external) to validate client credentials. Differing authentication schemes can be supported, including host-based authentication, Kerberos, PKI, or other; the authentication service is beyond the scope of this International Standard.

5.12.4 Use of TLS and HTTP

Recommendations for using HTTP and TLS are as follows:

- A client connecting to a CMDI server using TLS should use TCP port 443, and a client connecting without TLS should use TCP port 80.
- A client that fails to connect to a CDMI server on port 443 should retry without TLS on TCP port 80 if their security policy allows it.
- Servers may respond to HTTP requests on port 80 with an HTTP REDIRECT to the appropriate TLS URI (using port 443). Clients should honor such redirects in this situation.

5.12.5 Further information

For further information pertaining to storage security techniques, see ISO 14701:2012.

5.13 Required HTTP support

5.13.1 RFC 2616 support requirements

A conformant implementation of CDMI shall also be a conformant implementation of RFC2616 (see RFC 2616) (i.e., HTTP 1.1). The subclauses below list the sections of RFC 2616 that shall be supported; however, this list is not comprehensive.

5.13.2 Content-type negotiation

For CDMI operations, media types for CDMI objects are used as defined in RFC 6208. All CDMI representations follow the rules established for "application/json" as defined in RFC 4627. The use of the CDMI media types with the "+json" suffix shall be supported as defined in RFC 6839.

A client can optionally supply an HTTP Accept header, as per section 14.1 of RFC 2616. If a client is restricting the response to a specific CDMI media type, the corresponding media type shall be specified in the Accept header. Otherwise, the Accept header can contain "*/*" or a list of media types, or it can be omitted.

If a request body is present, the client shall include a Content-Type header, as per section 14.17 of RFC 2616. If the client does not provide a Content-Type header when required or provides a media type in the Content-Type header that does not match with the existing resource media type, the server shall return an HTTP status code of `400 Bad Request`.

If a response body is present, the server shall provide a Content-Type header.

This International Standard can further qualify content negotiation (e.g., in 9.3, the absence of a Content-Type header has a specific meaning).

5.13.3 Range support

The server shall support HTTP Range headers and partial content responses (see Section 14.16 of RFC 2616).

The values of the childrange, valuerange and queuerange fields are formatted based on the HTTP byte-range-resp-spec, as defined in clause 14.16 of RFC 2616.

5.13.4 URI escaping

Percent escaping of reserved characters specified in RFC 3986 shall be applied to all text strings used in HTTP request URIs and HTTP header URIs. This includes user-supplied field names, metadata names, data object names, container object names, queue object names, and domain object names when used in HTTP request URIs and HTTP header URIs.

Field names and values shall not be escaped when stored and when sent in request body and response bodies.

EXAMPLE A client retrieving a metadata item named "@user" from a container object with the name of "@MyContainer" would perform the following request:

```
GET /%40MyContainer/?objectName;metadata:%40user HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-container
X-CDMI-Specification-Version: 1.1
```

The response shall be:

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

{
    "objectName": "@MyContainer/",
    "metadata": {
        "@user": "test",
     ...
    }
}
```

5.13.5 Use of URIs

The format and syntax of URIs are defined by RFC 3986.

Every CDMI client shall maintain one or more root URIs that each correspond to a root container on the CDMI server. Since all URIs to CDMI containers end in a trailing slash, all root URIs will end in a trailing slash.

All URIs in this International Standard are relative to the root URI unless otherwise noted. As a consequence, the algorithm used for calculating the resolved URI is as described in Section 5.2 of RFC 3986.

Table 5 shows how relative URIs are resolved against root URIs.

Table 5 — Relative URIs resolved against root URIs

Root URI	+ Relative URI	=> Resolved URI
http://cloud.example.com/	cdmi_object/testObject	http://cloud.example.com/cdmi_object/testObject
http://cloud.example.com/	/cdmi_object/testObject	http://cloud.example.com/cdmi_object/testObject
http://cloud.example.com/p1/	cdmi_object/testObject	http://cloud.example.com/p1/cdmi_object/testObject
http://cloud.example.com/p1/	/cdmi_object/testObject	http://cloud.example.com/cdmi_object/testObject
http://cloud.example.com/p1/p2/	cdmi_object/testObject	http://cloud.example.com/p1/p2/cdmi_object/testObject
http://cloud.example.com/p1/p2/	/cdmi_object/testObject	http://cloud.example.com/cdmi_object/testObject

This International Standard places no restrictions on root and relative URIs. All of the examples in this specification use a root URI of http://cloud.example.com/ and return absolute path references as shown in the second line of Table 5.

- If the root URI is "/", the container located at the root URI shall omit the parentID field and shall return an empty string ("") for the value of the parentURI field.
- If the root URI is not "/" and the parent is a CDMI container, the container located at the root URI shall populate parentID field with the CDMI object ID of the CDMI container corresponding to the parent path component, and populate the parentURI field with the URI of the parent path component.
- If the root URI is not "/" and the parent is not a CDMI container, the container located at the root URI shall omit the parentID field, and populate the parentURI field with the URI of the parent path component.
- If the root URI is not "/" and the parent is not accessible, the server can omit the parentID field and return an empty string ("") for the value of the parentURI field.

5.13.6 Reserved characters

The name of CDMI data objects, container objects, queue objects, domain objects and capability objects shall not contain the "/" or "?" characters, as these characters are reserved for delimiters.

5.14 Time representations

Unless otherwise specified, all date/time values are in the ISO 8601:2004 extended representation (YYYY-MM-DDThh:mm:ss.ssssssZ). The full precision shall be specified, the sub-second separator shall be a ".", the Z UTC zone indicator shall be included, and all timestamps shall be in UTC time zone. The YYYY-MM-DDT24:00:00.000000Z hour shall not be used, and instead, it shall be represented as YYYY-MM-DDT00:00:00.000000Z.

Unless otherwise specified, all date/time intervals are in the ISO 8601:2004 start date/end date representation (YYYY-MM-DDThh:mm:ss.ssssssZ/YYYY-MM-DDThh:mm:ss.ssssssZ). The end date shall be equal to or later than the start date. The full precision shall be specified, the sub-second separator shall be a ".", the Z UTC zone indicator shall be included, and all timestamps shall be in UTC time zone. The YYYY-MM-DDT24:00:00.000000Z hour shall not be used, and instead, it shall be represented as YYYY-MM-DDT00:00:00.000000Z.

5.15 Backwards compatibility

5.15.1 Value transfer encoding

CDMI version 1.0.1 introduces the concept of value transfer encoding to enable the storage and retrieval of arbitrary binary data via CDMI content-type operations. Data objects created by CDMI 1.0 clients through CDMI content-type operations shall have a value transfer encoding of "utf-8", and data objects created through non-CDMI content-type operations shall have a value transfer encoding of "base64".

Data objects with a value transfer encoding of base 64 shall not have their value field accessible to CDMI 1.0 clients through CDMI content-type operations. Attempts to read the value of these objects shall return an empty value field ("") to these clients. CDMI 1.0 clients can detect this condition when the cdmi_size metadata is not 0 and the value field is empty.

5.15.2 Container export capabilities

CDMI version 1.0.2 normalizes the names of capabilities used by a client to discover if a container can be exported via various protocols and deprecates the following container export capability names:

- cdmi_cifs_export,
- cdmi_nfs_export,
- cdmi_iscsi_export, and
- cdmi_occi_export.

5.16 Object references

Object references are URIs within the cloud storage namespace that redirect to another URI within the same or another cloud storage namespace. References are similar to soft links in a file system. 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 in container children listings by the presence of 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 an HTTP status code of `302 Found`, with the HTTP Location header containing the absolute redirect destination URI that was specified at the time the reference was created. The reference's destination URI shall not be changed after a reference has been created.

To continue, when CDMI clients receive an HTTP status code of `302 Found`, they should retry the operation using the URI contained within the Location header.

A delete operation on a reference URI shall delete the reference. References cannot be updated. To update the destination of a redirect, the client shall first delete the reference and then create a new reference to the desired destination.

EXAMPLE 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: 1.1
```

The following shows the response.

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

References by object ID shall always redirect to a URI that ends with the same object ID as the request URI.

EXAMPLE 2 GET to an object ID URI, where the URI is a reference:

```
GET /cdmi_objectid/00006FFD0010AA33D8CEF9711E0835CA HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
X-CDMI-Specification-Version: 1.1
```

The following shows the response.

```
HTTP/1.1 302 Found
Location: http://archive.example.com/cdmi_objectid/00006FFD0010AA33D8CEF9711E0835CA
```

EXAMPLE 3 PUT to create a reference:

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

```
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.1

{
    "reference": "http://cloud.example.com/MyContainer/MyOtherDataObject.txt"
}
```

The following shows the response.

```
HTTP/1.1 201 Created
```

EXAMPLE 4 POST to create a reference:

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

{
    "reference": "http://cloud.example.com/MyContainer/MyOtherDataObject.txt"
}
```

The following shows the response.

```
HTTP/1.1 201 Created
Location: http://cloud.example.com/cdmi_objectid/00007ED90010DF417BAD70A0C7F5CDDA
```

EXAMPLE 5 DELETE to delete a reference:

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

The following shows the response.

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

Section II

Basic Cloud Storage

6 Data object resource operations using HTTP

6.1 Overview

Data objects are the fundamental storage components within CDMI™ and are analogous to files in a file system.

As CDMI builds on top of, and is compatible with, the HTTP standard (RFC 2616), this allows unmodified HTTP clients to communicate with a CDMI server. This also allows CDMI operations to coexist with other HTTP-based storage protocols, such as WebDAV, S3, and OpenStack Swift.

A CDMI server differentiates between HTTP and CDMI operations using the standard Content-Type and Accept headers. When CDMI MIME types defined in RFC 6208 are used in these headers, this indicates that CDMI behaviors, as described in Clause 8, are used in addition to the standard HTTP behaviors. When CDMI MIME types are used, the X-CDMI-Specification-Version header is included to indicate which version of CDMI is being requested by the client and provided by the server.

In CDMI 1.0.2, basic HTTP operations were described as "Non-CDMI" operations to distinguish them from operations using CDMI MIME types.

A CDMI implementation that supports data objects shall include support for basic data object HTTP operations corresponding with the CDMI capabilities that are published by the implementation. Capabilities allow a client to discover which operations (such as create, update, delete, etc.) are supported and are described in Clause 12.

6.2 Create a data object using HTTP

6.2.1 Synopsis

The following HTTP PUT creates a new data object at the specified URI:

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

Where:

- <root URI> is the path to the CDMI cloud.
- <ContainerName> is zero or more intermediate containers that already exist, with one slash (i.e., "/") between each pair of container names.
- <DataObjectName> is the name specified for the data object to be created.

After it is created, the data object shall also be accessible at <root URI>/cdmi_objectid/<objectID>.

6.2.2 Capabilities

The following capabilities describe the supported operations that can 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.
- Support for the ability to create the value of a new data object in specified byte ranges is indicated by the presence of the "cdmi_create_value_range" capability in the parent container.

6.2.3 Request headers

The HTTP request headers for creating a data object using HTTP are shown in Table 6.

Table 6 — Request headers - Create a data object using HTTP

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 as the mimetype field of the data object. • If the content type includes the charset parameter as defined in RFC 2046 of "utf-8" (e.g., ";charset=utf-8"), the valuetransferencoding field of the data object shall be set to "utf-8". Otherwise, the valuetransferencoding field of the data object shall be set to "base64". • If not specified, the mimetype field shall be set to "application/octet-stream".	Optional
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". X-CDMI-Partial works across CDMI and non-CDMI operations.	Optional
Content-Range	Header string	A valid ranges-specifier (see RFC 2616 Section 14.35.1	Optional

6.2.4 Request message body

The request message body contains the data to be stored in the value of the data object.

6.2.5 Response headers

No response headers are specified.

6.2.6 Response message body

No response message body fields are specified.

6.2.7 Response status

The HTTP status codes that occur when creating a data object using HTTP are described in Table 7.

Table 7 — HTTP status codes - Create a data object using HTTP

HTTP status	Description
`201 Created`	The new data object was created.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.
`409 Conflict`	The operation conflicts with a non-CDMI access protocol lock or has caused a state transition error on the server.

6.2.8 Example

EXAMPLE 1 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;charset=utf-8
Content-Length: 37

This is the Value of this Data Object
```

The following shows the response.

```
HTTP/1.1 201 Created
```

6.3 Read a data object using HTTP

6.3.1 Synopsis

The following HTTP GET reads from an existing data object at the specified URI:

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

Where:

- <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.

The object shall also be accessible at <root URI>/cdmi_objectid/<objectID>.

6.3.2 Capabilities

The following capabilities describe the supported operations that can be performed when reading an existing data 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.

6.3.3 Request header

The HTTP request header for reading a data object using HTTP is shown in Table 8.

Table 8 — Request header - Read a data object using HTTP

Header	Type	Description	Requirement
Range	Header string	A valid ranges-specifier (see RFC 2616 Section 14.35.1)	Optional

6.3.4 Request message body

A request body shall not be provided.

6.3.5 Response headers

The HTTP response headers for reading a data object using HTTP are shown in Table 9.

Table 9 — Response headers - Read a data object using HTTP

Header	Type	Description	Requirement
Content-Type	Header string	The content type returned shall be the mimetype field in the data object.	Mandatory
Location	Header string	The server shall respond with the URI that the reference redirects to if the object is a reference.	Conditional

6.3.6 Response message body

When reading a data object using HTTP, the following applies:

- The response message body shall be the contents of the data object's value field.
- When reading a value, zeros shall be returned for any gaps resulting from non-contiguous writes.

6.3.7 Response status

The HTTP status codes that occur when reading a data object using HTTP are described in Table 10.

Table 10 — HTTP status codes - Read a data object using HTTP

HTTP status	Description
`200 OK`	The data object content was returned in the response.
`206 Partial Content`	A requested range of the data object content was returned in the response.
`302 Found`	The resource is a reference to another resource.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI, or a requested field within the resource was not found.

6.3.8 Examples

EXAMPLE 1 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 following shows the response.

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

This is the Value of this Data Object
```

EXAMPLE 2 GET to the data object URI to read the first 11 bytes of the value of the data object:

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

The following shows the response.

```
HTTP/1.1 206 Partial Content
Content-Type: text/plain
Content-Range: bytes 0-10/37
Content-Length: 11

This is the
```

6.4 Update a data object using HTTP

6.4.1 Synopsis

The following HTTP PUT updates an existing data object at the specified URI:

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

Where:

- <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 shall also be accessible at <root URI>/cdmi_objectid/<objectID>. An update shall not result in a change to the object ID.

6.4.2 Capabilities

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

- Support for the ability to modify the value of an existing data object or MIME type 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.

6.4.3 Request headers

The HTTP request headers for updating a data object using HTTP are shown in Table 11.

Table 11 — Request headers - Update a data object using HTTP

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 data object.	Mandatory
Content-Range	Header string	A valid ranges-specifier (see RFC 2616 Section 14.35.1)	Optional
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". X-CDMI-Partial works across CDMI and non-CDMI operations.	Optional

6.4.4 Request message body

The request message body contains the data to be stored in the value of the data object.

6.4.5 Response header

The HTTP response header for updating a data object using HTTP is shown in Table 12.

Table 12 — Response header - Update a data object using HTTP

Header	Type	Description	Requirement
Location	Header string	The server shall respond with the URI to which the reference redirects if the object is a reference.	Conditional

6.4.6 Response message body

A response body can be provided as per RFC 2616.

6.4.7 Response status

The HTTP status codes that occur when updating a data object using HTTP are described in Table 13.

Table 13 — HTTP status codes - Update a data object using HTTP

HTTP status	Description
`204 No Content`	The data object content was returned in the response.
`302 Found`	The resource is a reference to another resource.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.
`409 Conflict`	The operation conflicts with a non-CDMI access protocol lock or has caused a state transition error on the server.

6.4.8 Examples

EXAMPLE 1 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 following shows the response.

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

EXAMPLE 2 PUT to the data object URI to update four bytes within the value of the data object:

```
PUT /MyContainer/MyDataObject.txt HTTP/1.1
Host: cloud.example.com
Content-Range: bytes 21-24/37
Content-Type: text/plain
Content-Length: 4

that
```

The following shows the response.

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

6.5 Delete a data object using HTTP

6.5.1 Synopsis

The following HTTP DELETE deletes an existing data object at the specified URI:

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

Where:

- <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 shall also be accessible at <root URI>/cdmi_objectid/<objectID>.

6.5.2 Capability

The following capability describes 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_dataobject capability in the specified object.

6.5.3 Request headers

Request headers can be provided as per RFC 2616.

6.5.4 Request message body

A request body can be provided as per RFC 2616.

6.5.5 Response headers

Response headers can be provided as per RFC 2616.

6.5.6 Response message body

A response body can be provided as per RFC 2616.

6.5.7 Response status

Table 14 describes the HTTP status codes that occur when deleting a data object using HTTP.

Table 14 — HTTP status codes - Delete a data object using HTTP

HTTP status	Description
`204 No Content`	The data object was successfully deleted.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.
`409 Conflict`	The operation conflicts with a non-CDMI access protocol lock or has caused a state transition error on the server, or the data object cannot be deleted.

6.5.8 Example

EXAMPLE DELETE to the data object URI:

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

The following shows the response.

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

7 Container object resource operations using HTTP

7.1 Overview

Container objects are the fundamental grouping mechanism of stored data within CDMITM and are analogous to directories in a file system. Each container object has zero or more child objects.

Following the URI conventions for hierarchical paths, container URIs shall consist of one or more container names that are separated by forward slashes ("/") and that end with a forward slash ("/").

As basic HTTP operations do not use the CDMI MIME types that distinguish data object operations from container object operations, a CDMI implementation shall use the presence or absence of a forward slash at the end of a URI to distinguish between a container object create or a data object create, respectively.

If a basic HTTP read, update, or delete operation is performed against an existing container resource and the trailing slash at the end of the URI is omitted, the server shall respond with an HTTP status code of `301 Moved Permanently`. In addition, a Location header containing the URI with the trailing slash added shall be returned.

A CDMI server differentiates between HTTP and CDMI operations using the standard Content-Type and Accept headers. When CDMI MIME types defined in RFC 6208 are used in these headers, this indicates that CDMI behaviors, as described in Clause 9, are used in addition to the standard HTTP behaviors. When CDMI MIME types are used, the X-CDMI-Specification-Version header is included to indicate which version of CDMI is being requested by the client and provided by the server.

A CDMI implementation that supports container objects shall include support for basic container object HTTP operations corresponding with the CDMI capabilities that are published by the implementation. Capabilities allow a client to discover which operations (such as create, update, delete, etc.) are supported and are described in Clause 12.

7.2 Create a container object using HTTP

7.2.1 Synopsis

To create a new container object, the following request shall be performed:

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

Where:

- <root URI> is the path to the CDMI cloud.
- <ContainerName> is zero or more intermediate container objects that already exist, with one slash (i.e., "/") between each pair of container object names.
- <NewContainerName> is the name specified for the container object to be created.

After it is created, the container object shall also be accessible at <root URI>/cdmi_objectid/<objectID>/.

The presence of a trailing slash at the end of the HTTP PUT URI indicates that a container object is being created and distinguishes it from a request to create a data object.

7.2.2 Capability

The following capability describes the supported operations that may be performed when creating a new container object:

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 object.

7.2.3 Request headers

Request headers can be provided as per RFC 2616.

7.2.4 Request message body

A request body shall not be provided.

7.2.5 Response headers

Response headers can be provided as per RFC 2616.

7.2.6 Response message body

A response body can be provided as per RFC 2616.

7.2.7 Response status

Table 15 describes the HTTP status codes that occur when creating a container object using HTTP.

Table 15 — HTTP status codes - Create a container object using HTTP

HTTP status	Description
`201 Created`	The new container object was created.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.
`409 Conflict`	The operation conflicts with a non-CDMI access protocol lock or has caused a state transition error on the server.

7.2.8 Example

EXAMPLE PUT to the URI the container object name:

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

The following shows the response.

```
HTTP/1.1 201 Created
```

7.3 Read a container object using HTTP

Reading a container object using HTTP is not defined by this version of this International Standard. 9.3 describes how to read a container object using CDMI. A server is allowed to implement responses such as an Apache directory listing or an S3/Swift-style bucket listing.

7.4 Update a container object using HTTP

Updating a container object using HTTP is not defined by this version of this International Standard. 9.4 describes how to update a container object using CDMI.

7.5 Delete a container object using HTTP

7.5.1 Synopsis

To delete an existing container object, including all contained children and snapshots, the following request shall be performed:

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

Where:

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

The object shall also be accessible at <root URI>/cdmi_objectid/<objectID>/.

7.5.2 Capability

The following capability describes the supported operations that may be performed when deleting an existing container object:

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

7.5.3 Request headers

Request headers can be provided as per RFC 2616.

7.5.4 Request message body

A request body can be provided as per RFC 2616.

7.5.5 Response headers

Response headers can be provided as per RFC 2616.

7.5.6 Response message body

A response body can be provided as per RFC 2616.

7.5.7 Response status

Table 16 describes the HTTP status codes that occur when deleting a container object using HTTP.

Table 16 — HTTP status codes - Delete a container object using HTTP

HTTP status	Description
`204 No Content`	The container object was successfully deleted.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.
`409 Conflict`	The operation conflicts with a non-CDMI access protocol lock or has caused a state transition error on the server.

7.5.8 Example

EXAMPLE DELETE to the container object URI:

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

The following shows the response.

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

7.6 Create (POST) a new data object using HTTP

7.6.1 Synopsis

To create a new data object in a specified container where the name of the data object is a server-assigned object identifier, the following request shall be performed:

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

Where:

- <root URI> is the path to the CDMI cloud.
- <ContainerName> is zero or more intermediate container objects that already exist, with one slash (i.e., "/") between each pair of container object names.

The data object shall be accessible as a child of the container with a server-assigned name and shall also be accessible at <root URI>/cdmi_objectid/<objectID>.

HTTP POST to a container is used to enable CDMI servers to support RFC 1867 form-based file uploading. When implementing RFC 1867, the CDMI server-assigned name can be the user-provided file name.

7.6.2 Capabilities

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

- Support for the ability to create data objects through this operation is indicated by the presence of both the cdmi_post_dataobject and cdmi_create_dataobject capabilities in the specified container object.
- If the new data object is being created in "/cdmi_objectid/", support for the ability to create the value of the new data object in specified byte ranges is indicated by the presence of the "cdmi_create_value_range_by_ID" system capability.
- If the new data object is being created in a container object, support for the ability to create the value of the new data object in specified byte ranges is indicated by the presence of the "cdmi_create_value_range" capability in the parent container.
- Support for the ability to create a new data object by ID using multi-part MIME is indicated by the presence of the "cdmi_multipart_mime" system-wide capability.

7.6.3 Request headers

The HTTP request header for creating a new data object using HTTP is shown in Table 17.

Table 17 — Request headers - Create a new data object using HTTP

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 converted to lower case and stored in the mimetype field of the data object. If the content type includes the charset parameter as defined in RFC 2616 of "utf-8" (e.g., ";charset=utf-8"), the valuetransferencoding field of the data object shall be set to "utf-8". Otherwise, the valuetransferencoding field of the data object shall be set to "base64".	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". X-CDMI-Partial works across CDMI and non-CDMI operations.	Optional

7.6.4 Request message body

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

7.6.5 Response header

The HTTP response header for creating a new data object using HTTP is shown in Table 18.

Table 18 — Response header - Create a new data object using HTTP

Header	Type	Description	Requirement
Location	Header string	The unique absolute URI 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 URI in the form: http://host:port/<root URI>/<ContainerName>/<ObjectID> or https://host:port/<root URI>/<ContainerName>/<ObjectID>.	Mandatory

7.6.6 Response message body

A response body can be provided as per RFC 2616.

7.6.7 Response status

Table 19 describes the HTTP status codes that occur when creating a new data object using HTTP.

Table 19 — HTTP status codes - Create a new data object using HTTP

HTTP status	Description
`201 Created`	The new data object was created.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.

7.6.8 Examples

EXAMPLE 1 POST to the container object URI the data object contents:

```
POST /MyContainer/ HTTP/1.1
Host: cloud.example.com
Content-Type: text/plain;charset=utf-8

<object contents>
```

The following shows the response.

```
HTTP/1.1 201 Created
Location: http://cloud.example.com/MyContainer/00007ED900104E1D14771DC67C27BF8B
utf-8
```

Section III

CDMI Core

8 Data object resource operations using CDMI

8.1 Overview

Data objects are the fundamental storage component within CDMI™ and are analogous to files within a file system. 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 two ways:

- by name (e.g., http://cloud.example.com/dataobject); and
- by object ID (e.g., http://cloud.example.com/cdmi_objectid/00007ED90010D891022876A8DE0BC0FD).

Every data object has a single, globally-unique object identifier (ID) that remains constant for the life of the object. Each data object shall 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.

EXAMPLE 1 The "budget.xls" data object stored at the following URI would inherit data system metadata from its parent container, "finance":

http://cloud.example.com/finance/budget.xls

Individual fields within a data object can be accessed by specifying the field name after a question mark "?" that is appended to the end of the data object URI.

EXAMPLE 2 The following URI returns the value field in the response body:

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

The encoding of the data transported in the data object value field depends on the data object valuetransferencoding field.

- If the value transfer encoding of the object is set to "utf-8", the data stored in the value of the data object shall be a valid UTF-8 string and shall be transported as a UTF-8 string in the value field.
- If the value transfer encoding of the object is set to "base64", the data stored in the value of the data object can contain arbitrary binary sequences, and it shall be transported as a base 64-encoded string in the value field.

Specific ranges of the value of a data object can be accessed by specifying a byte range after the value field name.

EXAMPLE 3 The following URI returns the first thousand bytes in the value field:

http://cloud.example.com/dataobject?value:0-999

Because a byte range of a UTF-8 string is often not a valid UTF-8 string, the response to a range request shall always be transported in the value field as a base 64-encoded string. Likewise, when updating a range of bytes within the value of a data object, the contents of the value field shall be transported as a base 64-encoded string.

Byte ranges are specified as single inclusive byte ranges as per Section 14.35.1 of RFC 2616.

A list of unique fields, separated by a semicolon ";" can be specified, allowing multiple fields to be accessed in a single request.

EXAMPLE 4 The following URI returns the value and metadata fields in the response body:

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

If read access to any of the requested fields is not permitted by the object ACL, only the permitted fields shall be returned. If no requested fields are permitted to be read, an HTTP status code of `403 Forbidden` shall be returned to the client.

If write access to any of the requested fields is not permitted by the object ACL, no updates shall be performed, and an HTTP status code of `403 Forbidden` shall be returned to the client.

When a client provides fields that are not defined in this International Standard or deserializes an object containing fields that are not defined in this International Standard, these fields shall be stored as part of the object but shall not be interpreted.

The value of a data object can also be specified and retrieved using multi-part MIME, where the CDMI JSON is transferred in the first MIME part, and the raw object value is transferred in the second MIME part. Each MIME part, including any header fields, shall conform to RFC 2045, RFC 2046, and RFC 2616. The length of each part can optionally be specified by a Content-Length header in addition to the MIME boundary delimiter.

Multiple non-overlapping ranges of the value of a data object can also be accessed or updated in a multi-part MIME operation by transferring one MIME part for each range of the value. The byte ranges for these operations shall be specified as per Section 14.35.1 of RFC 2616.

Multi-part MIME enables the efficient transfer of binary data alongside CDMI object metadata without incurring the overhead of the UTF-8 or Base64 encoding and validation required to represent binary data in JSON.

8.1.1 Data object metadata

Data object metadata can also include arbitrary user-supplied metadata, storage system metadata, and data system metadata, as specified in Clause 16. Metadata shall be stored as a valid UTF-8 string. Binary data stored in user metadata shall be first encoded such that it can be contained in a UTF-8 string, with the use of base 64 encoding recommended.

8.1.2 Data object consistency

Writing to a data object is an atomic operation.

- If a client reads a data object simultaneously with a write to that same data object, the reading client shall get either the old version or the new version, but not a mixture of both.
- If a write is terminated due to errors, the contents of the data object shall be as if the write never occurred (i.e., writes are atomic in the face of errors).

Create and update timestamps that are returned in response to multiple client writes to a given object indicate that a specific write is the newest (i.e., the write whose data is expected to be returned to subsequent reads until another write is processed). However, there is no guarantee that the write with the latest timestamp is the one whose data is returned on subsequent reads.

Range writes can result in a gap in an object value that have had no data written to them. Reading from a gap in a data object value shall return zero for each byte read.

Implementations of this International Standard shall provide the atomicity features described in this subclause for data objects that are accessed via CDMI. The atomicity properties of data objects that are accessed by protocols other than CDMI are outside the scope of this International Standard.

8.1.3 Data object representations

The representations in this clause are shown using JSON notation. Both clients and servers shall support UTF-8 JSON representation. The request and response body JSON fields can be specified or returned in any order, with the exception that, if present, for data objects, the valuerange and value fields shall appear last and in that order.

8.2 Create a data object using CDMI

8.2.1 Synopsis

To create a new data object, the following request shall be performed:

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

To create a new data object by ID, see 9.7 "Create (POST) a new queue object using CDMI".

Where:

- <root URI> is the path to the CDMI cloud.
- <ContainerName> is zero or more intermediate containers that already exist, with one slash (i.e., "/") between each pair of container names.
- <DataObjectName> is the name specified for the data object to be created.

After it is created, the data object shall also be accessible at <root URI>/cdmi_objectid/<objectID>.

8.2.2 Delayed completion of create

In response to a create operation for a data object, the server can return an HTTP status code of `202 Accepted` to indicate that the object is in the process of being created. This response is useful for long-running operations (e.g., copying a large data object from a source URI). Such a response has the following implications.

- The server shall return a Location header with an absolute URI to the object to be created along with an HTTP status code of `202 Accepted`.
- With an HTTP status code of `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; and
  - availability of space to create the object or at least enough space to create a URI to report an error.
- A client might not be able to immediately access the created object, e.g., due to delays resulting from the implementation's use of eventual consistency.

The client performs GET operations to the URI to track the progress of the operation. In response, 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 contains the percentage of the operation that has completed (0 to 100).

GET shall not return any value for the data object when completionStatus is not "Complete". If 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.

8.2.3 Capabilities

The following capabilities describe the supported operations that can 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_copy_dataobject 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_move_dataobject 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.
- Support for the ability to create the value of a new data object in specified byte ranges is indicated by the presence of the "cdmi_create_value_range" capability in the parent container.
- Support for the ability to create a new data object using multi-part MIME is indicated by the presence of the "cdmi_multipart_mime" system-wide capability.

8.2.4 Request headers

The HTTP request headers for creating a data object using CDMI are shown in Table 20.

Table 20 — Request headers for creating a data object using CDMI

Header	Type	Description	Requirement
Accept	Header string	"application/cdmi-object" or a consistent value as per clause 5.13.2 "Content-type negotiation".	Optional
Content-Type	Header string	"application/cdmi-object" or "multipart/mixed" • If "multipart/mixed" is specified, the body shall consist of at least two MIME parts, where the first part shall contain a body of content-type "application/cdmi-object", and the second and subsequent parts shall contain one or more byte ranges of the value as described in 6.2 "Create a data object using HTTP". • If multiple byte ranges are included and the Content-Range header is omitted for a part, the data in the part shall be appended to the data in the preceding part, with the first part having a byte offset of zero.	Mandatory
X-CDMI-Specification-Version	Header string	A comma-separated list of versions that the client supports, e.g., "1.1, 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. If X-CDMI-Partial is present, the completionStatus field in the response body shall be set to "Processing". X-CDMI-Partial works across CDMI and non-CDMI operations.	Optional

8.2.5 Request message body

The request message body fields for creating a data object using CDMI are shown in Table 21.

Table 21 — Request message body - Create a data object using CDMI (Sheet 1 of 3)

Field name	Type	Description	Requirement
mimetype	JSON string	MIME type of the data contained within the value field of the data object • This field may be included when creating by value or when deserializing, serializing, copying, and moving a data object. • If this field is not included and multi-part MIME is not being used, the value of "text/plain" shall be assigned as the field value. • If this field is not included and multi-part MIME is being used, the value of the Content-Type header of the second MIME part shall be assigned as the field value. • This field shall be stored as part of the data object. • This MIME type value shall be converted to lower case before being stored.	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 (i.e., "{}") 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 (see cdmi_member_privileges in Table 63 "Required settings for domain member user objects"). • If not specified, the domain of the parent container shall be used.	Optional
deserialize	JSON string	URI of a serialized data object that shall be deserialized to create the new data object	Optional[a]
serialize	JSON string	URI of a CDMI object that shall be serialized into the new data object	Optional[a]

[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored. If more than one of these fields is supplied, the server shall respond with an HTTP status code of `400 Bad Request`.

Table 21 — Request message body - Create a data object using CDMI (Sheet 2 of 3)

Field name	Type	Description	Requirement
copy	JSON string	URI of a source data object or queue object that shall be copied into the new destination data object. • If the destination data object URI and the copy source object URI both do not specify individual fields, the destination data object shall be a complete copy of the source data object. • If the destination data object URI or the copy source object URI specifies individual fields, only the fields specified shall be used to create the destination data object. If specified fields are not present in the source, default field values shall be used. • If the destination data object URI and the copy source object URI both specify fields, an HTTP status code of `400 Bad Request` shall be returned to the client. • If the copy source object URI points to a queue object, as part of the copy operation, multiple queue values shall be concatenated into a single data object value. • If the copy source object URI points to one or more queue object values, as part of the copy operation, the specified queue values shall be concatenated into a single data object value. • If there are insufficient permissions to read the data object at the source URI or create the data object at the destination URI, or if the read operation fails, the copy shall return an HTTP status code of `400 Bad Request`, and the destination object shall not be created.	Optional[a]
move	JSON string	URI of an existing local or remote data object (source URI) that shall be relocated to the URI specified in the PUT. The contents of the object, including the object ID, shall be preserved by a move, and the data object at the source URI shall be removed after the data object at the destination has been successfully created. If there are insufficient permissions to read the data object at the source URI, write the data object at the destination URI, or delete the data object at the source URI, or if any of these operations fail, the move shall return an HTTP status code of `400 Bad Request`, and the source and destination are left unchanged.	Optional[a]
reference	JSON string	URI of a data object that shall be redirected to by a reference. If any other fields are supplied when creating a reference, the server shall respond with an HTTP status code of `400 Bad Request`.	Optional[a]
deserializevalue	JSON string	A data object serialized as specified in Clause 15 and encoded using base 64 encoding rules described in RFC 4648.	Optional[a]

[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored. If more than one of these fields is supplied, the server shall respond with an HTTP status code of `400 Bad Request`.

Table 21 — Request message body - Create a data object using CDMI (Sheet 3 of 3)

Field name	Type	Description	Requirement
valuetransferencoding	JSON string	The value transfer encoding used for the data object value. Two value transfer encodings are defined. • "utf-8" indicates that the data object contains a valid UTF-8 string, and it shall be transported as a UTF-8 string in the value field. • "base64" indicates that the data object may contain arbitrary binary sequences, and it shall be transported as a base 64-encoded string in the value field. Setting the contents of the data object value field to any value other than a valid base 64 string shall result in an HTTP status code of `400 Bad Request` being returned to the client. • This field shall only be included when creating a data object by value. • If this field is not included and multi-part MIME is not being used, the value of "utf-8" shall be assigned as the field value. • If this field is not included and multi-part MIME is being used, the value of "utf-8" shall be assigned as the field value if the Content-Type header of the second and all MIME parts includes the charset parameter as defined in RFC 2046 of "utf-8" (e.g., ";charset=utf-8"). Otherwise, the value of "base64" shall be assigned as the field value. This field applies only to the encoding of the value when represented in JSON; the Content-Transfer-Encoding header of the part specifies the encoding of the value within a multi-part MIME request, as defined in RFC 2045. • This field shall be stored as part of the object.	Optional
value	JSON string	The data object value • If this field is not included and multi-part MIME is not being used, an empty JSON string (i.e., "") shall be assigned as the field value. • If this field is not included and multi-part MIME is being used, the contents of the second MIME part shall be assigned as the field value. • If the valuetransferencoding field indicates UTF-8 encoding, the value shall be a UTF-8 string escaped using the JSON escaping rules described in RFC 4627. • If the valuetransferencoding field indicates base 64 encoding, the value shall be first encoded using the base 64 encoding rules described in RFC 4648.	Optional[a]
[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored. If more than one of these fields is supplied, the server shall respond with an HTTP status code of `400 Bad Request`.			

8.2.6 Response headers

The HTTP response headers for creating a data object using CDMI are shown in Table 22.

Table 22 — Response headers - Create a data object using CDMI

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, e.g., "1.1". If the server does not support any of the versions that the client supports, the server shall return an HTTP status code of `400 Bad Request`.	Mandatory
Location	Header string	When an HTTP status code of `202 Accepted` is returned, the server shall respond with the absolute URL of the object that is in the process of being created.	Conditional

8.2.7 Response message body

The response message body fields for creating a data object using CDMI are shown in Table 23.

Table 23 — Response message body - Create a data object using CDMI (Sheet 1 of 2)

Field name	Type	Description	Requirement
objectType	JSON string	"application/cdmi-object"	Mandatory
objectID	JSON string	Object ID of the object	Mandatory
objectName	JSON string	Name of the object	Mandatory
parentURI	JSON string	URI for the parent object. Appending the objectName to the parentURI shall always produce a valid URI for the object.	Mandatory
parentID	JSON string	Object ID of the parent container object	Mandatory
domainURI	JSON string	URI of the owning domain	Mandatory
capabilitiesURI	JSON string	URI to the capabilities for the object	Mandatory
completionStatus	JSON string	A string indicating if the object is still in the process of being created or updated by another operation, and after that operation is complete, indicates if it was successfully created or updated or if an error occurred. The value shall be the string "Processing", the string "Complete", or an error string starting with the value "Error".	Mandatory

Table 23 — Response message body - Create a data object using CDMI (Sheet 2 of 2)

Field name	Type	Description	Requirement
percentComplete	JSON string	• When the value of completionStatus is "Processing", this field, if provided, shall indicate the percentage of completion as a numeric integer value from 0 through 100. • When the value of completionStatus is "Complete", this field, if provided, shall contain the value "100". • When the value of completionStatus is "Error", this field, if provided, can contain any integer 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 Clause 16 for a further description of metadata.	Mandatory

8.2.8 Response status

The HTTP status codes that occur when creating a data object using CDMI are described in Table 24.

Table 24 — HTTP status codes - Create a data object using CDMI

HTTP status	Description
`201 Created`	The new data object was created.
`202 Accepted`	The data object is in the process of being created. The CDMI client should monitor the completionStatus and percentComplete fields to determine the current status of the operation.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.
`409 Conflict`	The operation conflicts with a non-CDMI access protocol lock or has caused a state transition error on the server.

8.2.9 Examples

EXAMPLE 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: 1.1

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

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

The following shows the response.

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

{
    "objectType" : "application/cdmi-object",
    "objectID" : "00007ED90010D891022876A8DE0BC0FD",
    "objectName" : "MyDataObject.txt",
    "parentURI" : "/MyContainer/",
    "parentID" : "00007E7F00102E230ED82694DAA975D2",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/dataobject/",
    "completionStatus" : "Complete",
    "mimetype" : "text/plain",
    "metadata" : {
        "cdmi_size" : "37"
    }
}
```

EXAMPLE 2 PUT to the container URI the data object name and binary 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: 1.1

{
   "mimetype" : "text/plain",
   "metadata" : { },
   "valuetransferencoding" : "base64"
   "value" : "VGhpcyBpcyB0aGUgVmFsdWUgb2YgdGhpcyBEYXRhIE9iamVjdA=="
}
```

The following shows the response.

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

{
   "objectType": "application/cdmi-object",
   "objectID": "00007ED9001008C174ABCE6AC3287E5F",
   "objectName": "MyDataObject.txt",
   "parentURI": "/MyContainer/",
   "parentID": "00007E7F00102E230ED82694DAA975D2",
   "domainURI": "/cdmi_domains/MyDomain/",
   "capabilitiesURI": "/cdmi_capabilities/dataobject/",
   "completionStatus": "Complete",
   "mimetype": "text/plain",
   "metadata": {
      "cdmi_size": "37"
   }
```

EXAMPLE 3 } PUT to the container URI the data object name and binary contents using multi-part MIME:

```
PUT /MyContainer/MyDataObject.txt HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
Content-Type: multipart/mixed; boundary=gc0p4Jq0M2Yt08j34c0p
X-CDMI-Specification-Version: 1.1

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/cdmi-object

{
    "domainURI": "/cdmi_domains/MyDomain/",
    "metadata": {
        "colour": "blue"
    }
}

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary

<37 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p--
```

The following shows the response.

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

{
  "objectType": "application/cdmi-object",
  "objectID": "00007ED900103ADE9DE3A8D1CF5436A3",
  "objectName": "MyDataObject.txt",
  "parentURI": "/MyContainer/",
  "parentID" : "00007E7F00102E230ED82694DAA975D2",
  "domainURI": "/cdmi_domains/MyDomain/",
  "capabilitiesURI": "/cdmi_capabilities/dataobject/",
  "completionStatus": "Complete",
  "mimetype": "application/octet-stream",
  "metadata": {
    "cdmi_size": "37",
    "colour": "blue",
    ...
  }
}
```

EXAMPLE 4 PUT to the container URI the data object name and binary contents using multi-part MIME with optional content-lengths for the parts:

```
PUT /MyContainer/MyDataObject.txt HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
Content-Type: multipart/mixed; boundary=gc0p4Jq0M2Yt08j34c0p
X-CDMI-Specification-Version: 1.1

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/cdmi-object
Content-Length: 82

{
  "domainURI": "/cdmi_domains/MyDomain/",
  "metadata": {
    "colour": "blue"
  }
}
```

```
--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary
Content-Length: 37

<37 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p--
```

The following shows the response.

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

{
  "objectType": "application/cdmi-object",
  "objectID": "00007ED900103ADE9DE3A8D1CF5436A3",
  "objectName": "MyDataObject.txt",
  "parentURI": "/MyContainer/",
  "parentID" : "00007E7F00102E230ED82694DAA975D2",
  "domainURI": "/cdmi_domains/MyDomain/",
  "capabilitiesURI": "/cdmi_capabilities/dataobject/",
  "completionStatus": "Complete",
  "mimetype": "application/octet-stream",
  "metadata": {
    "cdmi_size": "37",
    "colour": "blue",
    ...
  }
}
```

8.3 Read a data object using CDMI

8.3.1 Synopsis

The following HTTP GET 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>;...
```

Where:

- <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 of the data object value to be returned in the value field.<prefix> is a matching prefix that returns all metadata items that start with the prefix value.

The object shall also also be accessible at <root URI>/cdmi_objectid/<objectID>.

8.3.2 Capabilities

The following capabilities describe the supported operations that can 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.
- 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.
- Support for the ability to read a data object using multi-part MIME is indicated by the presence of the "cdmi_multipart_mime" system-wide capability.

8.3.3 Request headers

The HTTP request headers for reading a data object using CDMI are shown in Table 25.

Table 25 — Request headers - Read a data object using CDMI

Header	Type	Description	Requirement
Accept	Header string	"application/cdmi-object", "multipart/mixed", or a consistent value as per clause 5.13.2 "Content-type negotiation"	Optional
X-CDMI-Specification-Version	Header string	A comma-separated list of versions that the client supports, e.g., "1.1, 1.5, 2.0"	Mandatory

8.3.4 Request message body

A request body shall not be provided.

8.3.5 Response headers

The HTTP response headers for reading a data object using CDMI are shown in Table 26.

Table 26 — Response headers - Read a data object using CDMI

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, e.g., "1.1". If the server does not support any of the versions that the client supports, the server shall return an HTTP status code of `400 Bad Request`.	Mandatory
Content-Type	Header string	"application/cdmi-object" or "multipart/mixed" • If "multipart/mixed", the body shall consist of at least two MIME parts, where the first part shall contain a body of content-type "application/cdmi-object" and the second and subsequent parts shall contain the requested byte ranges of the value as described in 8.4 "Update a data object using CDMI". • If multiple byte ranges are included and the Content-Range header is omitted for a part, the data in the part shall be appended to the data in the preceding part, with the first part having a byte offset of zero.	Mandatory
Location	Header string	The server shall respond with the URI that the reference redirects to if the object is a reference.	Conditional

8.3.6 Response message body

The response message body fields for reading a data object using CDMI are shown in Table 27.

Table 27 — Response message body - Read a data object using CDMI (Sheet 1 of 2)

Field name	Type	Description	Requirement
objectType	JSON string	"application/cdmi-object"	Mandatory
objectID	JSON string	Object ID of the object	Mandatory
objectName	JSON string	Name of the object • For objects in a container, the objectName field shall be returned. • For objects not in a container (objects that are only accessible by ID), the objectName field does not exist and shall not be returned.	Conditional
parentURI	JSON string	URI for the parent object • For objects in a container, the parentURI field shall be returned. • For objects not in a container (objects that are only accessible by ID), the parentURI field does not exist and shall not be returned. Appending the objectName to the parentURI shall always produce a valid URI for the object.	Conditional
parentID	JSON string	Object ID of the parent container object • For objects in a container, the parentID field shall be returned. • For objects not in a container (objects that are only accessible by ID), the parentID field does not exist and shall not be returned.	Conditional
domainURI	JSON string	URI of the owning domain	Mandatory
capabilitiesURI	JSON string	URI to the capabilities for the object	Mandatory
completionStatus	JSON string	A string indicating if the object is still in the process of being created or updated by another operation, and after that operation is complete, indicates if it was successfully created or updated or if 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	• When the value of completionStatus is "Processing", this field, if provided, shall indicate the percentage of completion as a numeric integer value from 0 through 100. • When the value of completionStatus is "Complete", this field, if provided, shall contain the value "100". • When the value of completionStatus is "Error", this field, if provided, may contain any integer value from 0 through 100.	Optional
mimetype	JSON string	MIME type of the value of the data object	Mandatory

Table 27 — Response message body - Read a data object using CDMI (Sheet 2 of 2)

Field name	Type	Description	Requirement
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 Clause 16 for a further description of metadata.	Mandatory
valuerange	JSON string	The range of bytes of the data object to be returned in the value field • If a specific value range has been requested, the valuerange field shall correspond to the bytes requested. If the request extends beyond the end of the value, the valuerange field shall indicate the smaller byte range returned. • If the object value has gaps (due to PUTs with non-contiguous value ranges), the value range will indicate the range to the first gap in the object value. • The cdmi_size storage system metadata of the data object shall always indicate the complete size of the object, including zero-filled gaps.	Mandatory
valuetransferencoding	JSON string	The value transfer encoding used for the data object value. Two value transfer encodings are defined: • "utf-8" indicates that the data object contains a valid UTF-8 string, and it shall be transported as a UTF-8 string in the value field. • "base64" indicates that the data object may contain arbitrary binary sequences, and it shall be transported as a base 64-encoded string in the value field.	Mandatory
value	JSON string	The data object value • If the valuetransferencoding field indicates UTF-8 encoding, the value field shall contain a UTF-8 string using JSON escaping rules described in RFC 4627. • If the valuetransferencoding field indicates base 64 encoding, the value field shall contain a base 64-encoded string as described in RFC 4648. • The value field shall not be provided when using multi-part MIME. • The value field shall only be provided when the completionStatus field contains "Complete". • When reading a value, zeros shall be returned for any gaps resulting from non-contiguous writes.	Conditional

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.

8.3.7 Response status

The HTTP status codes that occur when reading a data object using CDMI are described in Table 28.

Table 28 — HTTP status codes - Read a data object using CDMI

HTTP status	Description
200 OK	The data object content was returned in the response.
202 Accepted	The data object is in the process of being created. The CDMI client should monitor the completionStatus and percentComplete fields to determine the current status of the operation.
302 Found	The resource is a reference to another resource.
400 Bad Request	The request contains invalid parameters or field names.
401 Unauthorized	The authentication credentials are missing or invalid.
403 Forbidden	The client lacks the proper authorization to perform this request.
404 Not Found	The resource was not found at the specified URI.
406 Not Acceptable	The server is unable to provide the object in the specified in the Accept header.

8.3.8 Examples

EXAMPLE 1 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: 1.1
```

The following shows the response.

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

{
    "objectType" : "application/cdmi-object",
    "objectID" : "00007ED90010D891022876A8DE0BC0FD",
    "objectName" : "MyDataObject.txt",
    "parentURI" : "/MyContainer/",
    "parentID" : "00007E7F00102E230ED82694DAA975D2",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/dataobject/",
    "completionStatus" : "Complete",
    "mimetype" : "text/plain",
    "metadata" : {
        "cdmi_size" : "37"
    },
    "valuerange" : "0-36",
    "valuetransferencoding" : "utf-8",
    "value" : "This is the Value of this Data Object"
}
```

EXAMPLE 2 GET to the data object URI by ID to read all fields of the data object:

```
GET /cdmi_objectid/00007ED90010D891022876A8DE0BC0FD HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
X-CDMI-Specification-Version: 1.1
```

The following shows the response.

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

{
    "objectType" : "application/cdmi-object",
    "objectID" : "00007ED90010D891022876A8DE0BC0FD",
    "objectName" : "MyDataObject.txt",
    "parentURI" : "/MyContainer/",
    "parentID" : "00007E7F00102E230ED82694DAA975D2",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/dataobject/",
    "completionStatus" : "Complete",
    "mimetype" : "text/plain",
    "metadata" : {
        "cdmi_size" : "37"
    },
    "valuetransferencoding" : "utf-8",
    "valuerange" : "0-36",
    "value" : "This is the Value of this Data Object"
}
```

EXAMPLE 3 GET to the data object URI to read the value and mimetype fields 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: 1.1
```

The following shows the response.

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

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

EXAMPLE 4 GET to the data object URI to read the first 11 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: 1.1
```

The following shows the response.

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

{
    "valuerange" : "0-10",
    "value" : "VGhpcyBpcyB0aGU="
}
```

EXAMPLE 5 GET to the data object URI to read the data object using multi-part MIME:

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

The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: multipart/mixed; boundary=gc0p4Jq0M2Yt08j34c0p
X-CDMI-Specification-Version: 1.1

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/cdmi-object

{
  "objectType": "application/cdmi-object",
  "objectID": "00007ED90010C2414303B5C6D4F83170",
  "objectName": "MyDataObject.txt",
  "parentURI": "/MyContainer/",
  "parentID" : "00007E7F00102E230ED82694DAA975D2",
  "domainURI": "/cdmi_domains/MyDomain/",
  "capabilitiesURI": "/cdmi_capabilities/dataobject/",
  "completionStatus": "Complete",
  "mimetype": "application/octet-stream",
  "metadata": {
    "cdmi_size": "37",
    "colour": "blue",
    ...
  },
  "valuerange": "0-36",
  "valuetransferencoding": "base64"
}

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary

<37 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p–
```

EXAMPLE 6 GET to the data object URI to read the data object using multi-part MIME, with optional content-lengths for the parts:

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

The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: multipart/mixed; boundary=gc0p4Jq0M2Yt08j34c0p
X-CDMI-Specification-Version: 1.1

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/cdmi-object
Content-Length: 505

{
  "objectType": "application/cdmi-object",
  "objectID": "00007ED90010C2414303B5C6D4F83170",
  "objectName": "MyDataObject.txt",
  "parentURI": "/MyContainer/",
  "parentID" : "00007E7F00102E230ED82694DAA975D2",
  "domainURI": "/cdmi_domains/MyDomain/",
  "capabilitiesURI": "/cdmi_capabilities/dataobject/",
  "completionStatus": "Complete",
  "mimetype": "application/octet-stream",
  "metadata": {
    "cdmi_size": "37",
    "colour": "blue",
    ...
  },
```

```
  "valuerange": "0-36",
  "valuetransferencoding": "base64"
}

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary
Content-Length: 37

<37 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p–
```

EXAMPLE 7 GET to the data object URI to read the metadata and multiple byte ranges of the binary contents using multi-part MIME:

```
GET /MyContainer/MyDataObject.txt?metadata;value:0-10;value:21-24 HTTP/1.1
Host: cloud.example.com
Accept: multipart/mixed
X-CDMI-Specification-Version: 1.1
```

The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: multipart/mixed; boundary=gc0p4Jq0M2Yt08j34c0p
X-CDMI-Specification-Version: 1.1

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/cdmi-object

{
  "metadata": {
    "cdmi_size": "37",
    "colour": "blue",
    ...
  }
}

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary
Content-Range: bytes 0-10/37

<11 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary
Content-Range: bytes 21-24/37

<4 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p--
```

8.4 Update a data object using CDMI

8.4.1 Synopsis

The following HTTP PUT updates an existing data object at the specified URI:

```
PUT <root URI>/<ContainerName>/<DataObjectName>
PUT <root URI>/<ContainerName>/<DataObjectName>?value:<range>
PUT <root URI>/<ContainerName>/<DataObjectName>?metadata:<metadataname>;....
```

Where:

- <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 for the data object value to be updated.

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

8.4.2 Capabilities

The following capabilities describe the supported operations that can 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 or MIME type 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.
- Support for the ability to modify an existing data object using multi-part MIME is indicated by the presence of the "cdmi_multipart_mime" system-wide capability.

8.4.3 Request headers

The HTTP request headers for updating a data object using CDMI are shown in Table 29.

Table 29 — Request headers - Update a data object using CDMI

Header	Type	Description	Requirement
Content-Type	Header string	"application/cdmi-object" or "multipart/mixed" • If multipart/mixed is specified, the body shall consist of at least two MIME parts, where the first part shall contain a body of content-type "application/cdmi-object" and the second and subsequent parts shall contain one or more byte ranges of the value as described in 8.7. • If multiple byte ranges are included and the "Content-Range" header is omitted for a part, the data in the part shall be appended to the data in the preceding part, with the first part having a byte offset of zero.	Mandatory
X-CDMI-Specification-Version	Header string	A comma-separated list of versions that the client supports, e.g., "1.1, 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". X-CDMI-Partial works across CDMI and non-CDMI operations.	Optional

8.4.4 Request message body

The request message body fields for updating a data object using CDMI are shown in Table 30.

Table 30 — Request message body - Update a data object using CDMI (Sheet 1 of 3)

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 value replaces the existing mimetype field value. • This field may be included when updating by value, deserializing, and copying a data object. • If this field is not included, the existing value of the mimetype field shall be left unchanged. • This field shall be stored as part of the data object. • This mimetype field value shall be converted to lower case before being stored.	Optional
metadata	JSON object	Metadata for the data object. If present, the new metadata specified replaces the existing object metadata. If individual metadata items are specified in the URI, only those items are replaced; other items are preserved. See Clause 16 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 (see cdmi_member_privileges in Table 63). • If not specified, the existing domain shall be preserved.	Optional
deserialize	JSON string	URI of a serialized data object that shall be deserialized to update an existing data object. The object ID of the serialized data object shall match the object ID of the destination data object.	Optional[a]

[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored. If more than one of these fields is supplied, the server shall respond with an HTTP status code of `400 Bad Request`.

Table 30 — Request message body - Update a data object using CDMI (Sheet 2 of 3)

Field name	Type	Description	Requirement
copy	JSON string	URI of a source data object or queue object that shall be copied into an existing destination data object. • If the destination data object URI and the copy source object URI both do not specify individual fields, the destination data object shall be replaced with the source data object. • If the destination data object URI or the copy source object URI specifies individual fields, only the fields specified shall be used to update the destination data object. If specified fields are not present in the source, these fields shall be ignored. • If the destination data object URI and the copy source object URI both specify fields, an HTTP status code of `400 Bad Request` shall be returned to the client. If the copy source object URI points to a queue object, as part of the copy operation, multiple queue values shall be concatenated into a single data object value. If there are insufficient permissions to read the data object at the source URI, update the data object at the destination URI, or if the read operation fails, the copy shall return an HTTP status code of `400 Bad Request`, and the destination shall be left unchanged.	Optional[a]
deserializevalue	JSON string	A data object serialized as specified in Clause 15 and encoded using base 64 encoding rules described in RFC 4648. The object ID of the serialized data object shall match the object ID of the destination data object.	Optional[a]

[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored. If more than one of these fields is supplied, the server shall respond with an HTTP status code of `400 Bad Request`.

Table 30 — Request message body - Update a data object using CDMI (Sheet 3 of 3)

Field name	Type	Description	Requirement
valuetransferencoding	JSON string	The value transfer encoding used for the data object value. Two value transfer encodings are defined: • "utf-8" indicates that the data object contains a valid UTF-8 string and shall be transported as a UTF-8 string in the value field. • "base64" indicates that the data object may contain arbitrary binary sequence and shall be transported as a base 64 encoded string in the value field. Setting the contents of the data object value field to any value other than a valid base 64 string shall result in an HTTP status code of `400 Bad Request` being returned to the client. This field shall only be included when updating a data object by value. • If this field is not included and multi-part MIME is not being used, the existing value of "valuetransferencoding" shall be left unchanged. • If this field is not included and multi-part MIME is being used, the value of "utf-8" shall be assigned as the field value if the "Content-Type" header of the second and all subsequent MIME parts includes the charset parameter as defined in RFC 2046 of "utf-8" (e.g., ";charset=utf-8"). Otherwise, the value of "base64" shall be assigned as the field value. This field applies only to the encoding of the value when represented in JSON; the "Content-Transfer-Encoding" header of the part specifies the encoding of the value within a multi-part MIME request, as defined in RFC 2045. This field shall be stored as part of the object.	Optional
value	JSON string	This field contains the new data for the object. If present, this value replaces the existing value. • If this field is not included and multi-part MIME is being used, the contents of the second and subsequent MIME parts shall be assigned to the corresponding byte ranges of the field value. • If the valuetransferencoding field indicates UTF-8 encoding, the value shall be a UTF-8 string escaped using the JSON escaping rules described in RFC 4627. • If the valuetransferencoding field indicates base 64 encoding, the value shall be first encoded using the base 64 encoding rules described in RFC 4648. • If a value range was specified in the request, the new data shall be 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. When storing a range, the value shall be encoded using base 64, and the valuetransferencoding field shall be set to "base64".	Optional[a]
[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored. If more than one of these fields is supplied, the server shall respond with an HTTP status code of `400 Bad Request`.			

8.4.5 Response header

The HTTP response header for updating a data object using CDMI is shown in Table 31.

Table 31 — Response header - Update a data object using CDMI

Header	Type	Description	Requirement
Location	Header string	The server shall respond with the URI to which the reference redirects if the object is a reference.	Conditional

8.4.6 Response message body

A response body can be provided as per RFC 2616.

8.4.7 Response status

The HTTP status codes that occur when updating a data object using CDMI are described in Table 32.

Table 32 — HTTP status codes - Update a data object using CDMI

HTTP status	Description
`204 No Content`	The data object content was returned in the response.
`302 Found`	The resource is a reference to another resource.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.
`409 Conflict`	The operation conflicts with a non-CDMI access protocol lock or has caused a state transition error on the server.

8.4.8 Examples

EXAMPLE 1 PUT to the data object URI to set new field values:

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

{
    "mimetype" : "text/plain",
    "metadata" : {
        "colour" : "blue",
        "length" : "10"
    },
    "value" : "This is the Value of this Data Object"
}
```

The following shows the response.

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

EXAMPLE 2 PUT to the data object URI to set a new MIME type:

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

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

The following shows the response.

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

EXAMPLE 3 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
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.1

{
    "value" : "dGhhdA=="
}
```

The following shows the response.

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

When updating a value without specifying a value transfer encoding, the client must be aware of the current value transfer encoding of the object.

- If a client sends a value containing a UTF-8 string that is not a valid base 64 string to update an existing object with a value transfer encoding of "base64", the server shall return an error.
- If a client sends a value containing a base 64 string to update an existing object with a value transfer encoding of "utf-8", the server shall not return an error. Instead, the server shall store the literal base 64 character sequence in the data object instead of the data encoded in the base 64 string.

EXAMPLE 4 PUT to the data object URI to replace all metadata with new metadata:

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

{
    "metadata" : {
        "colour" : "red",
        "number" : "7"
    }
}
```

The following shows the response.

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

EXAMPLE 5 PUT to the data object URI to add a new metadata item while preserving existing metadata:

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

{
    "metadata" : {
        "shape" : "round"
    }
}
```

The following shows the response.

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

EXAMPLE 6 PUT to the data object URI to replace just one metadata item with a new value:

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

{
    "metadata" : {
        "colour" : "green"
    }
}
```

The following shows the response.

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

EXAMPLE 7 Delete a single metadata item:

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

{
    "metadata": {}
}
```

The following shows the response.

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

EXAMPLE 8 Add, update, and delete metadata items. Assume a starting condition where the object has a metadata item "colour" with value "green" and a metadata item "shape" with value "round" and does not have a metadata item "size". After the update, "colour" has value "red", "shape" is deleted, and "size" has been added with value "10".

```
PUT /MyContainer/MyDataObject.txt?metadata:colour;metadata:shape;metadata:size
   HTTP/1.1

Host: cloud.example.com
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.1

{
    "metadata": {
        "colour": "red",
        "size": "10"
    }
}
```

The following shows the response.

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

EXAMPLE 9 PUT to the data object URI to set new field values and the binary contents using multi-part MIME:

```
PUT /MyContainer/MyDataObject.txt HTTP/1.1
Host: cloud.example.com
Content-Type: multipart/mixed; boundary=gc0p4Jq0M2Yt08j34c0p
X-CDMI-Specification-Version: 1.1

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/cdmi-object

{
  "metadata": {
    "colour": "red",
    "number": "7"
  }
}

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary

<37 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p--
```

The following shows the response.

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

EXAMPLE 10 PUT to the data object URI to replace just one metadata item and update multiple byte ranges within the binary contents of the data object using multi-part MIME:

```
PUT /MyContainer/BinaryObject.txt?metadata:colour HTTP/1.1
Host: cloud.example.com
Content-Type: multipart/mixed; boundary=gc0p4Jq0M2Yt08j34c0p
X-CDMI-Specification-Version: 1.1

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/cdmi-object

{
  "metadata": {
    "colour": "green"
  }
}

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/octet-stream
Content-Range: bytes 0-10/37

<11 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/octet-stream
Content-Range: bytes 21-24/37

<4 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p--
```

The following shows the response.

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

8.5 Delete a data object using CDMI

8.5.1 Synopsis

The following HTTP DELETE deletes an existing data object at the specified URI:

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

Where:

- <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 shall also be accessible at <root URI>/cdmi_objectid/<objectID>.

8.5.2 Capability

The following capability describes 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_dataobject capability in the specified object.

8.5.3 Request header

The HTTP request header for deleting a data object using CDMI is shown in Table 33.

Table 33 — Request header - Delete a data object using CDMI

Header	Type	Description	Requirement
X-CDMI-Specification-Version	Header string	A comma-separated list of versions that the client supports, e.g., "1.1, 1.5, 2.0"	Mandatory

8.5.4 Request message body

A request body can be provided as per RFC 2616.

8.5.5 Response headers

Response headers can be provided as per RFC 2616.

8.5.6 Response message body

A response body can be provided as per RFC 2616.

8.5.7 Response status

Table 34 describes the HTTP status codes that occur when deleting a data object using CDMI.

Table 34 — HTTP status codes - Delete a data object using CDMI

HTTP status	Description
`204 No Content`	The data object was successfully deleted.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.
`409 Conflict`	The operation conflicts with a non-CDMI access protocol lock or has caused a state transition error on the server, or the data object cannot be deleted.

8.5.8 Example

EXAMPLE DELETE to the data object URI:

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

The following shows the response.

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

9 Container object resource operations using CDMI

9.1 Overview

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

Containers are addressed in CDMI in two ways:

- by name (e.g., http://cloud.example.com/container/); and
- by object ID (e.g., http://cloud.example.com/cdmi_objectid/00007ED900104E1D14771DC67C27BF8B/).

Every container object has a single, globally-unique object ID that remains constant for the life of the object. Each container object may also have one or more URI addresses that allow the container object to be accessed. Following the URI conventions for hierarchical paths, container URIs shall consist of one or more container names that are separated by forward slashes ("/") and that end with a forward slash ("/").

If a request is performed against an existing container resource and the trailing slash at the end of the URI is omitted, the server shall respond with an HTTP status code of `301 Moved Permanently`. In addition, a Location header containing the URI with the trailing slash added shall be returned.

If a CDMI request is performed to create a new container resource and the trailing slash at the end of the URI is omitted, the server shall respond with an HTTP status code of `400 Bad Request`.

Non-CDMI requests to create a container resource shall include the trailing slash at the end of the URI; otherwise, the request shall be considered a request to create a data object.

Containers may also be nested.

EXAMPLE 1 The following URI represents a nested container:

http://cloud.example.com/container/subcontainer/

A nested container has a parent container object, shall be included in the children field of the parent container object, and shall inherit data system metadata and ACLs from its parent container.

This model allows direct mapping between CDMI-managed cloud storage and file systems (e.g., NFSv4 or WebDAV). If a CDMI container object is exported as a file system, then the file system may make the CDMI metadata accessible via file system-specific mechanisms. As files and directories are created by the file system, they become visible through the CDMI interface acting as a data path. The mapping between file system constructs and CDMI data objects, container objects, and metadata is outside the scope of this International Standard.

Individual fields within a container object may be accessed by specifying the field name after a question mark "?" appended to the end of the container object URI.

EXAMPLE 2 The following URI returns 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.

EXAMPLE 3 The following URI returns the first three children from the children field:

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

Children ranges are specified in a way that is similar to byte ranges as per Section 14.35.1 of RFC 2616. A client can determine the number of children present by requesting the childrenrange field without requesting a range of children.

A list of fields, separated by a semicolon ";" may be specified, allowing multiple fields to be accessed in a single request.

EXAMPLE 4 The following URI would return the children and metadata fields in the response body:

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

If read access to any of the requested fields is not permitted by the object ACL, only the permitted fields shall be returned. If no requested fields are permitted to be read, an HTTP status code of `403 Forbidden` shall be returned to the client.

If write access to any of the requested fields is not permitted by the object ACL, no updates shall be performed, and an HTTP status code of `403 Forbidden` shall be returned to the client.

When a client includes deserialized fields that are not defined in this International Standard, these fields shall be stored as part of the object.

9.1.1 Container metadata

The following optional data system metadata may be provided (see Table 35).

Table 35 — Container metadata

Metadata name	Type	Description	Requirement
cdmi_assignedsize	JSON string	The number of bytes that is reported via exported protocols (e.g., the device may be thin provisioned). This number may limit cdmi_size.	Optional

Container metadata may also include arbitrary user-supplied metadata, storage system metadata, and data system metadata as described in Clause 16.

9.1.2 Reserved container names

This International Standard defines reserved container names that shall not be used when creating new containers. These container names are reserved for use by this International Standard, and if an attempt is made to create or delete them, an HTTP status code of `400 Bad Request` shall be returned to the client.

The reserved container names include

- cdmi_objectid,
- cdmi_domains,
- cdmi_capabilities,
- cdmi_snapshots, and
- cdmi_versions.

As additional names may be added in future versions of this International Standard, server implementations shall prevent the creation of user-defined containers if the container name starts with "cdmi_".

9.1.3 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 container 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 shall be managed the same way that conflicting writes via one path are managed, via the principle of eventual consistency (see 9.2).

9.1.4 Container object representations

The representations in this clause are shown using JSON notation. Both clients and servers shall support UTF-8 JSON representation. The request and response body JSON fields may be specified or returned in any order, with the exception that, if present, for container objects, the childrenrange and children fields shall appear last and in that order.

9.2 Create a container object using CDMI

9.2.1 Synopsis

To create a new container object, the following request shall be performed:

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

Where:

- <root URI> is the path to the CDMI cloud.
- <ContainerName> is zero or more intermediate container objects that already exist, with one slash (i.e., "/") between each pair of container object names.
- <NewContainerName> is the name specified for the container object to be created.

After it is created, the container object shall also be accessible at <root URI>/cdmi_objectid/<objectID>/.

9.2.2 Delayed completion of create

In response to a create operation for a container object, the server may return an HTTP status code of `202 Accepted` to indicate that the object is in the process of being created. This response is useful for long-running operations (e.g., deserializing a source data object to create a large container object hierarchy). Such a response has the following implications.

- The server shall return a Location header with an absolute URI to the object to be created along with an HTTP status code of `202 Accepted`.
- With an HTTP status code of `202 Accepted`, the server implies that the following checks have passed:
  - user authorization for creating the container object;
  - user authorization for read access to any source object for move, copy, serialize, or deserialize; and
  - availability of space to create the container object or at least enough space to create a URI to report an error.
- A client might not be able to immediately access the created object, e.g., due to delays resulting from the implementation's use of eventual consistency.

The client performs GET operations to the URI to track the progress of the operation. In response, 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 contains the percentage that the accepted PUT has completed (0 to 100). GET does not return any children for the container 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.

9.2.3 Capabilities

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

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

9.2.4 Request headers

The HTTP request headers for creating a container object using CDMI are shown in Table 36.

Table 36 — Request headers - Create a container object using CDMI

Header	Type	Description	Requirement
Accept	Header string	"application/cdmi-container" or a consistent value as per clause 5.13.2 "Content-type negotiation"	Optional
Content-Type	Header string	"application/cdmi-container"	Mandatory
X-CDMI-Specification-Version	Header string	A comma-separated list of versions that the client supports, for example, "1.1, 1.5, 2.0"	Mandatory

9.2.5 Request message body

The request message body fields for creating a container object using CDMI are shown in Table 37.

Table 37 — Request message body - Create a container object using CDMI (Sheet 1 of 2)

Field name	Type	Description	Requirement
metadata	JSON object	Metadata for the container object • If this field is included when deserializing, serializing, copying, or moving a container 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 container object, the metadata from the source URI shall be used. • If this field is included when creating a new container 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 container object by specifying a value, an empty JSON object (i.e., "{}") shall be assigned as the field value. • This field shall not be included when referencing a container object.	Optional
domainURI	JSON string	URI of the owning domain • If different from the parent domain, the user shall have the "cross-domain" privilege (see cdmi_member_privileges in Table 63 "Required settings for domain member user objects"). • If not specified, the parent domain shall be used.	Optional
exports	JSON object	A structure for each protocol enabled for this container object (see Clause 13). This field shall not be included when referencing a container object.	Optional
deserialize	JSON string	URI of a data object that shall be deserialized to create the new container object, including all child objects inside the source serialized data object (see Clause 15). When deserializing a container object, any exported protocols from the original serialized container object are not applied to the newly created container objects.	Optional[a]

[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored. If more than one of these fields is supplied, the server shall respond with an HTTP status code of `400 Bad Request`.

Table 37 — Request message body - Create a container object using CDMI (Sheet 2 of 2)

Field name	Type	Description	Requirement
copy	JSON string	URI of a source container object that shall be copied into the new destination container object. • If the destination container object URI and the copy source object URI both do not specify individual fields, the destination container object shall be a complete copy of the source container object, including all child objects under the source container object. • If the destination container object URI or the copy source object URI specifies individual fields, only the fields specified shall be used to create the destination container object. If specified fields are not present in the source, default field values shall be used. • If the destination container object URI and the copy source object URI both specify fields, an HTTP status code of `400 Bad Request` shall be returned to the client. When copying a container object, exported protocols are not preserved across the copy. If there are insufficient permissions to read the container object at the source URI or create the container object at the destination URI, or if the read operation fails, the copy shall return an HTTP status code of `400 Bad Request`, and the destination container object shall not be created.	Optional[a]
move	JSON string	URI of an existing local or remote container object (source URI) that shall be relocated, along with all child objects, to the URI specified in the PUT. The contents of the container object and all children, including the object ID, shall be preserved by a move, and the container object and all children of the source URI shall be removed after the objects at the destination have been successfully created. If there are insufficient permissions to read the objects at the source URI, write the objects at the destination URI, or delete the objects at the source URI, or if any of these operations fail, the move shall return an HTTP status code of `400 Bad Request`, and the source and destination are left unchanged.	Optional[a]
reference	JSON string	URI of a container object that shall be redirected to by a reference. If other fields are supplied when creating a reference, the server shall respond with an HTTP status code of `400 Bad Request`.	Optional[a]
deserializevalue	JSON string	A container object serialized as specified in Clause 15 and encoded using base 64 encoding rules described in RFC 4648. The object ID of the serialized container object shall match the object ID of the destination container object.	Optional[a]

[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored. If more than one of these fields is supplied, the server shall respond with an HTTP status code of `400 Bad Request`.

9.2.6 Response headers

The HTTP response headers for creating a container object using CDMI are shown in Table 38.

Table 38 — Response headers - Create a container object using CDMI

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, e.g., "1.1". If the server does not support any of the versions that the client supports, the server shall return an HTTP status code of `400 Bad Request`.	Mandatory
Location	Header string	When an HTTP status code of `202 Accepted` is returned, the server shall respond with the absolute URL of the object that is in the process of being created.	Conditional

9.2.7 Response message body

The response message body fields for creating a container object using CDMI are shown in Table 39.

Table 39 — Response message body - Create a container object using CDMI (Sheet 1 of 2)

Field name	Type	Description	Requirement
objectType	JSON string	"application/cdmi-container"	Mandatory
objectID	JSON string	Object ID of the object	Mandatory
objectName	JSON string	Name of the object	Mandatory
parentURI	JSON string	URI for the parent object Appending the objectName to the parentURI shall always produce a valid URI for the object.	Mandatory
parentID	JSON string	Object ID of the parent container object	Mandatory
domainURI	JSON string	URI of the owning domain	Mandatory
capabilitiesURI	JSON string	URI to the capabilities for the object	Mandatory
completionStatus	JSON string	A string indicating if the object is still in the process of being created or updated by another operation, and after that operation is complete, indicates if it was successfully created or updated or if an error occurred. The value shall be the string "Processing", the string "Complete", or an error string starting with the value "Error".	Mandatory
[a]Returned only if present.			

Table 39 — Response message body - Create a container object using CDMI (Sheet 2 of 2)

Field name	Type	Description	Requirement
percentComplete	JSON string	• When the value of completionStatus is "Processing", this field, if provided, shall indicate the percentage of completion as a numeric integer value from 0 through 100. • When the value of completionStatus is "Complete", this field, if provided, shall contain the value "100". • When the value of completionStatus is "Error", this field, if provided, may contain any integer value from 0 through 100.	Optional
metadata	JSON object	Metadata for the container object. • This field includes any user and data system metadata that is specified in the request body metadata field, along with storage system metadata that is generated by the cloud storage system. • See Clause 16 for a further description of metadata.	Mandatory
exports	JSON object	A structure for each protocol that is enabled for this container object. See Clause 13.	Optional[a]
snapshots	JSON array of JSON strings	URIs of the snapshot container objects. See Clause 14.	Optional[a]
childrenrange	JSON string	The children of the container expressed as a range. If a range of children is requested, this field indicates the children returned as a range. This field should not be returned in the response message body that is associated with a copy, move, deserialize, or deserialize value operation.	Optional
children	JSON array of JSON strings	Names of the children objects in the container object. Child container objects end with "/". This field should not be returned in the response message body that is associated with a copy, move, deserialize, or deserialize value operation.	Optional
[a]Returned only if present.			

9.2.8 Response status

Table 40 describes the HTTP status codes that occur when creating a container object using CDMI.

Table 40 — HTTP status codes - Create a container object using CDMI

HTTP status	Description
`201 Created`	The new container object was created.
`202 Accepted`	The container is in the process of being created. The CDMI client should monitor the completionStatus and percentComplete fields to determine the current status of the operation.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.
`409 Conflict`	The operation conflicts with a non-CDMI access protocol lock or has caused a state transition error on the server.

9.2.9 Examples

EXAMPLE 1 Create a new container with no metadata:

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

{
}
```

The following shows the response:

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

{
    "objectType": "application/cdmi-container",
    "objectID": "00007ED900104E1D14771DC67C27BF8B",
    "objectName": "MyContainer/",
    "parentURI": "/",
    "parentID": "00007E7F0010128E42D87EE34F5A6560",
    "domainURI": "/cdmi_domains/MyDomain/",
    "capabilitiesURI": "/cdmi_capabilities/container/",
    "completionStatus": "Complete",
    "metadata": {
        ...
    },
    "childrenrange": "",
    "children": []
}
```

EXAMPLE 2 Create a container with metadata:

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

```
{
    "metadata": {
        "Colour": "Yellow"
    }
}
```

The following shows the response.

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

{
    "objectType": "application/cdmi-container",
    "objectID": "00007ED900104E1D14771DC67C27BF8B",
    "objectName": "MyContainer/",
    "parentURI": "/",
    "parentID": "00007E7F0010128E42D87EE34F5A6560",
    "domainURI": "/cdmi_domains/MyDomain/",
    "capabilitiesURI": "/cdmi_capabilities/container/",
    "completionStatus": "Complete",
    "metadata": {
        "Colour": "Yellow",
        ...
    },
    "childrenrange": "",
    "children": []
}
```

EXAMPLE 3 Create a container that is a copy of a container:

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

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

The following shows the response.

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

{
    "objectType": "application/cdmi-container",
    "objectID": "00007ED900104E1D14771DC67C27BF8B",
    "objectName": "MyContainerCopy/",
    "parentURI": "/",
    "parentID": "00007E7F0010128E42D87EE34F5A6560",
    "domainURI": "/cdmi_domains/MyDomain/",
    "capabilitiesURI": "/cdmi_capabilities/container/",
    "completionStatus": "Complete",
    "metadata": {
        "Colour": "Yellow",
        ...
    },
}
```

EXAMPLE 4 Rename a container:

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

```
X-CDMI-Specification-Version: 1.1

{
    "move": "/MyContainer/"
}
```

The following shows the response.

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

{
    "objectType": "application/cdmi-container",
    "objectID": "00007ED900104E1D14771DC67C27BF8B",
    "objectName": "MyContainerRenamed/",
    "parentURI": "/",
    "parentID": "00007E7F0010128E42D87EE34F5A6560",
    "domainURI": "/cdmi_domains/MyDomain/",
    "capabilitiesURI": "/cdmi_capabilities/container/",
    "completionStatus": "Complete",
    "metadata": {
        "Colour": "Yellow",
        ...
    },
}
```

9.3 Read a container object using CDMI

9.3.1 Synopsis

To read all fields from an existing container object, the following request shall be performed:

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

To read one or more requested fields from an existing container object, one of the following requests shall be performed:

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

Where:

- <root URI> is the path to the CDMI cloud.
- <ContainerName> is zero or more intermediate container objects.
- <TheContainerName> is the name specified for the container object 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 container object shall also be accessible at <root URI>/cdmi_objectid/<objectID>/.

9.3.2 Capabilities

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

- 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 object.
- Support for the ability to list the children of an existing container object is indicated by the presence of the cdmi_list_children capability in the specified container object.

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

9.3.3 Request headers

The HTTP request headers for reading a container object using CDMI are shown in Table 41.

Table 41 — Request headers - Read a container object using CDMI

Header	Type	Description	Requirement
Accept	Header string	"application/cdmi-container" or a consistent value as per clause 5.13.2 "Content-type negotiation"	Optional
X-CDMI-Specification-Version	Header string	A comma-separated list of versions that the client supports, e.g., "1.1, 1.5, 2.0"	Mandatory

9.3.4 Request message body

A request body shall not be provided.

9.3.5 Response headers

The HTTP response headers for reading a container object using CDMI are shown in Table 42.

Table 42 — Response headers - Read a container object using CDMI

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, e.g., "1.1". If the server does not support any of the versions that the client supports, the server shall return an HTTP status code of `400 Bad Request`.	Mandatory
Content-Type	Header string	"application/cdmi-container"	Mandatory
Location	Header string	The server shall respond with an absolute URI to which the reference redirects if the object is a reference.	Conditional

9.3.6 Response message body

The response message body fields for reading a container object using CDMI are shown in Table 43.

Table 43 — Response message body - Read a container object using CDMI (Sheet 1 of 3)

Field name	Type	Description	Requirement
objectType	JSON string	"application/cdmi-container"	Mandatory
objectID	JSON string	Object ID of the object	Mandatory
[a]Returned only if present.			

Table 43 — Response message body - Read a container object using CDMI (Sheet 2 of 3)

Field name	Type	Description	Requirement
objectName	JSON string	Name of the object • For objects in a container, the objectName field shall be returned. • For objects not in a container (objects that are only accessible by ID), the objectName field does not exist and shall not be returned.	Conditional
parentURI	JSON string	URI for the parent object • For objects in a container, the parentURI field shall be returned. • For objects not in a container (objects that are only accessible by ID), the parentURI field does not exist and shall not be returned. Appending the objectName to the parentURI shall always produce a valid URI for the object.	Conditional
parentID	JSON string	Object ID of the parent container object • For objects in a container, the parentID field shall be returned. • For objects not in a container (objects that are only accessible by ID), the parentID field does not exist and shall not be returned.	Conditional
domainURI	JSON string	URI of the owning domain	Mandatory
capabilitiesURI	JSON string	URI to the capabilities for the object	Mandatory
completionStatus	JSON string	A string indicating if the object is still in the process of being created or updated by another operation, and after that operation is complete, indicates if it was successfully created or updated or if 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	• When the value of completionStatus is "Processing", this field, if provided, shall indicate the percentage of completion as a numeric integer value from 0 through 100. • When the value of completionStatus is "Complete", this field, if provided, shall contain the value "100". • When the value of completionStatus is "Error", this field, if provided, may contain any integer value from 0 through 100.	Optional
metadata	JSON object	Metadata for the container 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 Clause 16 for a further description of metadata.	Mandatory
exports	JSON object	A structure for each protocol that is enabled for this container object (see Clause 13)	Optional[a]
[a]Returned only if present.			

Table 43 — Response message body - Read a container object using CDMI (Sheet 3 of 3)

Field name	Type	Description	Requirement
snapshots	JSON array of JSON strings	URIs of the snapshot container objects	Optional[a]
childrenrange	JSON string	The children of the container expressed as a range. If a range of children is requested, this field indicates the children returned as a range.	Mandatory
children	JSON array of JSON strings	Names of the children objects in the container object. When a client uses a child name in a request URI or a header URI, the client shall escape reserved characters according to RFC 3986, e.g., a "%" character in a child name shall be replaced with "%25". • Children that are container objects shall have "/" appended to the child name. • Children that are references shall have "?" appended to the child name.	Mandatory

[a]Returned only if present.

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.

9.3.7 Response status

Table 44 describes the HTTP status codes that occur when reading a container object using CDMI.

Table 44 — HTTP status codes - Read a container object using CDMI

HTTP status	Description
`200 OK`	The metadata for the container object is provided in the message body.
`302 Found`	The resource is a reference to another resource.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource 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.

9.3.8 Examples

EXAMPLE 1 GET to the container object URI to read all the fields of the container object:

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

The following shows the response.

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

```
{
    "objectType" : "application/cdmi-container",
    "objectID" : "00007ED900104E1D14771DC67C27BF8B",
    "objectName" : "MyContainer/",
    "parentURI" : "/",
    "parentID" : "00007E7F0010128E42D87EE34F5A6560",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/container/",
    "completionStatus" : "Complete",
    "metadata" : {
        ...
    },
    "exports" : {
    "OCCI/iSCSI": {
        "identifier": "00007E7F00104BE66AB53A9572F9F51E",
        "permissions": [
            "http://example.com/compute/0/",
            "http://example.com/compute/1/"
        ]
    },
        "Network/NFSv4" : {
            "identifier" : "/users",
            "permissions" : "domain"
        },
        "childrenrange" : "0-4",
        "children" : [
            "red",
            "green",
            "yellow",
            "orange/",
            "purple/"
        ]
    }
}
```

EXAMPLE 2 GET to the container object URI to read parentURI and children of the container object:

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

The following shows the response.

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

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

EXAMPLE 3 GET to the container object URI to read children 0..2 and childrenrange of the container object:

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

The following shows the response.

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

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

EXAMPLE 4 GET to the container object by ID to read children 0..2 and childrenrange of the container object:

```
GET /cdmi_objectid/0000706D0010B84FAD185C425D8B537E/?childrenrange;children:0-2
   HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-container
X-CDMI-Specification-Version: 1.1
```

The following shows the response.

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

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

9.4 Update a container object using CDMI

9.4.1 Synopsis

To update some or all fields in an existing container object, the following request shall be performed:

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

To add, update, and remove specific metadata items of an existing container object, the following request shall be performed:

```
PUT <root URI>/<ContainerName>/<TheContainerName>/?metadata:<metadataname>;...
```

Where:

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

The container object shall also be accessible at <root URI>/cdmi_objectid/<objectID>/. An update shall not result in a change to the object ID.

9.4.2 Delayed completion of snapshot

If the creation of a snapshot (see Clause 14) is requested by including a snapshot field in the request message body, the server may return an HTTP status code of `202 Accepted`. Such a response has the following implications:

- With an HTTP status code of `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 object, and
  - availability of space to create the snapshot or at least enough space to create a URI to report an error.
- A client might not be able to immediately access the snapshot, e.g., due to delays resulting from the implementation's 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 field contains either "Processing", "Complete", or an error string starting with the value "Error".
- An optional percentComplete field contains the percentage that the accepted 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.

9.4.3 Capabilities

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

- Support for the ability to modify the metadata of an existing container object is indicated by the presence of the cdmi_modify_metadata capability in the specified container object.
- Support for the ability to snapshot the contents of an existing container object is indicated by the presence of the cdmi_snapshot capability in the specified container object.
- Support for the ability to add an exported protocol to an existing container object is indicated by the presence of the cdmi_export_<protocol> capabilities for the specified container object.

9.4.4 Request headers

The HTTP request headers for updating a container object using CDMI are shown in Table 45.

Table 45 — Request headers - Update a container object using CDMI

Header	Type	Description	Requirement
Content-Type	Header string	"application/cdmi-container"	Mandatory
X-CDMI-Specification-Version	Header string	A comma-separated list of versions that the client supports, e.g., "1.1, 1.5, 2.0"	Mandatory

9.4.5 Request message body

The request message body fields for updating a container object using CDMI are shown in Table 46.

Table 46 — Request message body - Update a container object using CDMI (Sheet 1 of 2)

Field name	Type	Description	Requirement
metadata	JSON object	Metadata for the container object. If present, the new metadata specified replaces the existing object metadata. If individual metadata items are specified in the URI, only those items are replaced; other items are preserved. See Clause 16 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 (see cdmi_member_privileges in Table 63 "Required settings for domain member user objects"). • If not specified, the existing domain shall be preserved.	Optional
snapshot	JSON string	Name of the snapshot to be taken. This is not a URL, but rather, the final component of the absolute URL where the snapshot will exist when the snapshot operation successfully completes. • If a snapshot is added or changed, the PUT operation only returns after the snapshot is added to the snapshot list. • After they are created, snapshots may be accessed as children container objects under the cdmi_snapshots child container object of the container object receiving a snapshot. • When creating a snapshot with the same name as an existing snapshot, the new snapshot will replace the existing snapshot.	Optional
deserialize	JSON string	URI of a container object that shall be deserialized to update an existing container object. The object ID of the serialized container object shall match the object ID of the destination container object. • If the serialized container object does not contain children, the update is applied only to the container object, and any existing children are left as is. • If the serialized container object does contain children, then creates, updates, and deletes are recursively applied for each child, depending on the differences between the provided serialized state and the current state of the child.	Optional[a]

[a] Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored.

Table 46 — Request message body - Update a container object using CDMI (Sheet 2 of 2)

Field name	Type	Description	Requirement
copy	JSON string	URI of a source container object that shall be copied into the existing destination container object. • If the destination container object URI and the copy source object URI both do not specify individual fields, the destination container object shall be replaced with the source container object, including all child objects under the source container object. • If the destination container object URI or the copy source object URI specifies individual fields, only the fields specified shall be used to update the destination container object. If specified fields are not present in the source, these fields shall be ignored. • If the destination container object URI and the copy source object URI both specify fields, an HTTP status code of `400 Bad Request` shall be returned to the client. Note: When copying a container object, exported protocols are not preserved across the copy. If there are insufficient permissions to read the container object at the source URI or create the container object at the destination URI, or if the read operation fails, the copy shall return an HTTP status code of `400 Bad Request`, and the destination container object shall not be updated.	Optional[a]
deserializevalue	JSON Sting	A container object serialized as specified in Clause 15 and encoded using base 64 encoding rules described in RFC 4648. The object ID of the serialized container object shall match the object ID of the destination container object. Otherwise, the server shall return an HTTP status code of `400 Bad Request`. • If the serialized container object does not contain children, the update is applied only to the container object, and any existing children are left as is. • If the serialized container object does contain children, then creates, updates, and deletes are recursively applied for each child, depending on the differences between the provided serialized state and the current state of the children.	Optional[a]
exports	JSON object	A structure for each protocol that is enabled for this container object (see Clause 13). If an exported protocol is added or changed, the PUT operation only returns after the export operation has completed. If not specified, the existing exports shall be preserved.	Optional

[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored.

9.4.6 Response header

The HTTP response header for updating a container object using CDMI is shown in Table 47.

Table 47 — Response header - Update a container object using CDMI

Header	Type	Description	Requirement
Location	Header string	The server shall respond with an absolute URI to which the reference redirects if the object is a reference.	Conditional

9.4.7 Response message body

A response body can be provided as per RFC 2616.

9.4.8 Response status

Table 48 describes the HTTP status codes that occur when updating a container object using CDMI.

Table 48 — HTTP status codes - Update a container object using CDMI

HTTP status	Description
`204 No Content`	The data object content was returned in the response.
`202 Accepted`	The container or snapshot (subcontainer object) is in the process of being created. The CDMI client should montitor the completionStatus and percentComplete fields to determine the current status of the operation.
`302 Found`	The resource is a reference to another resource.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.
`409 Conflict`	The operation conflicts with a non-CDMI access protocol lock or has caused a state transition error on the server.

9.4.9 Examples

EXAMPLE 1 PUT to the container object URI to set new field values:

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

{
    "metadata" : {

    } ,
    "exports" : {
        "OCCI/iSCSI": {
        "identifier": "00007E7F00104BE66AB53A9572F9F51E",
        "permissions": [
            "http://example.com/compute/0/",
            "http://example.com/compute/1/"
        ]
    },
        "Network/NFSv4" : {
            "identifier" : "/users",
            "permissions" : "domain"
```

```
            }
        }
    }
```

The following shows the response.

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

EXAMPLE 2 PUT to the container object URI to set a new exported protocol value:

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

{
    "exports" : {
        "OCCI/iSCSI" : {
            "identifier" : "00007ED900104E1D14771DC67C27BF8B",
            "permissions" : "00007E7F00104EB781F900791C70106C"
        } ,
        "Network/NFSv4" : {
            "identifier" : "/users",
            "permissions" : "domain"
        }
    }
}
```

The following shows the response.

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

9.5 Delete a container object using CDMI

9.5.1 Synopsis

To delete an existing container object, including all contained children and snapshots, the following request shall be performed:

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

Where:

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

The object shall also be accessible at <root URI>/cdmi_objectid/<objectID>/.

9.5.2 Capability

The following capability describes the supported operations that may be performed when deleting an existing container object:

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

9.5.3 Request header

The HTTP request header for deleting a container object using CDMI is shown in Table 49.

Table 49 — Request header - Delete a container object using CDMI

Header	Type	Description	Requirement
X-CDMI-Specification-Version	Header string	A comma-separated list of versions that the client supports, e.g., "1.1, 1.5, 2.0"	Mandatory

9.5.4 Request message body

A request body can be provided as per RFC 2616.

9.5.5 Response headers

Response headers can be provided as per RFC 2616.

9.5.6 Response message body

A response body can be provided as per RFC 2616.

9.5.7 Response status

Table 50 describes the HTTP status codes that occur when deleting a container object using CDMI.

Table 50 — HTTP status codes - Delete a container object using CDMI

HTTP status	Description
`204 No Content`	The container object was successfully deleted.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.
`409 Conflict`	The operation conflicts with a non-CDMI access protocol lock or has caused a state transition error on the server.

9.5.8 Example

EXAMPLE DELETE to the container object URI:

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

The following shows the response.

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

9.6 Create (POST) a new data object using CDMI

9.6.1 Synopsis

To create a new data object in a specified container where the name of the data object is a server-assigned object identifier, the following request shall be performed:

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

To create a new data object where the data object does not belong to a container and is only accessible by ID (see 5.8 "Object model for CDMI"), the following request shall be performed:

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

Where:

- <root URI> is the path to the CDMI cloud.
- <ContainerName> is zero or more intermediate container objects that already exist, with one slash (i.e., "/") between each pair of container object names.

If created in "/cdmi_objectid/", the data object shall be accessible at <root URI>/cdmi_objectid/<objectID>.

If created in a container, the data object shall be accessible as a child of the container with a server-assigned name, and shall also be accessible at <root URI>/cdmi_objectid/<objectID>.

9.6.2 Delayed completion of create

In response to a create operation for a data object, the server may return an HTTP status code of `202 Accepted` to indicate that the object is in the process of being created. This response is useful for long-running operations (e.g., copying a large data object from a source URI). Such a response has the following implications.

- The server shall return a Location header with an absolute URI to the object to be created along with an HTTP status code of `202 Accepted`.
- With an HTTP status code of `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; and
  - availability of space to create the object or at least enough space to create a URI to report an error.
- A client might not be able to immediately access the created object, e.g., due to delays resulting from the implementation's use of eventual consistency.

The client performs GET operations to the URI to track the progress of the operation. In response, 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 contains the percentage that the Accepted POST 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.

9.6.3 Capabilities

The following capabilities describe the supported operations that can be performed when creating a new data object by ID in "/cdmi_objectid/":

- Support for the ability to create data objects through this operation is indicated by the presence of the cdmi_post_dataobject_by_ID system capability.
- If the object being created in "/cdmi_objectid/" is a reference, support for that ability is indicated by the presence of the cdmi_create_reference_by_ID system capability.
- If the new data object being created in "/cdmi_objectid/" is a copy of an existing data object, support for the ability to copy is indicated by the presence of the cdmi_copy_dataobject_by_ID system capability.
- If the new data object being created in "/cdmi_objectid/" is the destination of a move, support for the ability to move the data object to "/cdmi_objectid/" is indicated by the presence of the cdmi_object_move_to_ID system capability.
- If the new data object being created in "/cdmi_objectid/" is the destination of a deserialization operation, support for the ability to deserialize the data object is indicated by the presence of the cdmi_deserialize_dataobject_by_ID system capability.
- If the new data object being created in "/cdmi_objectid/" is the destination of a serialize operation, support for the ability to serialize the data object is indicated by the presence of the cdmi_serialize_dataobject_to_ID, cdmi_serialize_container_to_ID, cdmi_serialize_domain_to_ID, or cdmi_serialize_queue_to_ID system capabilities.

The following capabilities describe the supported operations that can be performed when creating a new data object by ID in a container:

- Support for the ability to create data objects through this operation is indicated by the presence of both the cdmi_post_dataobject and the cdmi_create_dataobject capabilities in the specified container object.
- If the object being created in the parent container object is a reference, support for that ability is indicated by the presence of the cdmi_create_reference capability in the parent container object.
- 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_copy_dataobject capability in the parent container object.
- 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_move_dataobject capability in the parent container object.
- If the new data object is the destination of a deserialize operation, support for the ability to deserialize the the data object is indicated by the presence of the cdmi_deserialize_dataobject capability in the parent container object.
- 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 capabilities in the parent container object.

9.6.4 Request headers

The HTTP request headers for creating a new data object using CDMI are shown in Table 51.

Table 51 — Request headers - Create a new data object using CDMI

Header	Type	Description	Requirement
Accept	Header string	"application/cdmi-object" or a consistent value as per clause 5.13.2 "Content-type negotiation"	Optional
Content-Type	Header string	"application/cdmi-object" or "multipart/mixed" • If multipart/mixed is specified, the body shall consist of at least two MIME parts, where the first part shall contain a body of content-type "application/cdmi-object" and the second and subsequent parts shall contain one or more byte ranges of the value as described in 8.3 "Read a data object using CDMI". • If multiple byte ranges are included and the "Content-Range" header is omitted for a part, the data in the part shall be appended to the data in the preceding part, with the first part having a byte offset of zero.	Mandatory
X-CDMI-Specification-Version	Header string	A comma-separated list of versions that the client supports, e.g., "1.1, 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. If X-CDMI-Partial is present, the completionStatus field in the response body shall be set to "Processing". X-CDMI-Partial works across CDMI and non-CDMI operations.	Optional

9.6.5 Request message body

The request message body fields for creating a new data object using CDMI are shown in Table 52.

Table 52 — Request message body - Create a new data object using CDMI (Sheet 1 of 3)

Field name	Type	Description	Requirement
mimetype	JSON string	MIME type of the data contained within the value field of the data object • This field may be included when creating by value or when deserializing, serializing, copying, or moving a data object. • If this field is not included and multi-part MIME is not being used, the value of "text/plain" shall be assigned as the field value. • If this field is not included and multi-part MIME is being used, the value of the "Content-Type" header of the second MIME part shall be assigned as the field value. • This field shall be stored as part of the data object. • This field shall not be included when creating a reference. • This mimetype field value shall be converted to lower case before being stored.	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 (i.e., "{}") 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 • Any domain may be specified, and the "cross_domain" privilege is not required (see cdmi_member_privileges in Table 63 "Required settings for domain member user objects"). • If not specified, the root domain "/cdmi_domains/" shall be used.	Optional
deserialize	JSON string	URI of a data object that shall be deserialized to create the new data object	Optional[a]
serialize	JSON string	URI of a CDMI object that shall be serialized into the new data object	Optional[a]

[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored. If more than one of these fields is supplied, the server shall respond with an HTTP status code of `400 Bad Request`.

Table 52 — Request message body - Create a new data object using CDMI (Sheet 2 of 3)

Field name	Type	Description	Requirement
copy	JSON string	URI of a data object or queue object that shall be copied into the new data object	Optional[a]
move	JSON string	URI of a data object or queue object value that shall be copied into the new data object. The data object or queue object value at the source URI shall be removed upon the successful completion of the copy.	Optional[a]
reference	JSON string	URI of a data object that shall be redirected to by a reference. If other fields are supplied when creating a reference, the server shall respond with an HTTP status code of `400 Bad Request`.	Optional[a]
deserializevalue	JSON string	A data object serialized as specified in Clause 15 and encoded using base 64 encoding rules described in RFC 4648. • If multi-part MIME is being used and this field contains the value of the MIME boundary parameter, the contents of the second MIME part shall be assigned as the field value. • If the serialized data object in the second MIME part does not include a value field, the contents of the third MIME part shall be assigned as the field value of the value field.	Optional[a]

[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored. If more than one of these fields is supplied, the server shall respond with an HTTP status code of `400 Bad Request`.

Table 52 — Request message body - Create a new data object using CDMI (Sheet 3 of 3)

Field name	Type	Description	Requirement
valuetransferencoding	JSON String	The value transfer encoding used for the container object value. Two value transfer encodings are defined: • "utf-8" indicates that the data object contains a valid UTF-8 string, and it shall be transported as a UTF-8 string in the value field. • "base64" indicates that the data object may contain arbitrary binary sequences, and it shall be transported as a base 64-encoded string in the value field. Setting the contents of the data object value field to any value other than a valid base 64 string shall result in an HTTP status code of `400 Bad Request` being returned to the client. This field shall only be included when creating a data object by value. • If this field is not included and multi-part MIME is not being used, the value of "utf-8" shall be assigned as the field value. • If this field is not included and multi-part MIME is being used, the value of "utf-8" shall be assigned as the field value if the "Content-Type" header of the second and all subsequent MIME parts includes the charset parameter as defined in RFC 2046 of "utf-8" (e.g., ";charset=utf-8"). Otherwise, the value of "base64" shall be assigned as the field value. This field applies only to the encoding of the value when represented in JSON; the "Content-Transfer-Encoding" header of the part specifies the encoding of the value within a multi-part MIME request, as defined in RFC 2045. This field shall be stored as part of the object.	Optional
value	JSON string	The data object value • If this field is not included and multi-part MIME is not being used, an empty JSON string (i.e., "") shall be assigned as the field value. • If this field is not included and multi-part MIME is being used, the contents of the second MIME part shall be assigned as the field value. • If the valuetransferencoding field indicates UTF-8 encoding, the value shall be a UTF-8 string escaped using the JSON escaping rules described in RFC 4627. • If the valuetransferencoding field indicates base 64 encoding, the value shall be first encoded using the base 64 encoding rules described in RFC 4648.	Optional[a]

[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored. If more than one of these fields is supplied, the server shall respond with an HTTP status code of `400 Bad Request`.

9.6.6 Response headers

The HTTP response headers for creating a new data object using CDMI are shown in Table 53.

Table 53 — Response headers - Create a new data object using CDMI

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, e.g., "1.1". If the server does not support any of the versions that the client supports, the server shall return an HTTP status code of `400 Bad Request`.	Mandatory
Location	Header string	The unique absolute URI 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 URI in the form: http://host:port/<root URI>/<ContainerName>/<ObjectID> or https://host:port/<root URI>/<ContainerName>/<ObjectID>.	Mandatory

9.6.7 Response message body

The response message body fields for creating a new data object using CDMI are shown in Table 54.

Table 54 — Response message body - Create a new data object using CDMI (Sheet 1 of 2)

Field name	Type	Description	Requirement
objectType	JSON string	"application/cdmi-object"	Mandatory
objectID	JSON string	Object ID of the object	Mandatory
objectName	JSON string	Name of the object • For objects in a container, the objectName field shall be returned. • For objects not in a container (objects that are only accessible by ID), the objectName field does not exist and shall not be returned.	Conditional
parentURI	JSON string	URI for the parent object • For objects in a container, the parentURI field shall be returned. • For objects not in a container (objects that are only accessible by ID), the parentURI field does not exist and shall not be returned. Appending the objectName to the parentURI shall always produce a valid URI for the object.	Conditional
parentID	JSON string	Object ID of the parent container object • For objects in a container, the parentID field shall be returned. • For objects not in a container (objects that are only accessible by ID), the parentID field does not exist and shall not be returned.	Conditional

Table 54 — Response message body - Create a new data object using CDMI (Sheet 2 of 2)

Field name	Type	Description	Requirement
domainURI	JSON string	URI of the owning domain	Mandatory
capabilitiesURI	JSON string	URI to the capabilities for the object	Mandatory
completionStatus	JSON string	A string indicating if the object is still in the process of being created or updated by another operation, and after that operation is complete, indicates if it was successfully created or updated or if 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	• When the value of completionStatus is "Processing", this field, if provided, shall indicate the percentage of completion as a numeric integer value from 0 through 100. • When the value of completionStatus is "Complete", this field, if provided, shall contain the value "100". • When the value of completionStatus is "Error", this field, if provided, may contain any integer 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 Clause 16 for a further description of metadata.	Mandatory

9.6.8 Response status

Table 55 describes the HTTP status codes that occur when creating a new data object using CDMI.

Table 55 — HTTP status codes - Create a new data object using CDMI

HTTP status	Description
`201 Created`	The new data object was created.
`202 Accepted`	The data object is in the process of being created. The CDMI client should monitor the completionStatus and percentComplete fields to determine the current status of the operation.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.
`409 Conflict`	The operation conflicts with a non-CDMI access protocol lock or has caused a state transition error on the server.

9.6.9 Examples

EXAMPLE 1 POST to the container object 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: 1.1

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

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

The following shows the response.

```
HTTP/1.1 201 Created
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.1
Location: http://cloud.example.com/MyContainer/00007ED900104E1D14771DC67C27BF8B

{
    "objectType" : "application/cdmi-object",
    "objectID" : "00007ED900104E1D14771DC67C27BF8B",
    "objectName" : "00007ED900104E1D14771DC67C27BF8B",
    "parentURI" : "/MyContainer/",
    "parentID" : "00007ED900104E1D14771DC67C27BF8B",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/dataobject/",
    "completionStatus" : "Complete",
    "mimetype" : "text/plain",
    "metadata" : {
        ...
    }
}
```

EXAMPLE 2 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: 1.1

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

The following shows the response.

```
HTTP/1.1 201 Created
Location: http://cloud.example.com/cdmi_objectid/00007ED900104E1D14771DC67C27BF8B
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.1

{
    "objectType": "application/cdmi-object",
    "objectID": "00007ED900104E1D14771DC67C27BF8B",
    "domainURI": "/cdmi_domains/MyDomain/",
    "capabilitiesURI": "/cdmi_capabilities/dataobject/",
    "completionStatus": "Complete",
    "mimetype": "text/plain",
```

```
    "metadata": {
        "cdmi_acl": [
            {
                "acetype": "ALLOW",
                "identifier": "OWNER@",
                "aceflags": "NO_FLAGS",
                "acemask": "ALL_PERMS"
            }
        ],
        ...
    }
}
```

EXAMPLE 3 POST to the object ID URI the data object fields and binary contents using multi-part MIME:

```
POST /cdmi_objectid/ HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
Content-Type: multipart/mixed; boundary=gc0p4Jq0M2Yt08j34c0p
X-CDMI-Specification-Version: 1.1

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/cdmi-object

{
  "domainURI": "/cdmi_domains/MyDomain/",
  "metadata": {
    "colour": "blue"
  }
}

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary

<37 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p--
```

The following shows the response:

```
HTTP/1.1 201 Created
Location: http://cloud.example.com/cdmi_objectid/00007ED90010C2414303B5C6D4F83170
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.1

{
  "objectType": "application/cdmi-object",
  "objectID": "00007ED90010C2414303B5C6D4F83170",
  "domainURI": "/cdmi_domains/MyDomain/",
  "capabilitiesURI": "/cdmi_capabilities/dataobject/",
  "completionStatus": "Complete",
  "mimetype": "application/octet-stream",
  "metadata": {
    "cdmi_size": "37",
    "colour": "blue",
    ...
  }
}
```

9.7 Create (POST) a new queue object using CDMI

9.7.1 Synopsis

To create a new queue object (see Clause 11) in a specified container where the name of the queue object is a server-assigned object identifier, the following request shall be performed:

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

To create a new queue object where the queue object does not belong to a container and is only accessible by ID (see 5.8 "Object model for CDMI"), the following request shall be performed:

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

Where:

- <root URI> is the path to the CDMI cloud.
- <ContainerName> is zero or more intermediate container objects that already exist, with one slash (i.e., "/") between each pair of container object names.

If created in "/cdmi_objectid/", the queue object shall be accessible at <root URI>/cdmi_objectid/<objectID>.

If created in a container, the queue object shall be accessible as a child of the container with a server-assigned name, and shall also be accessible at <root URI>/cdmi_objectid/<objectID>.

9.7.2 Delayed completion of create

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

- The server shall return a Location header with an absolute URI to the object to be created along with an HTTP status code of `202 Accepted`.
- With an HTTP status code of `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; and
  - availability of space to create the object or at least enough space to create a URI to report an error.
- A client might not be able to immediately access the created object, e.g., due to delays resulting from the implementation's use of eventual consistency.

The client performs GET operations to the URI to track the progress of the operation. In response, 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 contains the percentage that the accepted POST 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.

9.7.3 Capabilities

The following capabilities describe the supported operations that can be performed when creating a new queue object by ID in "/cdmi_objectid/":

- Support for the ability to create queue objects through this operation is indicated by the presence of the cdmi_post_queue_by_ID system capability.
- If the object being created in "/cdmi_objectid/" is a reference, support for that ability is indicated by the presence of the cdmi_create_reference_by_ID system capability.
- If the new queue object being created in "/cdmi_objectid/" is a copy of an existing queue object, support for the ability to copy is indicated by the presence of the cdmi_copy_queue_by_ID system capability.
- If the new queue object being created in "/cdmi_objectid/" is the destination of a move, support for the ability to move the data object to "/cdmi_objectid/" is indicated by the presence of the cdmi_object_move_to_ID system capability.
- If the new queue object being created in "/cdmi_objectid/" is the destination of a deserialization operation, support for the ability to deserialize the data object is indicated by the presence of the cdmi_deserialize_queue_by_ID system capability.
- If the new data object is being created in "/cdmi_objectid/", support for the ability to create the value of the new data object in specified byte ranges is indicated by the presence of the "cdmi_create_value_range_by_ID" system capability.
- If the new data object is being created in a container object, support for the ability to create the value of the new data object in specified byte ranges is indicated by the presence of the "cdmi_create_value_range" capability in the parent container.

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

- Support for the ability to create queue objects through this operation is indicated by the presence of both the cdmi_post_queue and cdmi_create_queue capabilities in the specified container object.
- If the object being created in the parent container object is a reference, support for that ability is indicated by the presence of the cdmi_create_reference capability in the parent container object.
- 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 object.
- 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 object.
- If the new queue object is the destination of a deserialize operation, support for the ability to deserialize the the queue object is indicated by the presence of the cdmi_deserialize_queue capability in the parent container object.

9.7.4 Request headers

The HTTP request headers for creating a new queue object using CDMI are shown in Table 56.

Table 56 — Request headers - Create a new queue object using CDMI

Header	Type	Description	Requirement
Accept	Header string	"application/cdmi-queue" or a consistent value as per clause 5.13.2 "Content-type negotiation"	Optional
Content-Type	Header string	"application/cdmi-queue"	Mandatory
X-CDMI-Specification-Version	Header string	A comma-separated list of versions that the client supports, e.g., "1.1, 1.5, 2.0"	Mandatory
Content-Range	Header string	A valid ranges-specifier (see RFC 2616 Section 14.35.1)	Optional

9.7.5 Request message body

The request message body fields for creating a new queue object using CDMI are shown in Table 57.

Table 57 — Request message body - Create a new queue object using CDMI (Sheet 1 of 2)

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 (i.e., "{}") 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 • Any domain may be specified, and the "cross_domain" privilege is not required (see cdmi_member_privileges in Table 63 "Required settings for domain member user objects"). • If not specified, the root domain "/cdmi_domains/" shall be used.	Optional
deserialize	JSON string	URI of a data object that will be deserialized to create the new queue object	Optional[a]
copy	JSON string	URI of a queue object that will be copied into the new queue object	Optional[a]
[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored. If more than one of these fields is supplied, the server shall respond with an HTTP status code of `400 Bad Request`.			

Table 57 — Request message body - Create a new queue object using CDMI (Sheet 2 of 2)

Field name	Type	Description	Requirement
move	JSON string	URI of a 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[a]
reference	JSON string	URI of a queue object that shall be redirected to by a reference. If other fields are supplied when creating a reference, the server shall respond with an HTTP status code of `400 Bad Request`.	Optional[a]
deserializevalue	JSON string	A queue object serialized as specified in Clause 15 and encoded using base 64 encoding rules described in RFC 4648	Optional[a]
[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored. If more than one of these fields is supplied, the server shall respond with an HTTP status code of `400 Bad Request`.			

9.7.6 Response headers

The response headers for creating a new queue object using CDMI are shown in Table 58.

Table 58 — Response headers - Create a new queue object using CDMI

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, e.g., "1.1". If the server does not support any of the versions that the client supports, the server shall return an HTTP status code of `400 Bad Request`.	Mandatory
Location	Header string	The unique absolute URI 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 URI in the form: http://host:port/<root URI>/<ContainerName>/<ObjectID> or https://host:port/<root URI>/<ContainerName>/<ObjectID>.	Mandatory

9.7.7 Response message body

The response message body fields for creating a new queue object using CDMI are shown in Table 59.

Table 59 — Response message body - Create a new queue object using CDMI (Sheet 1 of 3)

Field name	Type	Description	Requirement
objectType	JSON string	"application/cdmi-queue"	Mandatory
objectID	JSON string	Object ID of the object	Mandatory

Table 59 — Response message body - Create a new queue object using CDMI (Sheet 2 of 3)

Field name	Type	Description	Requirement
objectName	JSON string	Name of the object • For objects in a container, the objectName field shall be returned. • For objects not in a container (objects that are only accessible by ID), the objectName field does not exist and shall not be returned.	Conditional
parentURI	JSON string	URI for the parent object • For objects in a container, the parentURI field shall be returned. • For objects not in a container (objects that are only accessible by ID), the parentURI field does not exist and shall not be returned. Appending the objectName to the parentURI shall always produce a valid URI for the object.	Conditional
parentID	JSON string	Object ID of the parent container object • For objects in a container, the parentID field shall be returned. • For objects not in a container (objects that are only accessible by ID), the parentID field does not exist and shall not be returned.	Conditional
domainURI	JSON string	URI of the owning domain	Mandatory
capabilitiesURI	JSON string	URI to the capabilities for the object	Mandatory
completionStatus	JSON string	A string indicating if the object is still in the process of being created or updated by another operation, and after that operation is complete, indicates if it was successfully created or updated or if 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	• When the value of completionStatus is "Processing", this field, if provided, shall indicate the percentage of completion as a numeric integer value from 0 through 100. • When the value of completionStatus is "Complete", this field, if provided, shall contain the value "100". • When the value of completionStatus is "Error", this field, if provided, may contain any integer value from 0 through 100.	Optional

Table 59 — Response message body - Create a new queue object using CDMI (Sheet 3 of 3)

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 Clause 16 for a further description of metadata.	Mandatory
queueValues	JSON string	The range of designators for enqueued values. Every enqueued value shall be assigned a unique, monotonically-incrementing positive integer designator, starting from 0. If no values are enqueued, an empty string shall be returned. If values are enqueued, the lowest designator, followed by a hyphen ("-"), followed by the highest designator shall be returned.	Mandatory

9.7.8 Response status

Table 60 describes the HTTP status codes that occur when creating a new queue object using CDMI.

Table 60 — HTTP status codes - Create a new queue object using CDMI

HTTP status	Description
`201 Created`	The new queue object was created.
`202 Accepted`	The queue object is in the process of being created. The CDMI client should monitor the completionStatus and percentComplete fields to determine the current status of the operation.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.
`409 Conflict`	The operation conflicts with a non-CDMI access protocol lock or could cause a state transition error on the server.

9.7.9 Example

EXAMPLE POST to the container object URI the queue object contents:

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

{
}
```

The following shows the response.

```
HTTP/1.1 201 Created
Content-Type: application/cdmi-queue
X-CDMI-Specification-Version: 1.1
Location: http://cloud.example.com/MyContainer/00007ED900104E1D14771DC67C27BF8B

{
    "objectType" : "application/cdmi-queue",
    "objectID" : "00007ED900104E1D14771DC67C27BF8B",
```

```
    "objectName" : "00007ED900104E1D14771DC67C27BF8B",
    "parentURI" : "/MyContainer/",
    "parentID" : "00007ED900104E1D14771DC67C27BF8B",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/queue/",
    "completionStatus" : "Complete",
    "metadata" : {
        ...
    },
    "queueValues" : ""
}
```

Section IV

CDMI Advanced

10 Domain object resource operations using CDMI

10.1 Overview

Domain objects represent the concept of administrative ownership of stored data within a CDMI™ storage system. A cloud service 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.

Domain objects 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.

Domains are addressed in CDMI in two ways:

- by name (e.g., http://cloud.example.com/cdmi_domains/myDomain/), and
- by object ID (e.g., http://cloud.example.com/cdmi_objectid/ 00007ED90010329E642EBFBC8B57E9AD/.

Every domain object has a single, globally-unique object ID that remains constant for the life of the object. Each domain object shall also have one URI address that allows the domain object to be accessed. Following the URI conventions for hierarchical paths, domain URIs shall start with "/cdmi_domains/" and consist of one or more domain names that are separated by forward slashes ("/") and that end with a forward slash ("/").

If a request is performed against an existing domain resource and the trailing slash at the end of the URI is omitted, the server shall respond with an HTTP status code of `301 Moved Permanently`, and a Location header containing the URI with the trailing slash will be added.

If a CDMI request is performed to create a new domain resource and the trailing slash at the end of the URI is omitted, the server shall respond with an HTTP status code of `400 Bad Request`.

Individual fields within a domain object may be accessed by specifying the field name after a question mark "?" appended to the end of the domain object URI.

EXAMPLE 1 The following URI returns just the children field in the response message body:

http://cloud.example.com/cdmi_domains/myDomain/?children

By specifying a range after the children field name, specific ranges of the children field may be accessed.

EXAMPLE 2 The following URI returns the first three children from the children field:

http://cloud.example.com/cdmi_domains/myDomain/?children:0-2

Children ranges are specified in a way that is similar to byte ranges as per Section 14.35.1 of RFC 2616. A client can determine the number of children present by requesting the childrenrange field without requesting a range of children.

A list of fields separated by a semicolon ";" may be specified, allowing multiple fields to be accessed in a single request.

EXAMPLE 3 The following URI would return the children and metadata fields in the response message body:

http://cloud.example.com/cdmi_domains/myDomain/?children;metadata

If read access to any of the requested fields is not permitted by the object ACL, only the permitted fields shall be returned. If no requested fields are permitted to be read, an HTTP status code of `403 Forbidden` shall be returned to the client.

If write access to any of the requested fields is not permitted by the object ACL, no updates shall be performed, and an HTTP status code of `403 Forbidden` shall be returned to the client.

When a client provides or includes deserialization fields that are not defined in this international standard, these fields shall be stored as part of the object.

10.1.1 Domain object metadata

The following domain-specific field shall be present for each domain (see Table 61).

Table 61 — Required metadata for a domain object

Metadata name	Type	Description	Requirement
cdmi_domain_enabled	JSON string	Indicates if the domain is enabled and specified at the time of creation. Values shall 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". • When a domain is disabled, all operations that are performed against URIs that are managed by a disabled domain shall return an HTTP status code of `403 Forbidden`.	Mandatory
cdmi_domain_delete_reassign	JSON string	If the domain is deleted, indicates to which domain the objects that belong to the domain shall be reassigned. • To delete a domain that contains objects, this metadata item shall be present. • If this metadata item is not present or does not contain the URI of a valid domain that is different from the URI of the domain being deleted, an attempt to delete a domain that has objects shall result in an HTTP status code of `400 Bad Request`.	Conditional

Domains may also contain domain-specific data system metadata items as defined in 16.4 "Support for data system metadata" and 16.5 "Support for provided data system metadata". Domain data system metadata shall be inherited to child domain objects.

10.1.2 Domain object summaries

Domain object summaries provide summary measurement information about domain usage and billing. If supported, a domain summary container named "cdmi_domain_summary" shall be present under each domain container. Like any container, the domain summary subcontainer may have an Access Control List (ACL) (see 16.1 "Access control") that restricts access to this information.

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

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.

EXAMPLE 1 If a domain were 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.

EXAMPLE 2 If a domain were 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.

EXAMPLE 3 If a domain were 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 62 shall be present within the contents of each domain summary object, which are in JSON representation.

Table 62 — Contents of domain summary objects (Sheet 1 of 2)

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

Table 62 — Contents of domain summary objects (Sheet 2 of 2)

Metadata name	Type	Description	Requirement
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
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	An ISO 4217 currency code (see ISO 4217:2008) that is followed or preceded by a numeric value and separated by a space, where the numeric value represents the closing charge in the indicated currency for the use of the service associated with the domain over the summary time period	Optional
cdmi_summary_kwhours	JSON string	The sum of energy consumed (in kilowatt hours) by the domain during the summary time period	Optional
cdmi_summary_kwmin	JSON string	The minimum rate at which energy is consumed (in kilowatt hours per hour) by the domain during the summary time period	Optional
cdmi_summary_kwmax	JSON string	The maximum rate at which energy is consumed (in kilowatt hours per hour) by the domain during the summary time period	Optional
cdmi_summary_kwaverage	JSON string	The average rate at which energy is consumed (in kilowatt hours per hour) by the domain during the summary time period	Optional

EXAMPLE 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 vendors to include additional metadata or domain reports beyond the metadata items specified by this International Standard, as long as the field names for those metadata items do not begin with "cdmi_".

10.1.3 Domain object membership

In cloud storage environments, in the same way that 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 using CDMI or through software systems that interface with 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 (see 16.1 "Access control"), but rather to provide a single, unified place to map identities and credentials to principals used by ACLs within the context of a domain (see model described in 10.1.4 "Domain usage in access control"). It also provides a place for authentication mappings to external authentication providers, such as LDAP and Active Directory, 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 has an Access Control List (see 16.1) that restricts access to this information.

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 subcontainers with data objects. Data objects in these subcontainers are treated the same as data objects in the domain membership container, and no meaning is inferred from the subcontainer name. This organization is used to create different access security relationships for groups of user objects and to allow delegation to a common set of members.

Table 63 lists the domain settings that shall be present within each domain member user object.

Table 63 — Required settings for domain member user objects

Metadata name	Type	Description	Requirement
cdmi_member_enabled	JSON string	If true, this field indicates that requests associated with this domain member are allowed. If false, all requests performed by this domain member shall result in an HTTP status code of `403 Forbidden`.	Mandatory
cdmi_member_type	JSON string	This field indicates the type of member record. Values include "user", "group", and "delegation".	Mandatory
cdmi_member_name	JSON string	This field contains the user or group name as presented by the client. This will normally be the standard full name of the principal.	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 shall be present and shall be used to resolve user credentials. As one cannot log in as a group but only as a member of a group, the "group" type member records shall not have credentials.	Optional
cdmi_member_principal	JSON string	This field indicates to which principal name (used in ACLs) the user or group is mapped. If this field is not present, one or more delegations shall be present and shall be used to resolve the principal.	Optional
cdmi_member_privileges	JSON array of JSON strings	This field explicitly confers zero or more special privileges to a user or group. When delegated, privileges are conferred based on the information returned from the external system to which the delegation points. The following privileges are defined: • "administrator". Allows the principal to take ownership of any object/container. • "backup_operator". Bypass regular ACL checks to allow backup and restore of objects and containers, including all associated attributes, metadata, ACLs and ownership. • "cross_domain". Operations specifying a domain other than the domain of the parent object are permitted. Unless this privilege is conferred by the user record or a group (possibly nested) to which the user or group belongs, all attempts to change the domain of objects to a domain other than the parent domain shall fail.	Mandatory
cdmi_member_groups	JSON array of JSON strings	This field contains a JSON array of group names to which the user or group belongs.	Optional

Table 64 lists the domain settings that shall be present within each domain member delegation object.

Table 64 — Required settings for domain member delegation objects

Metadata name	Type	Description	Requirement
cdmi_member_enabled	JSON string	If true, this field indicates that requests associated with this domain member are allowed. If false, all requests performed by this domain member shall result in an HTTP status code of `403 Forbidden`.	Mandatory
cdmi_member_type	JSON string	This field indicates 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 object. External delegations are expressed in the form of ldap:// or ad://.	Mandatory

EXAMPLE 1 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_member_groups" : [
        "users"
    ],
    "cdmi_member_principal" : "jdoe",
    "cdmi_privileges" : [
        "administrator",
        "cross_domain"
    ]
}
```

EXAMPLE 2 An example of a domain membership object for a delegation is as follows:

```
{
    "cdmi_member_enabled" : "true",
    "cdmi_member_type" : "delegation",
    "cdmi_delegation_URI" : "/cdmi_domains/MyDomain/",

}
```

10.1.4 Domain usage in access control

When a transaction is performed against a CDMI object, the associated domain object (i.e., the domain object indicated by the domainURI) specifies the authentication context. The user identity and credentials presented as part of the transaction are compared to the domain membership list to determine if the user is authorized within the domain and to resolve the user's principal. If resolved, the user's principal is 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 that is a delegation in which the delegation URI is set to the URI of the parent domain.

10.1.5 Domain object representations

The representations in this clause are shown using JSON notation. Both clients and servers shall support UTF-8 JSON representation. The request and response body JSON fields may be specified or returned in any order, with the exception that, if present, for domain objects, the childrenrange and children fields shall appear last and in that order.

10.2 Create a domain object using CDMI

10.2.1 Synopsis

To create a new domain object, the following request shall be performed:

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

Where:

- <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.

After it is created, the domain shall also be accessible at <root URI>/cdmi_objectid/<objectID>/.

10.2.2 Capabilities

The following capabilities describe the supported operations that can 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_create_domain capability in the parent domain.
- 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.

10.2.3 Request headers

The HTTP request headers for creating a domain object using CDMI are shown in Table 65.

Table 65 — Request headers - Create a domain object using CDMI

Header	Type	Description	Requirement
Accept	Header string	"application/cdmi-domain" or a consistent value as per clause 5.13.2 "Content-type negotiation"	Optional
Content-Type	Header string	"application/cdmi-domain"	Mandatory
X-CDMI-Specification-Version	Header string	A comma-separated list of versions that the client supports, for example, "1.1, 1.5, 2.0"	Mandatory

10.2.4 Request message body

The request message body fields for creating a domain object using CDMI are shown in Table 66.

Table 66 — Request message body - Create a domain object using CDMI

Field name	Type	Description	Requirement
metadata	JSON object	Metadata for the domain object • If this field is included when deserializing, serializing, copying, or moving a domain 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 domain object, the metadata from the source URI shall be used. • If this field is included when creating a new domain 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 domain object by specifying a value, an empty JSON object (i.e., "{}") 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[a]
move	JSON string	URI of an existing local domain object (source URI) that shall be relocated, along with all child domains, to the URI specified in the PUT. The contents of the domain and all sub-domains, including the object ID, shall be preserved by a move, and the domain and sub-domains of the source URI shall be removed after the objects at the destination have been successfully created. If there are insufficient permissions to read the objects at the source URI, write the objects at the destination URI, or delete the objects at the source URI, or if any of these operations fail, the move shall return an HTTP status code of `400 Bad Request`, and the source and destination are left unchanged.	Optional[a]
deserialize	JSON string	URI of a serialized data object that shall be deserialized to create the new domain, including all child objects inside the source serialized data object	Optional[a]
deserializevalue	JSON string	A domain object serialized as specified in Clause 15 and encoded using base 64 encoding rules described in RFC 4648.	Optional[a]

[a] Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored. If more than one of these fields is supplied, the server shall respond with an HTTP status code of `400 Bad Request`.

10.2.5 Response headers

The HTTP response headers for creating a domain object using CDMI are shown in Table 67.

Table 67 — Response headers - Create a domain object using CDMI

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, e.g., "1.1". If the server does not support any of the versions that the client supports, the server shall return an HTTP status code of `400 Bad Request`.	Mandatory

10.2.6 Response message body

The response message body fields for creating a domain object using CDMI are shown in Table 68.

Table 68 — Response message body - Create a domain object using CDMI

Field name	Type	Description	Requirement
objectType	JSON string	"application/cdmi-domain"	Mandatory
objectID	JSON string	Object ID of the domain	Mandatory
objectName	JSON string	Name of the object	Mandatory
parentURI	JSON string	URI for the parent objectAppending the objectName to the parentURI shall always produce a valid URI for the object.	Mandatory
parentID	JSON string	Object ID of the parent container 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	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 Clause 16 for a further description of metadata.	Mandatory
childrenrange	JSON string	The sub-domains of the domain expressed as a range. If a range of sub-domains is requested, this field indicates the children returned as a range.	Mandatory
children	JSON array of JSON strings	Names of the children domains in the domain. Child containers end with "/".	Mandatory

10.2.7 Response status

Table 69 describes the HTTP status codes that occur when creating a domain object using CDMI.

Table 69 — HTTP status codes - Create a domain object using CDMI

HTTP status	Description
`201 Created`	The new domain object was created.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.
`409 Conflict`	The operation conflicts with a non-CDMI access protocol lock or has caused a state transition error on the server.

10.2.8 Example

EXAMPLE 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: 1.1

"metadata":
{
    "cdmi_domain_enabled": "true"
}
```

The following shows the response.

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

{
    "objectType" : "application/cdmi-domain",
    "objectID" : "00007E7F00104BE66AB53A9572F9F51E",
    "objectName" : "MyDomain/",
    "parentURI" : "/cdmi_domains/",
    "parentID" : "00007E7F0010C058374D08B0AC7B3550",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/domain/",
    "metadata" : {
        "cdmi_domain_enabled": "true",
        "cdmi_authentication_methods": "anonymous, basic",
        ...
    },

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

10.3 Read a domain object using CDMI

10.3.1 Synopsis

To read all fields from an existing domain object, the following request shall be performed:

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

To read one or more requested fields from an existing domain object, one of the following requests shall be performed:

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

Where:

- <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 shall also be accessible at <root URI>/cdmi_objectid/<objectID>/.

10.3.2 Capabilities

The following capabilities describe the supported operations that can 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.

10.3.3 Request headers

The HTTP request headers for reading a domain object using CDMI are shown in Table 70.

Table 70 — Request headers - Read a domain object using CDMI

Header	Type	Description	Requirement
Accept	Header string	"application/cdmi-domain" or a consistent value as per clause 5.13.2 "Content-type negotiation"	Optional
X-CDMI-Specification-Version	Header string	A comma-separated list of versions that the client supports, e.g., "1.1, 1.5, 2.0"	Mandatory

10.3.4 Request message body

A request body shall not be provided.

10.3.5 Response headers

The HTTP response headers for reading a domain object using CDMI are shown in Table 71.

Table 71 — Response headers - Read a domain object using CDMI

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, e.g., "1.1". If the server does not support any of the versions that the client supports, the server shall return an HTTP status code of `400 Bad Request`.	Mandatory
Content-Type	Header string	"application/cdmi-domain"	Mandatory
Location	Header string	The server shall respond with an absolute URI to which the reference redirects if the object is a reference.	Conditional

10.3.6 Response message body

The response message body fields for reading a domain object using CDMI are shown in Table 72.

Table 72 — Response message body - Read a domain object using CDMI

Field name	Type	Description	Requirement
objectType	JSON string	"application/cdmi-domain"	Mandatory
objectID	JSON string	Object ID of the domain	Mandatory
objectName	JSON string	Name of the object	Mandatory
parentURI	JSON string	URI for the parent object	Mandatory
parentID	JSON string	Object ID of the parent container 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	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 Clause 16 for a further description of metadata.	Mandatory
childrenrange	JSON string	The sub-domains of the domain expressed as a range. If a range of sub-domains is requested, this field indicates the children returned as a range.	Mandatory
children	JSON array of JSON strings	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.

10.3.7 Response status

Table 73 describes the HTTP status codes that occur when reading a domain object using CDMI.

Table 73 — HTTP status codes - Read a domain object using CDMI

HTTP status	Description
200 OK	The domain object content was returned in the response.
302 Found	The resource is a reference to another resource.
400 Bad Request	The request contains invalid parameters or field names.
401 Unauthorized	The authentication credentials are missing or invalid.
403 Forbidden	The client lacks the proper authorization to perform this request.
404 Not Found	The resource 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.

10.3.8 Examples

EXAMPLE 1 GET to the domain URI to read all the fields of the domain:

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

The following shows the response.

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

{
    "objectType": "application/cdmi-domain",
    "objectID": "00007E7F00104BE66AB53A9572F9F51E",
    "objectName": "MyDomain/",
    "parentURI": "/cdmi_domains/",
    "parentID": "00007E7F0010C058374D08B0AC7B3550",
    "domainURI": "/cdmi_domains/MyDomain/",
    "capabilitiesURI": "/cdmi_capabilities/domain/",
    "metadata": {
        "cdmi_domain_enabled": "true",
        "cdmi_authentication_methods": "anonymous, basic",
     ...
    },
    "childrenrange": "0-1",
    "children": [
        "cdmi_domain_summary/",
        "cdmi_domain_members/"
    ]
}
```

EXAMPLE 2 GET to the domain URI to read 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: 1.1
```

The following shows the response.

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

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

EXAMPLE 3 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: 1.1
```

The following shows the response.

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

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

10.4 Update a domain object using CDMI

10.4.1 Synopsis

To update some or all fields in an existing domain object, the following request shall be performed:

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

To add, update, and remove specific metadata items of an existing domain object, the following request shall be performed:

```
PUT <root URI>/cdmi_domains/<DomainName>/<TheDomainName>/
    ?metadata:<metadataname>;...
```

Where:

- <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 shall also be accessible at <root URI>/cdmi_objectid/<objectID>/. An update shall not result in a change to the object ID.

10.4.2 Capability

The following capability describes 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.

10.4.3 Request headers

The HTTP request headers for updating a domain object using CDMI are shown in Table 74.

Table 74 — Request headers - Update a domain object using CDMI

Header	Type	Description	Requirement
Content-Type	Header string	"application/cdmi-domain"	Mandatory
X-CDMI-Specification-Version	Header string	A comma-separated list of versions that the client supports, e.g., "1.1, 1.5, 2.0"	Mandatory

10.4.4 Request message body

The request message body fields for updating a domain object using CDMI are shown in Table 75.

Table 75 — Request message body - Update a domain object using CDMI (Sheet 1 of 2)

Field name	Type	Description	Requirement
metadata	JSON object	Metadata for the domain object. If present, the new metadata specified replaces the existing object metadata. If individual metadata items are specified in the URI, only those items are replaced; other items are preserved. See Clause 16 for a further description of metadata.	Optional
copy	JSON string	URI of a domain object that shall be copied into the existing domain object. Only the metadata and membership of the domain shall be copied, not any sub-domains of the domain.	Optional[a]
[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored.			

Table 75 — Request message body - Update a domain object using CDMI (Sheet 2 of 2)

Field name	Type	Description	Requirement
deserialize	JSON string	URI of a serialized domain object that shall be deserialized to update an existing domain object. The object ID of the serialized domain object shall match the object ID of the destination domain object. If the serialized domain does not contain children, the update is applied only to the domain object, and any existing children are left as is. If the serialized domain object does contain children, then creates, updates, and deletes are recursively applied for each child, depending on the differences between the provided serialized state and the current state of the children.	Optional[a]
deserializevalue	JSON string	A domain object serialized as specified in Clause 15 and encoded using base 64 encoding rules described in RFC 4648. The object ID of the serialized domain object shall match the object ID of the destination domain object. If the serialized domain does not contain children, the update is applied only to the domain object, and any existing children are left as is. If the serialized domain object does contain children, then creates, updates, and deletes are recursively applied for each child, depending on the differences between the provided serialized state and the current state of the children.	Optional[a]

[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored.

10.4.5 Response header

The HTTP response header for updating a domain object using CDMI is shown in Table 76.

Table 76 — Response header - Update a domain object using CDMI

Header	Type	Description	Requirement
Location	Header string	The server shall respond with an absolute URI to which the reference redirects if the object is a reference.	Conditional

10.4.6 Response message body

A response body can be provided as per RFC 2616.

10.4.7 Response status

Table 77 describes the HTTP status codes that occur when updating a domain object using CDMI.

Table 77 — HTTP status codes - Update a domain object using CDMI

HTTP status	Description
`204 No Content`	The data object content was returned in the response.
`302 Found`	The resource is a reference to another resource.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.
`409 Conflict`	The operation conflicts with a non-CDMI access protocol lock or has caused a state transition error on the server.

10.4.8 Example

EXAMPLE 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: 1.1

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

The following shows the response:

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

10.5 Delete a domain object using CDMI

10.5.1 Synopsis

To delete an existing domain object and transfer all objects associated with that domain to another domain (to preserve access), the following request shall be performed:

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

Where:

- <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 shall also be accessible at <root URI>/cdmi_objectid/<objectID>/.

10.5.2 Capability

The following capability describes the supported operations that may be performed when deleting an existing domain:

Support for the ability to delete an existing domain object is indicated by the presence of the cdmi_delete_domain capability in the specified domain.

10.5.3 Request header

The HTTP request header for deleting a domain object using CDMI is shown in Table 78.

Table 78 — Request header - Delete a domain object using CDMI

Header	Type	Description	Requirement
X-CDMI-Specification-Version	Header string	A comma-separated list of versions that the client supports, e.g., "1.1, 1.5, 2.0"	Mandatory

10.5.4 Request message body

A request body can be provided as per RFC 2616.

10.5.5 Response headers

Response headers can be provided as per RFC 2616.

10.5.6 Response message body

A response body can be provided as per RFC 2616.

10.5.7 Response status

Table 79 describes the HTTP status codes that occur when deleting a domain object using CDMI.

Table 79 — HTTP status codes - Delete a domain object using CDMI

HTTP status	Description
`204 No Content`	The domain object was successfully deleted.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.
`409 Conflict`	The operation conflicts with a non-CDMI access protocol lock or has caused a state transition error on the server.

10.5.8 Example

EXAMPLE DELETE to the domain URI:

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

The following shows the response.

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

11 Queue object resource operations using CDMI

11.1 Overview

Queue objects provide first-in, first-out access when storing and retrieving data. A queue object writer POSTs data into a queue object, and a queue object reader GETs values from the queue object and subsequently deletes the values to acknowledge receipt of the values 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 mechanism described in 9.7 "Create (POST) a new queue object using CDMI" and this clause.

Queue objects are addressed in CDMI™ in two ways:

- by name (e.g., http://cloud.example.com/queueobject); and
- by object ID (e.g., http://cloud.example.com/cdmi_objectid/00007ED900104F67307652BAC9A37C93/).

Every queue object has a single, globally-unique object identifier (ID) that remains constant for the life of the object. Each queue object shall have one or more URI addresses that allow the object to be accessed.

A queue object may have a parent object. In this case, the queue object inherits data system metadata that is not explicitly specified in the queue object itself.

EXAMPLE 1 The "receipts.queue" queue object stored at the following URI would inherit data system metadata from its parent container, "finance":

http://cloud.example.com/finance/receipts.queue

Individual fields within a queue object may be accessed by specifying the field name after a question mark "?" that is appended to the end of the data object URI.

EXAMPLE 2 The following URI returns the value field containing the oldest queue object value in the response body:

http://cloud.example.com/queueobject?value

The encoding of the data transported in the queue object value field depends on the queue object valuetransferencoding field.

- If the value transfer encoding of the object is set to "utf-8", the data stored in the value of the queue object shall be a valid UTF-8 string, and it shall be transported as a UTF-8 string in the value field.
- If the value transfer encoding of the object is set to "base64", the data stored in the value of the queue object can contain arbitrary binary sequences, and it shall be transported as a base 64-encoded string in the value field.

Specific ranges of the value of a queue object may be accessed by specifying a byte range after the value field name.

EXAMPLE 3 The following URI returns the first thousand bytes of the oldest value enqueued:

http://cloud.example.com/queueobject?value:0-999

Because a byte range of a UTF-8 string is often not a valid UTF-8 string, the response to a range request shall always be transported in the value field as a base 64-encoded string.

Byte ranges are specified as single, inclusive byte ranges as per Section 14.35.1 of RFC 2616.

If read access to any of the requested fields is not permitted by the object ACL, only the permitted fields shall be returned. If no requested fields are permitted to be read, an HTTP status code of `403 Forbidden` shall be returned to the client.

If write access to any of the requested fields is not permitted by the object ACL, no updates shall be performed, and an HTTP status code of `403 Forbidden` shall be returned to the client.

When a client provides or includes deserialization fields that are not defined in this International Standard, these fields shall be stored as part of the object.

The value of a queue object may also be specified and retrieved using multi-part MIME, where the CDMI JSON is transferred in the first MIME part and the raw queue values are transferred in the subsequent MIME parts. Each MIME part, including any header fields, shall conform to RFC 2045, RFC 2046, and RFC 2616, and the length of each part may optionally be specified by a Content-Length header in addition to the MIME boundary delimiter.

11.1.1 Queue object metadata

Queue object metadata may also include arbitrary user-supplied metadata, storage system metadata, and data system metadata, as specified in Clause 16.

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 clause are shown using JSON notation. Both clients and servers shall support UTF-8 JSON representation. The request and response body JSON fields may be specified or returned in any order, with the exception that, if present, for queue objects, the valuerange and value fields shall appear last and in that order.

11.2 Create a queue object using CDMI

11.2.1 Synopsis

To create a new queue object, the following request shall be performed:

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

To create a new queue object by ID, see 9.7 "Create (POST) a new queue object using CDMI".

Where:

- <root URI> is the path to the CDMI cloud.
- <ContainerName> is zero or more intermediate containers that already exist, with one slash (i.e., "/") between each pair of container names.
- <QueueName> is the name specified for the queue object to be created.

After it is created, the object shall also be accessible at <root URI>/cdmi_objectid/<objectID>.

The newly created queue shall have no values unless the queue is created as a result of copying or moving a source queue that has values or as a result of deserializing a serialized queue that has values.

11.2.2 Delayed completion of create

In response to a create operation for a queue object, the server may return an HTTP status code of `202 Accepted`. In this case, the queue object is in the process of being created. This response is particularly useful for long-running operations, (e.g., for copying a queue object with a large number of enqueued values from a source URI). Such a response has the following implications:

- The server shall return a Location header with an absolute URI to the object to be created along with an HTTP status code of `202 Accepted`.
- With an HTTP status code of `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; and
  - availability of space to create the queue object or at least enough space to create a URI to report an error.
- A client might not be able to immediately access the created object, e.g., due to delays resulting from the implementation's use of eventual consistency.

The client performs GET operations to the URI to track the progress of the operation. In response, 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 contains the percentage that the accepted 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.

11.2.3 Capabilities

The following capabilities describe the supported operations that can 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 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.

11.2.4 Request headers

The HTTP request headers for creating a queue object using CDMI are shown in Table 80.

Table 80 — Request headers - Create a queue object using CDMI

Header	Type	Description	Requirement
Accept	Header string	"application/cdmi-queue" or a consistent value as per per clause 5.13.2 "Content-type negotiation"	Mandatory
Content-Type	Header string	"application/cdmi-queue"	Mandatory
X-CDMI-Specification-Version	Header string	A comma-separated list of versions that the client supports, e.g., "1.1, 1.5, 2.0"	Mandatory

11.2.5 Request message body

The request message body fields for creating a queue object using CDMI are shown in Table 81.

Table 81 — Request message body - Create a queue object using CDMI (Sheet 1 of 2)

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 (i.e., "{}") 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 (see cdmi_member_privileges in Table 63 "Required settings for domain member user objects"). • If not specified, the parent domain shall be used.	Optional
deserialize	JSON string	URI of a serialized data object that shall be deserialized to create the new queue object	Optional[a]
[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored. If more than one of these fields is supplied, the server shall respond with an HTTP status code of `400 Bad Request`.			

Table 81 — Request message body - Create a queue object using CDMI (Continued) (Sheet 2 of 2)

Field name	Type	Description	Requirement
copy	JSON string	URI of a source queue object that shall be copied into the new destination queue object. • If the destination queue object URI and the copy source queue object URI both do not specify individual fields, the destination queue object shall be a complete copy of the source queue object, including all enqueued values. • If the destination queue object URI or the copy source queue object URI specifies individual fields, only the fields specified shall be used to create the destination queue object. If specified fields are not present in the source, default field values shall be used. • If the destination queue object URI and the copy source queue object URI both specify fields, an HTTP status code of `400 Bad Request` shall be returned to the client. If there are insufficient permissions to read the queue object at the source URI or create the queue object at the destination URI, or if the read operation fails, the copy shall return an HTTP status code of `400 Bad Request`, and the destination queue object shall not be created.	Optional[a]
move	JSON string	URI of an existing local or remote queue object (source URI) that shall be relocated to the URI specified in the PUT. The contents of the queue object, including the object ID, shall be preserved by a move, and the queue object at the source URI shall be removed after the queue object at the destination has been successfully created. If there are insufficient permissions to read the queue object at the source URI, write the queue object at the destination URI, or delete the queue object at the source URI, or if any of these operations fail, the move shall return an HTTP status code of `400 Bad Request`, and the source and destination are left unchanged.	Optional[a]
reference	JSON string	URI of a queue object that shall be redirected to by a reference. If other fields are supplied when creating a reference, the server shall respond with an HTTP status code of `400 Bad Request`.	Optional[a]
deserializevalue	JSON string	A queue object serialized as specified in Clause 15 and encoded using base 64 encoding rules described in RFC 4648.	Optional[a]

[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored. If more than one of these fields is supplied, the server shall respond with an HTTP status code of `400 Bad Request`.

11.2.6 Response headers

The HTTP response headers for creating a queue object using CDMI are shown in Table 82.

Table 82 — Response headers - Create a queue object using CDMI

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, e.g., "1.1". If the server does not support any of the versions that the client supports, the server shall return an HTTP status code of `400 Bad Request`.	Mandatory
Location	Header string	When an HTTP status code of `202 Accepted` is returned, the server shall respond with the absolute URL of the object that is in the process of being created.	Conditional

11.2.7 Response message body

The response message body fields for creating a queue object using CDMI are shown in Table 83.

Table 83 — Response message body - Create a queue object using CDMI (Sheet 1 of 2)

Field name	Type	Description	Requirement
objectType	JSON string	"application/cdmi-queue"	Mandatory
objectID	JSON string	Object ID of the object	Mandatory
objectName	JSON string	Name of the object	Mandatory
parentURI	JSON string	URI for the parent object Appending the objectName to the parentURI shall always produce a valid URI for the object.	Mandatory
parentID	JSON string	Object ID of the parent container object	Mandatory
domainURI	JSON string	URI of the owning domain.	Mandatory
capabilitiesURI	JSON string	URI to the capabilities for the object	Mandatory
completionStatus	JSON string	A string indicating if the object is still in the process of being created or updated by another operation, and after that operation is complete, indicates if it was successfully created or updated or if an error occurred. The value shall be the string "Processing", the string "Complete", or an error string starting with the value "Error".	Mandatory

Table 83 — Response message body - Create a queue object using CDMI (Sheet 2 of 2)

Field name	Type	Description	Requirement
percentComplete	JSON string	• When the value of completionStatus is "Processing", this field, if provided, shall indicate the percentage of completion as a numeric integer value from 0 through 100. • When the value of completionStatus is "Complete", this field, if provided, shall contain the value "100". • When the value of completionStatus is "Error", this field, if provided, may contain any integer 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 Clause 16 for a further description of metadata.	Mandatory
queueValues	JSON string	The range of designators for enqueued values. Every enqueued value shall be assigned a unique, monotonically-incrementing positive integer designator, starting from 0. If no values are enqueued, an empty string shall be returned. If values are enqueued, the lowest designator, followed by a hyphen ("-"), followed by the highest designator shall be returned.	Mandatory

11.2.8 Response status

Table 84 describes the HTTP status codes that occur when creating a queue object using CDMI.

Table 84 — HTTP status codes - Create a queue object using CDMI

HTTP status	Description
`201 Created`	The new queue object was created.
`202 Accepted`	The queue object is in the process of being created. The CDMI client should monitor the completionStatus and percentComplete fields to determine the current status of the operation.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.
`409 Conflict`	The operation conflicts with a non-CDMI access protocol lock or has caused a state transition error on the server.

11.2.9 Examples

EXAMPLE 1 PUT to the queue 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: 1.1

{
    "metadata" : {

    }
}
```

The following shows the response.

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

{
    "objectType" : "application/cdmi-queue",
    "objectID" : "00007E7F00104BE66AB53A9572F9F51E",
    "objectName" : "MyQueue",
    "parentURI " : "/MyContainer/",
    "parentID" : "00007ED900104F67307652BAC9A37C93",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/queue/",
    "completionStatus" : "Complete",
    "metadata" : {
        ...
    },
    "queueValues" : ""
}
```

EXAMPLE 2 PUT to the queue object URI to create a new queue, copying from another queue:

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

{
    "copy": "/MyContainer/SourceQueue?value:0-9"
}
```

The following shows the response.

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

{
    "objectType": "application/cdmi-queue",
    "objectID": "00007E7F00104BE66AB53A9572F9F51E",
    "objectName": "MyQueue",
    "parentURI ": "/MyContainer/",
    "parentID": "00007ED900104F67307652BAC9A37C93",
    "domainURI": "/cdmi_domains/MyDomain/",
    "capabilitiesURI": "/cdmi_capabilities/queue/",
    "completionStatus": "Complete",
    "metadata": {
        ...
   },
    "queueValues": "0-9"
}
```

11.3 Read a queue object using CDMI

11.3.1 Synopsis

To read all fields from an existing queue object, the following request shall be performed:

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

To read one or more requested fields from an existing queue object, one of the following requests shall be performed:

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

To read one or more queue values from an existing queue object, the following request shall be performed:

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

Where:

- <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 of the queue object value to be returned in the value field. If a byte range is requested, the range returned shall be from the oldest queue object value.
- <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 object. If more queue object entries are requested to be retrieved than exist in the queue object, the count is processed as if it is equal to the number of entries in the queue object.

The object shall also be accessible at <root URI>/cdmi_objectid/<objectID>.

Reading a queue object shall, by default, return the complete value of the oldest item in the queue, unless the queueValues range is empty.

11.3.2 Capabilities

The following capabilities describe the supported operations that can 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 object.
- 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 object.
- Support for the ability to read a queue object using multi-part MIME is indicated by the presence of the "cdmi_multipart_mime" system-wide capability.

11.3.3 Request headers

The HTTP request headers for reading a queue object using CDMI are shown in Table 85.

Table 85 — Request headers - Read a queue object using CDMI

Header	Type	Description	Requirement
Accept	Header string	"application/cdmi-queue", "multipart/mixed", or a consistent value as per clause 5.13.2 "Content-type negotiation" If "multipart/mixed", the body shall consist of one or more MIME parts, where the first part shall contain a body of content-type "application/cdmi-queue", and the second and subsequent parts shall each contain a queue value as described in 8.4 "Update a data object using CDMI".	Optional
X-CDMI-Specification-Version	Header string	A comma-separated list of versions that the client supports, e.g., "1.1, 1.5, 2.0"	Mandatory

11.3.4 Request message body

A request body shall not be provided.

11.3.5 Response headers

The HTTP response headers for reading a queue object using CDMI are shown in Table 86.

Table 86 — Response headers - Read a queue object using CDMI

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, e.g., "1.1". If the server does not support any of the versions that the client supports, the server shall return an HTTP status code of `400 Bad Request`.	Mandatory
Content-Type	Header string	"application/cdmi-queue"	Mandatory
Location	Header string	The server shall respond with an absolute URI to which the reference redirects if the object is a reference.	Conditional

11.3.6 Response message body

The response message body fields for reading a queue object using CDMI are shown in Table 87.

Table 87 — Response message body - Read a queue object using CDMI (Sheet 1 of 4)

Field name	Type	Description	Requirement
objectType	JSON string	"application/cdmi-queue"	Mandatory
objectID	JSON string	Object ID of the object	Mandatory

Table 87 — Response message body - Read a queue object using CDMI (Sheet 2 of 4)

Field name	Type	Description	Requirement
objectName	JSON string	Name of the object • For objects in a container, the objectName field shall be returned. • For objects not in a container (objects that are only accessible by ID), the objectName field does not exist and shall not be returned.	Conditional
parentURI	JSON string	URI for the parent object • For objects in a container, the parentURI field shall be returned. • For objects not in a container (objects that are only accessible by ID), the parentURI field does not exist and shall not be returned. Appending the objectName to the parentURI shall always produce a valid URI for the object.	Conditional
parentID	JSON string	Object ID of the parent container object • For objects in a container, the parentID field shall be returned. • For objects not in a container (objects that are only accessible by ID), the parentID field does not exist and shall not be returned.	Conditional
domainURI	JSON string	URI of the owning domain	Mandatory
capabilitiesURI	JSON string	URI to the capabilities for the object	Mandatory
completionStatus	JSON string	A string indicating if the object is still in the process of being created or updated by another operation, and after that operation is complete, indicates if it was successfully created or updated or if 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	• When the value of completionStatus is "Processing", this field, if provided, shall indicate the percentage of completion as a numeric integer value from 0 through 100. • When the value of completionStatus is "Complete", this field, if provided, shall contain the value "100". • When the value of completionStatus is "Error", this field, if provided, may contain any integer 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 Clause 16 for a further description of metadata.	Mandatory

Table 87 — Response message body - Read a queue object using CDMI (Sheet 3 of 4)

Field name	Type	Description	Requirement
queueValues	JSON string	The range of designators for enqueued values. Every enqueued value shall be assigned a unique, monotonically-incrementing positive integer designator, starting from 0. If no values are enqueued, an empty string shall be returned. If values are enqueued, the lowest designator, followed by a hyphen ("-"), followed by the highest designator shall be returned.	Mandatory
mimetype	JSON array of JSON strings	MIME types for each queue object value • The MIME types of the values are returned, each corresponding to the value in the same position in the JSON array. • This field shall only be provided when completionStatus is "Complete" and when one or more values are enqueued.	Optional
valuerange	JSON array of JSON strings	The range of bytes of the queue object values to be returned in the value field • The value ranges of the values are returned, each corresponding to the value in the same position in the JSON array. • If a specific value range has been requested, the entry in the valuerange field shall correspond to the bytes requested. If the request extends beyond the end of the value, the valuerange field shall indicate the smaller byte range returned. • The valuerange field shall only be provided when the completionStatus field contains "Complete".	Optional
valuetransferencoding	JSON array of JSON strings	The value transfer encoding used for each queue object value. Two value transfer encodings are defined: • "utf-8" indicates that the queue object value contains a valid UTF-8 string, and it shall be transported as a UTF-8 string in the value field. • "base64" indicates that the queue object value may contain arbitrary binary sequences, and it shall be transported as a base 64-encoded string in the value field. The value transfer encodings are returned, each corresponding to the value in the same position in the JSON array. The valuetransferencoding field shall only be provided when the completionStatus field contains "Complete".	Optional

Table 87 — Response message body - Read a queue object using CDMI (Sheet 4 of 4)

Field name	Type	Description	Requirement
value	JSON array of JSON strings	The oldest enqueued queue object values • The values in the JSON array are returned in order from oldest to newest. • If the valuetransferencoding field indicates UTF-8 encoding, the corresponding value field shall contain a UTF-8 string using JSON escaping rules described in RFC 4627. • If the valuetransferencoding field indicates base 64 encoding, the corresponding value field shall contain a base 64-encoded string as described in RFC RFC 4648. • The value field shall not be provided when using multi-part MIME. • The value field shall only be provided when the completionStatus field contains "Complete".	Conditional

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.

11.3.7 Response status

Table 88 describes the HTTP status codes that occur when reading a queue object using CDMI.

Table 88 — HTTP status codes - Read a queue object using CDMI

HTTP status	Description
`200 OK`	The queue object content was returned in the response.
`302 Found`	The resource is a reference to another resource.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource 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.

11.3.8 Examples

EXAMPLE 1 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: 1.1
```

The following shows the response.

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

{
    "objectType": "application/cdmi-queue",
    "objectID": "00007E7F00104BE66AB53A9572F9F51E",
```

```
    "objectName": "MyQueue",
    "parentURI": "/MyContainer/",
    "parentID" : "00007ED900104F67307652BAC9A37C93",
    "domainURI": "/cdmi_domains/MyDomain/",
    "capabilitiesURI": "/cdmi_capabilities/queue/",
    "completionStatus": "Complete",
    "metadata": {},
    "queueValues": "1-1",
    "mimetype": [
        "text/plain"
    ],
    "valuerange": [
        "0-19"
    ],
    "valuetransferencoding": [
        "utf-8"
    ],
    "value": [
        "First Enqueued Value"
    ]
}
```

EXAMPLE 2 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: 1.1
```

The following shows the response.

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

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

EXAMPLE 3 GET to the queue object URI to read the first five bytes of the value of the queue object:

```
GET /MyContainer/MyQueue?value:0-4 HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-queue
X-CDMI-Specification-Version: 1.1
```

The following shows the response:

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

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

EXAMPLE 4 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: 1.1
```

The following shows the response.

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

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

EXAMPLE 5 GET to the queue object URI to read the queue object using multi-part MIME:

```
GET /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Accept: multipart/mixed
X-CDMI-Specification-Version: 1.1
```

The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: multipart/mixed; boundary=gc0p4Jq0M2Yt08j34c0p
X-CDMI-Specification-Version: 1.1

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/cdmi-queue

{
  "objectType": "application/cdmi-queue",
  "objectID": "00007ED9001035E14BD1BA70C2EE98FC",
  "objectName": "MyQueue",
  "parentURI": "/MyContainer/",
  "parentID" : " 00007ED90010C2414303B5C6D4F83170",
  "domainURI": "/cdmi_domains/MyDomain/",
  "capabilitiesURI": "/cdmi_capabilities/queue/",
  "completionStatus": "Complete",
  "metadata": {
    ...
},
  "queueValues": "1-2",
  "mimetype": [
    "application/octet-stream",
    "application/octet-stream"
  ],
  "valuerange": [
    "0-19",
    "0-36"
  ],
  "valuetransferencoding": [
    "base64",
    "base64"
  ]
}

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary

<20 bytes of binary data>
```

```
--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary

<37 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p--
```

11.4 Update a queue object using CDMI

11.4.1 Synopsis

To update some or all fields in an existing queue object (excluding the enqueueing of values), the following request shall be performed:

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

To add, update, and remove specific metadata items of an existing queue object, the following request shall be performed:

```
PUT <root URI>/<ContainerName>/<QueueName>?metadata:<metadataname>;...
```

Where:

- <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 updated.

The object shall also be accessible at <root URI>/cdmi_objectid/<objectID>. An update shall not result in a change to the object ID.

11.4.2 Capability

The following capability describes 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 object.

11.4.3 Request headers

The HTTP request headers for updating a queue object using CDMI are shown in Table 89.

Table 89 — Request headers - Update a queue object using CDMI

Header	Type	Description	Requirement
Content-Type	Header string	"application/cdmi-queue"	Mandatory
X-CDMI-Specification-Version	Header string	A comma-separated list of versions that the client supports, e.g., "1.1, 1.5, 2.0"	Mandatory

11.4.4 Request message body

The request message body fields for updating a queue object using CDMI are shown in Table 90.

Table 90 — Request message body - Update a queue object using CDMI (Sheet 1 of 2)

Field name	Type	Description	Requirement
metadata	JSON object	Metadata for the queue object. If present, the new metadata specified replaces the existing object metadata. If individual metadata items are specified in the URI, only those items are replaced; other items are preserved. See Clause 16 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 (see cdmi_member_privileges in Table 63 "Required settings for domain member user objects"). • If not specified, the existing domain shall be preserved.	Optional
deserialize	JSON string	URI of a serialized queue object that shall be deserialized to update an existing queue object. The object ID of the serialized queue object shall match the object ID of the destination queue object. All enqueued items in the serialized queue object shall be added to the destination queue object.	Optional[a]
copy	JSON string	URI of a source queue object that shall be copied into the existing destination queue object. • If the destination queue object URI and the copy source queue object URI both do not specify individual fields, the destination queue object shall be replaced with the source queue object, with the exception that the destination queue values shall be preserved. See 11.6 "Enqueue a new queue value using CDMI" to copy enqueued items. • If the destination queue object URI or the copy source queue object URI specifies individual fields, only the fields specified shall be used to update the destination queue object. If specified fields are not present in the source, these fields shall be ignored. If the value field is specified, it shall be ignored. • If the destination queue object URI and the copy source queue object URI both specify fields, an HTTP status code of `400 Bad Request` shall be returned to the client. If there are insufficient permissions to read the queue object at the source URI or update the queue object at the destination URI, or if the read operation fails, the copy shall return an HTTP status code of `400 Bad Request`, and the destination queue object shall not be updated.	Optional[a]

[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored.

Table 90 — Request message body - Update a queue object using CDMI (Sheet 2 of 2)

Field name	Type	Description	Requirement
deserializevalue	JSON string	A queue object serialized as specified in Clause 15 and encoded using base 64 encoding rules described in RFC 4648. The object ID of the serialized queue object shall match the object ID of the destination queue object. All enqueued items in the serialized queue object shall be added to the destination queue object.	Optional[a]
[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored.			

11.4.5 Response header

The HTTP response header for updating a queue object using CDMI is shown in Table 91.

Table 91 — Response header - Update a queue object using CDMI

Header	Type	Description	Requirement
Location	Header string	The server shall respond with an absolute URI to which the reference redirects if the object is a reference.	Conditional

11.4.6 Response message body

A response body can be provided as per RFC 2616.

11.4.7 Response status

Table 92 describes the HTTP status codes that occur when updating a queue object using CDMI.

Table 92 — HTTP status codes - Update a queue object using CDMI

HTTP status	Description
`204 No Content`	The data object content was returned in the response.
`302 Found`	The resource is a reference to another resource.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.
`409 Conflict`	The operation conflicts with a non-CDMI access protocol lock or has caused a state transition error on the server.

11.4.8 Examples

EXAMPLE 1 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: 1.1

{
    "metadata" : {
```

```
    }
}
```

The following shows the response.

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

EXAMPLE 2 PUT to the queue object URI to move six queue values from another queue:

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

{
    "move": "/MyContainer/SourceQueue?value:10-15"
}
```

The following shows the response.

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

11.5 Delete a queue object using CDMI

11.5.1 Synopsis

To delete an existing queue object, along with all enqueued values, the following request shall be performed:

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

Where:

- <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 shall also be accessible at <root URI>/cdmi_objectid/<objectID>.

11.5.2 Capability

The following capability describes the supported operations that may be performed when deleting an existing queue 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 object.

11.5.3 Request header

The HTTP request header for deleting a queue object using CDMI is shown in Table 93.

Table 93 — Request header - Delete a queue object using CDMI

Header	Type	Description	Requirement
X-CDMI-Specification-Version	Header string	A comma-separated list of versions that the client supports, e.g., "1.1, 1.5, 2.0"	Mandatory

11.5.4 Request message body

A request body can be provided as per RFC 2616.

11.5.5 Response headers

Response headers can be provided as per RFC 2616.

11.5.6 Response message body

A response body can be provided as per RFC 2616.

11.5.7 Response status

Table 94 describes the HTTP status codes that occur when deleting a queue object using CDMI.

Table 94 — HTTP status codes - Delete a queue object using CDMI

HTTP status	Description
`204 No Content`	The queue object was successfully deleted.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.
`409 Conflict`	The operation conflicts with a non-CDMI access protocol lock or has caused a state transition error on the server.

11.5.8 Example

EXAMPLE DELETE to the queue object URI:

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

The following shows the response.

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

11.6 Enqueue a new queue value using CDMI

11.6.1 Synopsis

To enqueue one or more values into an existing queue object, the following request shall be performed:

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

Where:

- <root URI> is the path to the CDMI cloud.
- <ContainerName> is zero or more intermediate containers that already exist, with one slash (i.e., "/") between each pair of container names.
- <QueueName> is the name of the queue object to be enqueued into.

The object shall also be accessible at <root URI>/cdmi_objectid/<objectID>.

11.6.2 Capabilities

The following capabilities describe the supported operations that can 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 object.
- Support for the ability to modify the value of an existing queue object using multi-part MIME is indicated by the presence of the "cdmi_multipart_mime" system-wide capability.

11.6.3 Request headers

The HTTP request headers for enqueuing a new queue object value using CDMI are shown in Table 95.

Table 95 — Request headers - Enqueue a new queue object value using CDMI

Header	Type	Description	Requirement
Content-Type	Header string	"application/cdmi-queue" or "multipart/mixed" If "multipart/mixed", the first part shall contain a body of content-type "application/cdmi-queue", and the subsequent parts shall contain the queue values as described in 8.3 "Read a data object using CDMI".	Mandatory
X-CDMI-Specification-Version	Header string	A comma-separated list of versions that the client supports, e.g., "1.1, 1.5, 2.0"	Mandatory

11.6.4 Request message body

The request message body fields for enqueuing a new queue object value using CDMI are shown in Table 96.

Table 96 — Request message body - Enqueue a new queue object value using CDMI (Sheet 1 of 3)

Field name	Type	Description	Requirement
mimetype	JSON array of JSON strings	MIME types of the data values to be enqueued into the queue object. • This field shall be stored as part of the queue object. • If this field is not included and multi-part MIME is not being used, the value of "text/plain" shall be assigned as the field value. • If this field is not included and multi-part MIME is being used, the value of the "Content-Type" header of the corresponding MIME part shall be assigned as the field value. • The same number of array elements shall be present as is present in the value field, and the mimetype field shall be associated with the value in the corresponding position. • This mimetype field value shall be converted to lower case before being stored.	Optional

[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored. If more than one of these fields is supplied, the server shall respond with an HTTP status code of `400 Bad Request`.

Table 96 — Request message body - Enqueue a new queue object value using CDMI (Sheet 2 of 3)

Field name	Type	Description	Requirement
copy	JSON string	URI of a source data object or queue object from which the value shall be copied and enqueued • If a copy source object URI to a data object is provided, the value, mimetype, and valuetransferencoding field values from the source data object are used to enqueue the new item into the destination queue object. • If a copy source object URI to a queue object is provided, the corresponding value, mimetype, and valuetransferencoding field values of the specified number of enqueued items in the source queue object are copied to the destination queue object.	Optional[a]
move	JSON string	URI of a source data object or queue object from which the value shall be moved and enqueued • If a move source object URI to a data object is provided, the value, mimetype, and valuetransferencoding field values from the source data object are used to enqueue the new item into the destination queue object, and the source data object is atomically deleted. • If a move source object URI to a queue object is provided, the corresponding value, mimetype, and valuetransferencoding field values of the specified number of enqueued items in the source queue object are transferred to the destination queue object and atomically removed from the source queue object.	Optional[a]

[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored. If more than one of these fields is supplied, the server shall respond with an HTTP status code of `400 Bad Request`.

Table 96 — Request message body - Enqueue a new queue object value using CDMI (Sheet 3 of 3)

Field name	Type	Description	Requirement
valuetransferencoding	JSON array of JSON strings	The value transfer encoding used for the queue object value. Two value transfer encodings are defined: • "utf-8" indicates that the queue object value contains a valid UTF-8 string, and shall be transported as a UTF-8 string in the value field. • "base64" indicates that the queue object value may contain arbitrary binary sequences, and shall be transported as a base 64 encoded string in the value field. Setting the contents of the queue object value field to any value other than a valid base 64 string shall result in an HTTP status code of `400 Bad Request` being returned to the client. • If this field is not included and multi-part MIME is not being used, the value of "utf-8" shall be assigned as the field value. • If this field is not included and multi-part MIME is being used, the value of "utf-8" shall be assigned as the field value if the "Content-Type" header of the corresponding MIME part includes the charset parameter as defined in RFC 2046 of "utf-8" (e.g., ";charset=utf-8"). Otherwise, the value of "base64" shall be assigned as the field value. This field applies only to the encoding of the value when represented in JSON; the "Content-Transfer-Encoding" header of the part specifies the encoding of the value with a multi-part MIME request, as defined in RFC 2045. • This field shall be stored as part of the object.	Optional
value	JSON array of JSON strings	Data to be enqueued into the queue object. • If this field is not included and multi-part MIME is being used, the contents of the MIME parts shall be assigned as the field value. • If the corresponding valuetransferencoding field indicates UTF-8 encoding, the value shall be a UTF-8 string escaped using the JSON escaping rules described in RFC 4627. • If the corresponding valuetransferencoding field indicates base 64 encoding, the value shall be first encoded using the base 64 encoding rules as described in RFC 4648.	Optional[a]

[a]Only one of these fields shall be specified in any given operation. Except for value, these fields shall not be stored. If more than one of these fields is supplied, the server shall respond with an HTTP status code of `400 Bad Request`.

11.6.5 Response headers

Response headers can be provided as per RFC 2616.

11.6.6 Response message body

A response body can be provided as per RFC 2616.

11.6.7 Response status

Table 97 describes the HTTP status codes that occur when enqueuing a new queue object using CDMI.

Table 97 — HTTP status codes - Enqueue a new queue object value using CDMI

HTTP status	Description
204 No Content	The new queue object values were enqueued.
400 Bad Request	The request contains invalid parameters or field names.
401 Unauthorized	The authentication credentials are missing or invalid.
403 Forbidden	The client lacks the proper authorization to perform this request.
404 Not Found	The resource was not found at the specified URI.
409 Conflict	The operation conflicts with a non-CDMI access protocol lock or has caused a state transition error on the server.

11.6.8 Examples

EXAMPLE 1 POST to the queue object URI a new value:

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

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

The following shows the response.

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

EXAMPLE 2 POST to the queue object 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: 1.1

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

The following shows the response.

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

EXAMPLE 3 POST to the queue object URI to transfer 20 values from another queue object:

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

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

The following shows the response.

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

EXAMPLE 4 POST to the queue object URI two new values:

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

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

The following shows the response.

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

EXAMPLE 5 POST to the queue object URI two new values, one with base 64 transfer encoding and one with utf-8 transfer encoding:

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

{
   "mimetype": [
       "text/plain",
       "text/plain"
   ],
   "valuetransferencoding": [
       "utf-8",
       "base64"
   ],
   "value": [
       "First",
       "U2Vjb25k"
   ]
}
```

The following shows the response.

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

EXAMPLE 6 POST to the queue object URI the binary contents of two new values using multi-part MIME:

```
POST /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Content-Type: multipart/mixed; boundary=gc0p4Jq0M2Yt08j34c0p
X-CDMI-Specification-Version: 1.1

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/cdmi-queue

{}

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary

<20 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary

<37 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p--
```

The following shows the response.

```
HTTP/1.1 204 No content
```

EXAMPLE 7 POST to the queue object URI the mime types and binary contents of two new values using multi-part MIME:

```
POST /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Content-Type: multipart/mixed; boundary=gc0p4Jq0M2Yt08j34c0p
X-CDMI-Specification-Version: 1.1

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/cdmi-queue

{
  "mimetype" : [
    "application/pdf",
    "image/jpeg"
  ]
}

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary

<20 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary

<37 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p--
```

The following shows the response.

```
HTTP/1.1 204 No content
```

11.7 Delete a queue object value using CDMI

11.7.1 Synopsis

To delete one or more of the oldest enqueued values in an existing queue, the following request shall be performed:

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

Where:

- <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, starting from the oldest, to be removed from the queue object. If more queue object entries are requested to be deleted than exist in the queue object, the count shall be considered equal to the number of entries in the queue object.
- <range> is the lowest to highest numbers as found in the queueValues field that are to be removed from the queue object. The first range value shall be smaller or equal to the lowest queue value. If the first range value is smaller than the lowest queue value, the lowest existing queue value shall be used. If the first range value is larger than the lowest queue value, an HTTP status code of `400 Bad Request` shall be returned to the client. If the second range value is higher than the highest existing queue value, the highest existing queue value shall be used, which allows for idempotent queue value deletion.

The object shall also be accessible at <root URI>/cdmi_objectid/<objectID>.

The "?value" suffix at the end of the queue resource URI shall be included to distinguish the deletion of the oldest value from the deletion of the queue object itself, as described in 11.5 "Delete a queue object using CDMI" (which deletes all enqueued values).

11.7.2 Capability

The following capability describes the supported operations that may be performed when deleting an existing queue object value:

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 object.

11.7.3 Request header

The HTTP request header for deleting a queue object value using CDMI is shown in Table 98.

Table 98 — Request header - Delete a queue object value using CDMI

Header	Type	Description	Requirement
X-CDMI-Specification-Version	Header string	A comma-separated list of versions that the client supports, e.g., "1.1, 1.5, 2.0"	Mandatory

11.7.4 Request message body

A request body can be provided as per RFC 2616.

11.7.5 Response headers

Response headers can be provided as per RFC 2616.

11.7.6 Response message body

A response body can be provided as per RFC 2616.

11.7.7 Response status

Table 99 describes the HTTP status codes that occur when deleting a queue object value using CDMI.

Table 99 — HTTP status codes - Delete a queue object value using CDMI

HTTP status	Description
`204 No Content`	The queue object value was successfully deleted.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.
`409 Conflict`	The operation conflicts with a non-CDMI access protocol lock or has caused a state transition error on the server.

11.7.8 Examples

EXAMPLE 1 DELETE to the queue object URI value to delete the oldest enqueued value:

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

The following shows the response.

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

EXAMPLE 2 DELETE to the queue object URI value to remove the ten oldest values:

```
DELETE /MyContainer/MyQueue?values:10 HTTP/1.1
Host: cloud.example.com
X-CDMI-Specification-Version: 1.1
```

The following shows the response.

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

EXAMPLE 3 DELETE to the queue object URI value to remove queue values 10 through 19:

```
DELETE /MyContainer/MyQueue?values:10-19 HTTP/1.1
Host: cloud.example.com
X-CDMI-Specification-Version: 1.1
```

The following shows the response.

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

12 Capability object resource operations using CDMI

12.1 Overview

Capability objects allow a CDMI™ client to discover what subset of this International Standard is implemented by a CDMI provider.

For each URI in a cloud storage system, the set of interactions that the system is capable of performing for 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.

Capabilities may differ from the operations permitted by an Access Control List (ACL) (see 16.1) associated with a given URI, e.g., 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 on a CDMI object that does not have a corresponding capability, an HTTP status code of `400 Bad Request` shall be returned to the client. All CDMI-compliant cloud storage systems shall implement the ability to read capabilities, but support for the functionality indicated by each capability is optional.

Every CDMI data object, container object, domain object, and queue object shall have a capabilitiesURI field that contains a valid URI of 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.

The capabilities defined as part of this International Standard are described starting in 12.1.1 "Cloud storage system-wide capabilities". Vendor-defined capabilities not specified in this International Standard shall not start with "cdmi_".

Figure 7 shows the hierarchy of capabilities and shows how the capabilitiesURI links data objects and container objects into the capabilities tree.

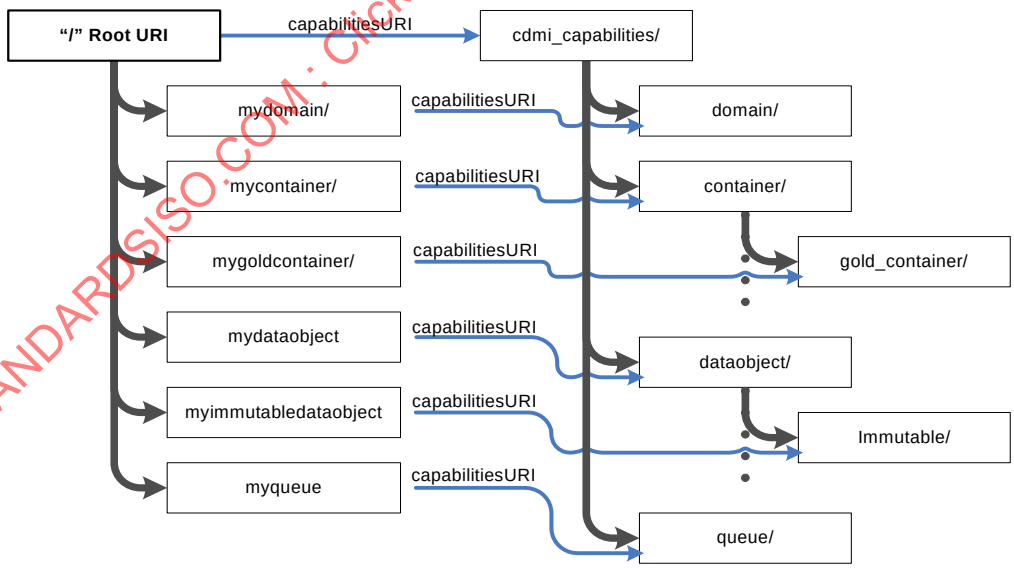

Figure 7 — Hierarchy of capabilities

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.

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 a CDMI metadata field.

12.1.1 Cloud storage system-wide capabilities

Table 100 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 100 — System-wide capabilities (Sheet 1 of 4)

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", this capability indicates that the cloud storage system supports CIFS exports.
cdmi_dataobjects	JSON string	If present and "true", this capability indicates that the cloud storage system supports data objects.
cdmi_export_iscsi	JSON string	If present and "true", this capability indicates that the cloud storage system supports iSCSI exports.
cdmi_export_nfs	JSON string	If present and "true", this capability indicates that the cloud storage system supports NFS protocol exports.
cdmi_export_occi_iscsi	JSON string	If present and "true", this capability indicates that the cloud storage system supports OCCI/iSCSI exports.
cdmi_export_webdav	JSON string	If present and "true", this capability indicates that the cloud storage system supports WebDAV exports.
cdmi_metadata_maxitems	JSON string	If present, this capability indicates the maximum number of user-defined metadata items supported per object. If absent, there is no limit placed on the number of user-defined metadata items.
cdmi_metadata_maxsize	JSON string	If present, this capability indicates the maximum size, in bytes, of each user-defined metadata item supported per object. If absent, there is no limit placed on the size of user- defined metadata items.
cdmi_metadata_maxtotalsize	JSON string	If present, this capability indicates the maximum size, in bytes, of user-defined metadata supported by the cloud storage system. If absent, there is no limit placed on the size of user-defined metadata.
cdmi_notification	JSON string	If present and "true", this capability indicates that the cloud storage system supports notification queues.
cdmi_logging	JSON string	If present and "true", this capability indicates that the cloud storage system supports logging queues.
cdmi_query	JSON string	If present and "true", this capability indicates that the cloud storage system supports query queues.

Table 100 — System-wide capabilities (Sheet 2 of 4)

Capability name	Type	Definition
cdmi_query_regex	JSON string	If present and "true", this capability indicates that the cloud storage system supports query with regular expressions.
cdmi_query_contains	JSON string	If present and "true", this capability indicates that the cloud storage system supports query with "contains" expressions.
cdmi_query_tags	JSON string	If present and "true", this capability indicates that the cloud storage system supports query with tag-matching expressions.
cdmi_query_value	JSON string	If present and "true", this capability indicates that the cloud storage system supports query of value fields.
cdmi_queues	JSON string	If present and "true", this capability indicates that the cloud storage system supports queue objects.
cdmi_security_access_control	JSON string	If present and "true", this capability indicates that the cloud storage system supports ACLs. See 12.1.3 "Data system metadata capabilities" for additional information.
cdmi_security_audit	JSON string	If present and "true", this capability indicates that the cloud storage system supports audit logging. See 20.3 "Security logging" for additional information.
cdmi_security_data_integrity	JSON string	If present and "true", this capability indicates that the cloud storage system supports data integrity/authenticity. See 12.1.3 "Data system metadata capabilities" for additional information.
cdmi_security_encryption	JSON string	If present and "true", this capability indicates that the cloud storage system supports data at-rest encryption. See 12.1.3 "Data system metadata capabilities" for additional information.
cdmi_security_immutability	JSON string	If present and "true", this capability indicates that the cloud storage system supports data immutability/retentions. See 12.1.3 "Data system metadata capabilities" for additional information.
cdmi_security_sanitization	JSON string	If present and "true", this capability indicates that the cloud storage system supports data/media sanitization. See 12.1.3 "Data system metadata capabilities" for additional information.
cdmi_serialization_json	JSON string	If present and "true", this capability indicates that the cloud storage system supports JSON as a serialization format.
cdmi_snapshots	JSON string	If present and "true", this capability indicates that the cloud storage system supports snapshots.
cdmi_references	JSON string	If present and "true", this capability indicates that the cloud storage system supports references.
cdmi_object_move_from_local	JSON string	If present and "true", this capability indicates that the cloud storage system supports moving CDMI objects from URIs within the same storage system.
cdmi_object_move_from_remote	JSON string	If present and "true", this capability indicates that the cloud storage system supports moving CDMI objects from URIs within other CDMI storage systems.
cdmi_object_move_from_ID	JSON string	If present and "true", this capability indicates that the cloud storage system supports moving CDMI objects without a path from a /cdmi_objectid/ URI within the same storage system. This effectively adds a path, allowing the object to be accessed by ID and by path.

Table 100 — System-wide capabilities (Sheet 3 of 4)

Capability name	Type	Definition
cdmi_object_move_to_ID	JSON string	If present and "true", this capability indicates that the cloud storage system supports moving CDMI objects with a path to a /cdmi_objectid/ URI within the same storage system. This effectively removes the path, leaving the object only accessible by ID.
cdmi_object_copy_from_local	JSON string	If present and "true", this capability indicates that the cloud storage system supports copying CDMI objects from URIs within the same storage system.
cdmi_object_copy_from_remote	JSON string	If present and "true", this capability indicates that the cloud storage system supports copying CDMI objects from URIs within other CDMI storage systems.
cdmi_object_access_by_ID	JSON string	If present and "true", this capability indicates that the cloud storage system supports accessing, updating, and deleting objects through /cdmi_objectid/.
cdmi_post_dataobject_by_ID	JSON string	If present and "true", this capability indicates that the cloud storage system supports adding a new data object by ID via POST to "/cdmi_objectid/".
cdmi_post_queue_by_ID	JSON string	If present and "true", this capability indicates that the cloud storage system supports adding a new queue object by ID via POST to "/cdmi_objectid/".
cdmi_deserialize_dataobject_by_ID	JSON string	If present and "true", this capability indicates that the cloud storage system supports deserializating serialized data objects when creating a new data object by ID via POST to /cdmi_objectid/.
cdmi_deserialize_queue_by_ID	JSON string	If present and "true", this capability indicates that the cloud storage system supports deserializating serialized queue objects when creating a new queue object by ID via POST to "/cdmi_objectid/".
cdmi_serialize_dataobject_to_ID	JSON string	If present and "true", this capability indicates that the cloud storage system supports serializing data objects when creating a new data object by ID via POST to "/cdmi_objectid/".
cdmi_serialize_domain_to_ID	JSON string	If present and "true", this capability indicates that the cloud storage system supports serializing domain objects when creating a new data object by ID via POST to "/cdmi_objectid/".
cdmi_serialize_container_to_ID	JSON string	If present and "true", this capability indicates that the cloud storage system allows serializing container objects when creating a new data object by ID via POST to "/cdmi_objectid/".
cdmi_serialize_queue_to_ID	JSON string	If present and "true", this capability indicates that the cloud storage system allows serializing queue objects when creating a new data object by ID via POST to "/cdmi_objectid/".
cdmi_copy_dataobject_by_ID	JSON string	If present and "true", this capability indicates that the cloud storage system supports copying an existing data object when creating a new data object by ID via POST to "/cdmi_objectid/".
cdmi_copy_queue_by_ID	JSON string	If present and "true", this capability indicates that the cloud storage system supports copying an existing queue object when creating a new queue object by ID via POST to "/cdmi_objectid/".
cdmi_create_reference_by_ID	JSON string	If present and "true", this capability indicates that the cloud storage system supports creating a new reference via POST to "/cdmi_objectid/".

Table 100 — System-wide capabilities (Sheet 4 of 4)

Capability name	Type	Definition
cdmi_copy_dataobject_from_queue	JSON string	If present and "true", this capability indicates that the cloud storage system supports the ability to copy to a data object from a queue object.
cdmi_multipart_mime	JSON string	If present and "true", this capability indicates that the cloud storage system supports storing and retrieving the value of data and queue objects using multi-part MIME.
cdmi_create_value_range_by_ID	JSON string	If present and "true", this capability indicates that the system allows a new data object's value to be created with byte ranges through "/cdmi_objectid/".

12.1.2 Storage system metadata capabilities

Table 101 defines the capabilities for storage system metadata in a cloud storage system. These capabilities are found in the capabilities objects for domain objects, data objects, container objects, and queue objects. See16.3 for a description of these storage system metadata items.

Table 101 — Capabilities for storage system metadata

Capability name	Type	Definition
cdmi_acl	JSON string	If present and "true", this capability 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 101 of 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", this capability indicates that the cloud storage system shall generate a cdmi_size storage system metadata for each stored object.
cdmi_ctime	JSON string	If present and "true", this capability indicates that the cloud storage system shall generate a cdmi_ctime storage system metadata for each stored object.
cdmi_atime	JSON string	If present and "true", this capability indicates that the cloud storage system shall generate a cdmi_atime storage system metadata for each stored object.
cdmi_mtime	JSON string	If present and "true", this capability indicates that the cloud storage system shall generate a cdmi_mtime storage system metadata for each stored object.
cdmi_acount	JSON string	If present and "true", this capability indicates that the cloud storage system shall generate a cdmi_acount storage system metadata for each stored object.
cdmi_mcount	JSON string	If present and "true", this capability indicates that the cloud storage system shall generate a cdmi_mcount storage system metadata for each stored object.

12.1.3 Data system metadata capabilities

Table 102 defines the capabilities that indicate which data system metadata items are supported for objects stored in a cloud storage system. These capabilities are found in the capabilities objects for

domains, data objects, containers, and queues. See 16.4 "Support for data system metadata" (Table 119) for a description of the meaning of the corresponding data system metadata items.

Table 102 — Capabilities for data system metadata (Sheet 1 of 3)

Capability name	Type	Definition
cdmi_assignedsize	JSON string	When the cloud storage system supports the cdmi_assignedsize data system metadata as defined in 16.4, the cdmi_assignedsize capability shall be present and set to the string value "true". When this capability is absent, or present and set to the string value "false", cdmi_assignedsize data system metadata shall not be used.
cdmi_data_redundancy	JSON string	When the cloud storage system supports the cdmi_data_redundancy data system metadata as defined in 16.4, the cdmi_data_redundancy capability shall be present and set to a positive numeric string representing the maximum value that the server supports. When this capability is absent, or present and set to an empty string value "", cdmi_data_redundancy data system metadata shall not be used.
cdmi_data_dispersion	JSON string	When the cloud storage system supports the cdmi_data_dispersion data system metadata as defined in 16.4, the cdmi_data_dispersion capability shall be present and set to the string value "true". When this capability is absent, or present and set to the string value "false", cdmi_data_dispersion data system metadata shall not be used.
cdmi_data_retention	JSON string	When the cloud storage system supports both the cdmi_retention_id and cdmi_retention_period data system metadata as defined in 16.4, the cdmi_data_retention capability shall be present and set to the string value "true". When this capability is absent, or present and set to the string value "false", cdmi_retention_id and cdmi_retention_period data system metadata shall not be used.
cdmi_data_autodelete	JSON string	When the cloud storage system supports the cdmi_data_autodelete data system metadata as defined in 16.4, the cdmi_data_autodelete capability shall be present and set to the string value "true". When this capability is absent, or present and set to the string value "false", cdmi_data_autodelete data system metadata shall not be used.
cdmi_data_holds	JSON string	When the cloud storage system supports the cdmi_hold_id data system metadata as defined in 16.4, the cdmi_data_holds capability shall be present and set to the string value "true". When this capability is absent, or present and set to the string value "false", cdmi_data_holds data system metadata shall not be used. When a cloud storage system supports holds for the purpose of making data immutable, the system-wide capability of cdmi_security_immutability specified in Table 100 of 12.1.1 shall be present and set to "true".
cdmi_encryption	JSON array of JSON strings	When the cloud storage system supports the cdmi_encryption data system metadata as defined in 16.4, the cdmi_encryption capability shall be present and set to one or more values described in the cdmi_encryption data system metadata section in 16.4. When this capability is absent, or present and is an empty JSON array, cdmi_encryption data system metadata shall not be used. When a cloud storage system supports at-rest encryption, the system-wide capability of cdmi_security_encryption specified in Table 100 of 12.1.1 shall be present and set to "true".

Table 102 — Capabilities for data system metadata (Sheet 2 of 3)

Capability name	Type	Definition
cdmi_geographic_placement	JSON string	When the cloud storage system supports the cdmi_geographic_placement data system metadata as defined in 16.4, the cdmi_geographic_placement capability shall be present and set to the string value "true". When this capability is absent, or present and set to the string value "false", cdmi_geographic_placement data system metadata shall not be used.
cdmi_immediate_redundancy	JSON string	When the cloud storage system supports the cdmi_immediate_redundancy data system metadata as defined in 16.4, the cdmi_immediate_redundancy capability shall be present and set to a positive numeric string representing the maximum value that the server supports. When this capability is absent, or present and set to an empty string value "", cdmi_immediate_redundancy data system metadata shall not be used.
cdmi_infrastructure_redundancy	JSON string	When the cloud storage system supports the cdmi_infrastructure_redundancy data system metadata as defined in 16.4, the cdmi_infrastructure_redundancy capability shall be present and set to a positive numeric string representing the maximum value that the server supports. When this capability is absent, or present and set to an empty string value "", cdmi_infrastructure_redundancy data system metadata shall not be used.
cdmi_latency	JSON string	When the cloud storage system supports the cdmi_latency data system metadata as defined in 16.4, the cdmi_latency capability shall be present and set to the string value "true". When this capability is absent, or present and set to the string value "false", cdmi_latency data system metadata shall not be used.
cdmi_RPO	JSON string	When the cloud storage system supports the cdmi_RPO data system metadata as defined in 16.4, the cdmi_RPO capability shall be present and set to the string value "true". When this capability is absent, or present and set to the string value "false", cdmi_RPO data system metadata shall not be used.
cdmi_RTO	JSON string	When the cloud storage system supports the cdmi_RTO data system metadata as defined in 16.4, the cdmi_RTO capability shall be present and set to the string value "true". When this capability is absent, or present and set to the string value "false", cdmi_RTO data system metadata shall not be used.
cdmi_sanitization_method	JSON array of JSON strings	When the cloud storage system supports the cdmi_sanitization_method data system metadata as defined in 16.4, the cdmi_sanitization_method capability shall be present and set to one or more values described in the cdmi_sanitization_method data system metadata section in 16.4. When this capability is absent, or present and is an empty JSON array, cdmi_sanitization_method data system metadata shall not be used. When a cloud storage system supports sanitization, the system-wide capability of cdmi_security_sanitization specified in Table 100 of 12.1.1 shall be present and set to "true".
cdmi_throughput	JSON string	When the cloud storage system supports the cdmi_throughput data system metadata as defined in 16.4, the cdmi_throughput capability shall be present and set to the string value "true". When this capability is absent, or present and set to the string value "false", cdmi_throughput data system metadata shall not be used.

Table 102 — Capabilities for data system metadata (Sheet 3 of 3)

Capability name	Type	Definition
cdmi_value_hash	JSON array of JSON strings	When the cloud storage system supports the cdmi_value_hash data system metadata as defined in 16.4, the cdmi_value_hash capability shall be present and set to one or more values described in the cdmi_value_hash data system metadata section in 16.4. When this capability is absent, or present and is an empty JSON array, cdmi_value_hash data system metadata shall not be used. When a cloud storage system supports value hashing, the system-wide capability of cdmi_security_data_integrity specified in Table 100 of 12.1.1 shall be present and set to "true".
cdmi_authentication_methods	JSON array of JSON strings	If present, this capability contains a list of server-supported authentication methods that are supported by a domain. The following values for authentication method strings are defined: • "anonymous" - Absence of authentication supported • "basic" - HTTP basic authentication supported (RFC 2617) • "digest" - HTTP digest authentication supported (RFC 2617) • "krb5" - Kerberos authentication supported, using the Kerberos domain specified in the CDMI domain (RFC 4559) • "x509" - certificate-based authentication via TLS (SNIA TLS) The following values are examples of other widely used authentication methods that may be supported by a CDMI server: • "s3" - S3 API signed header authentication supported • "openstack" - OpenStack Identity API header authentication supported Interoperability with these authentication methods are not defined by this International Standard. Servers may include other authentication methods not included in the above list. In these cases, it is up to the CDMI client and CDMI server to ensure interoperability. When present, the cdmi_authentication_methods data system metadata shall be supported for all domains.

12.1.4 Data object capabilities

Table 103 defines the capabilities for data objects in a cloud storage system.

Table 103 — Capabilities for data objects (Sheet 1 of 2)

Capability name	Type	Definition
cdmi_read_value	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to read the object's value.
cdmi_read_value_range	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to read the object's value with byte ranges.
cdmi_read_metadata	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to read the object's metadata.

Table 103 — Capabilities for data objects (Sheet 2 of 2)

Capability name	Type	Definition
cdmi_modify_value	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to modify the object's value.
cdmi_modify_value_range	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to modify the object's value with byte ranges.
cdmi_modify_metadata	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to modify the object's metadata.
cdmi_modify_deserialize_dataobject	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability of the data object to deserialize a serialized data object into the data object as an update.
cdmi_delete_dataobject	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to delete the object.

12.1.5 Container capabilities

Table 104 defines the capabilities for containers in a cloud storage system.

Table 104 — Capabilities for containers (Sheet 1 of 3)

Capability name	Type	Definition
cdmi_list_children	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to list the container's children.
cdmi_list_children_range	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to list the container's children with ranges.
cdmi_read_metadata	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to read the container's metadata.
cdmi_modify_metadata	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to modify the container's metadata.
cdmi_modify_deserialize_container	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability of the container object to deserialize a serialized container object into the container object as an update.
cdmi_snapshot	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability of the container object to create a new snapshot.
cdmi_serialize_dataobject	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to serialize a data object.
cdmi_serialize_container	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to serialize the container and all children's contents.

Table 104 — Capabilities for containers (Sheet 2 of 3)

Capability name	Type	Definition
cdmi_serialize_queue	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to serialize a queue object.
cdmi_serialize_domain	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to serialize the domain and all child domains.
cdmi_deserialize_container	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability of the container to deserialize the serialized containers and associated serialized children into the container.
cdmi_deserialize_queue	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability of the container to deserialize the serialized queue objects into the container.
cdmi_deserialize_dataobject	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability of the container to deserialize the serialized data objects into the container.
cdmi_create_dataobject	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability of the container to add a new data object.
cdmi_post_dataobject	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability of the container to add a new data object via POST.
cdmi_post_queue	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability of the container to add a new queue object via POST.
cdmi_create_container	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to create a new container object via PUT.
cdmi_create_queue	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to create new queue objects..
cdmi_create_reference	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to create a new child reference via PUT.
cdmi_export_container_cifs	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to export a container as a file system via CIFS.
cdmi_export_container_nfs	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to export a container as a file system via NFS.
cdmi_export_container_iscsi	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to export a container as a file system via iSCSI.
cdmi_export_container_occi	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to export a container as a file system via OCCI.
cdmi_export_container_webdav	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to export a container as a file system via WebDAV.

Table 104 — Capabilities for containers (Sheet 3 of 3)

Capability name	Type	Definition
cdmi_delete_container	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to delete a container.
cdmi_move_container	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to move a container object into a container.
cdmi_copy_container	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to copy a container object into a container.
cdmi_move_dataobject	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to move a data object into a container.
cdmi_copy_dataobject	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to copy a data object into a container.
cdmi_create_value_range	JSON string	If present and "true", this capability indicates that the container allows a new data object's value to be created with byte ranges.

12.1.6 Domain object capabilities

Table 105 defines the capabilities for domains in a cloud storage system. (All capabilities refer to what may be done via CDMI content-type operations.

Table 105 — Capabilities for domain objects (Sheet 1 of 2)

Capability name	Type	Definition
cdmi_create_domain	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to add a new subdomain.
cdmi_delete_domain	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to delete a domain.
cdmi_move_domain	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to move a domain.
cdmi_domain_summary	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to support domain summaries.
cdmi_domain_members	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to support domain user management.
cdmi_list_children	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to list the domain's children.
cdmi_read_metadata	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to read the domain's metadata.
cdmi_modify_metadata	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to modify the domain's metadata.

Table 105 — Capabilities for domain objects (Sheet 2 of 2)

Capability name	Type	Definition
cdmi_modify_deserialize_domain	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to deserialize a serialized domain object into the domain object as an update.
cdmi_copy_domain	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to copy the domain (via PUT) to another URI.
cdmi_deserialize_domain	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to deserialize serialized domains and associated serialized children into the domain.

12.1.7 Queue object capabilities

Table 106 defines the capabilities for queue objects in a cloud storage system.

Table 106 — Capabilities for queue objects

Capability name	Type	Definition
cdmi_read_value	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to read a queue's value.
cdmi_read_metadata	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to read the queue's metadata.
cdmi_modify_value	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to modify the queue's value.
cdmi_modify_metadata	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to modify the queue's metadata.
cdmi_modify_deserialize_queue	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to deserialize a serialized queue into the queue as an update.
cdmi_delete_queue	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to delete a queue.
cdmi_move_queue	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to move a queue to another URI.
cdmi_copy_queue	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to copy a queue to another URI.
cdmi_reference_queue	JSON string	If present and "true", this capability indicates that the cloud storage system shall support the ability to reference a queue from another queue.

12.1.8 Capability object representations

The representations in this clause are shown using JSON notation. Both clients and servers shall support UTF-8 JSON representation. The request and response body JSON fields may be specified or returned in

any order, with the exception that, if present, for capability objects, the childrenrange and children fields shall appear last and in that order.

12.2 Read a capabilities object using CDMI

12.2.1 Synopsis

To read all fields from an existing capability object, the following request shall be performed:

```
GET <root URI>/cdmi_capabilities/<Capability>/<TheCapability>/
```

To read one or more requested fields from an existing capability object, one of the following requests shall be performed:

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

Where:

- <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 shall also be accessible at <root URI>/cdmi_objectid/<objectID>/.

12.2.2 Capability

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

All CDMI implementations shall permit clients to read all fields of all capabilities objects.

12.2.3 Request headers

The HTTP request headers for reading a capabilities object using CDMI are shown in Table 107.

Table 107 — Request headers - Read a capabilities object using CDMI

Header	Type	Description	Requirement
Accept	Header string	"application/cdmi-capability" or a consistent value as per clause 5.13.2 "Content-type negotiation"	Optional
X-CDMI-Specification-Version	Header string	A comma-separated list of versions that the client supports, e.g., "1.1, 1.5, 2.0"	Mandatory

12.2.4 Request message body

A request body shall not be provided.

12.2.5 Response headers

The HTTP response headers for reading a capabilities object using CDMI are shown in Table 108.

Table 108 — Response headers - Read a capabilities object using CDMI

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, e.g., "1.1". If the server does not support any of the versions that the client supports, the server shall return an HTTP status code of `400 Bad Request`.	Mandatory
Content-Type	Header string	"application/cdmi-capability"	Mandatory

12.2.6 Response message body

The response message body fields for reading a capabilities object using CDMI are shown in Table 109.

Table 109 — Response message body - Read a capabilities object using CDMI

Field name	Type	Description	Requirement
objectType	JSON string	"application/cdmi-capability"	Mandatory
objectID	JSON string	Object ID of the object	Mandatory
objectName	JSON string	Name of the object	Mandatory
parentURI	JSON string	URI for the parent object	Mandatory
parentID	JSON string	Object ID of the parent container object	Mandatory
capabilities	JSON object	The capabilities supported by the corresponding object. Capabilities in the "/cdmi_capabilities/" object are system-wide capabilities. Capabilities found in children objects under "/cdmi_capabilities/" correspond to the capabilities of a specific subset of objects. Each capability is expressed as a JSON string.	Mandatory
childrenrange	JSON string	The child capabilities of the capability expressed as a range. If a range of child capabilities is requested, this field indicates the children returned as a range.	Mandatory
children	JSON array of JSON strings	Names of the children capabilities objects. For the root container capabilities, 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.

12.2.7 Response status

Table 110 describes the HTTP status codes that occur when reading a capabilities object using CDMI.

Table 110 — HTTP status codes - Read a capabilities object using CDMI

HTTP status	Description
200 OK	The capabilities object content was returned in the response.
400 Bad Request	The request contains invalid parameters or field names.
401 Unauthorized	The authentication credentials are missing or invalid.
403 Forbidden	The client lacks the proper authorization to perform this request.
404 Not Found	The resource 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.

12.2.8 Examples

EXAMPLE 1 GET to the root container capabilities URI to read all fields of the container:

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

The following shows the response.

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

{
    "objectType": "application/cdmi-capability",
    "objectID": "00007E7F00104BE66AB53A9572F9F51E",
    "objectName": "cdmi_capabilities/",
    "parentURI": "/",
    "parentID": "00007E7F0010128E42D87EE34F5A6560",
    "capabilities": {
        "cdmi_domains": "true",
        "cdmi_export_nfs": "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 2 GET to the root container capabilities 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
X-CDMI-Specification-Version: 1.1
```

The following shows the response.

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

{
    "capabilities": {
        "cdmi_domains": "true",
        "cdmi_export_nfs": "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 3 GET to the root container capabilities 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
X-CDMI-Specification-Version: 1.1
```

The following shows the response.

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

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

13 Exported protocols

13.1 Overview

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 8 shows.

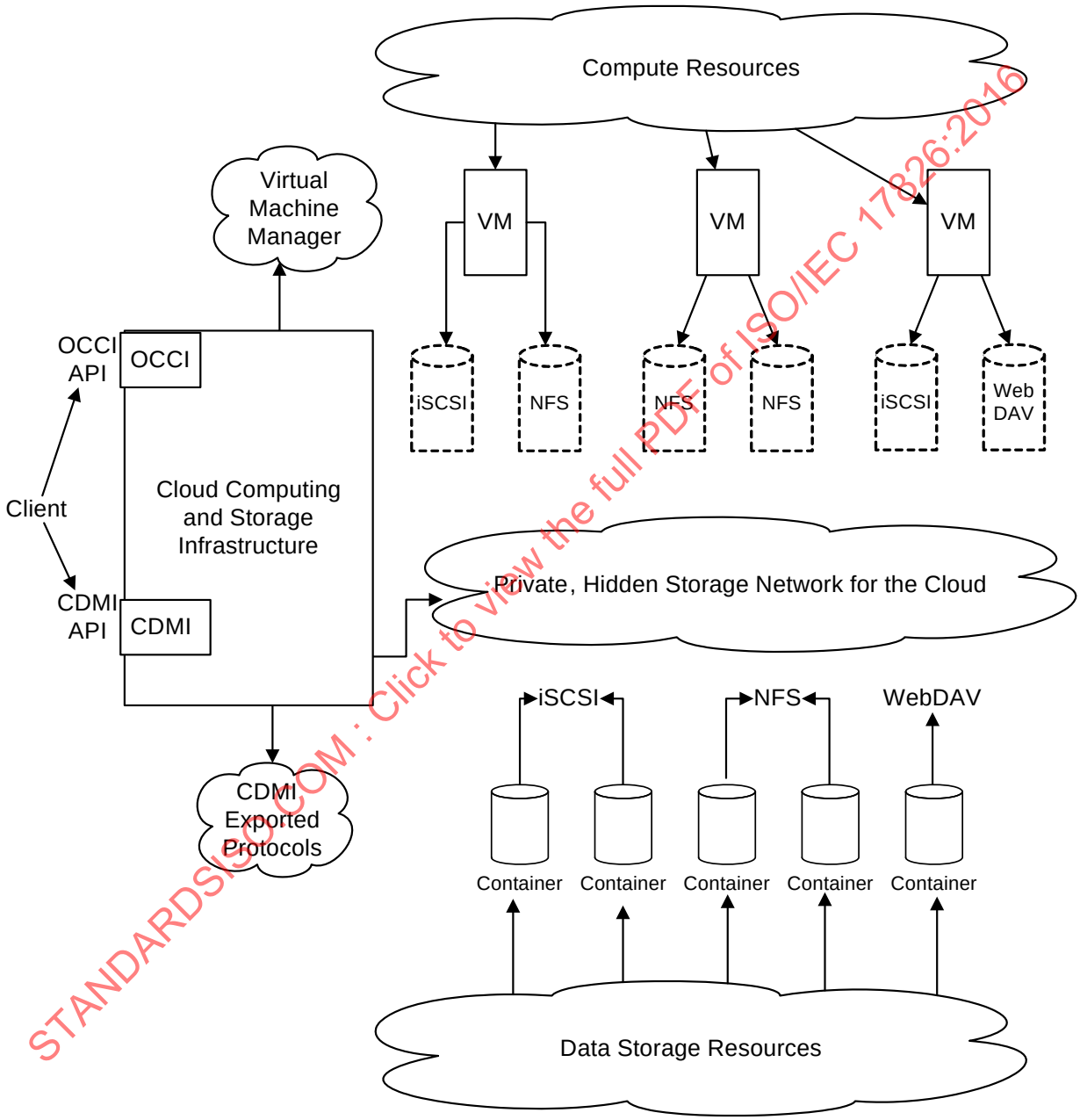

Figure 8 — CDMI and OCCI in an integrated cloud computing environment

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 associate the CDMI containers to the guests using the appropriate exported protocol.

To support exported protocols and improve their interoperability with CDMI, 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 performs operations that align the architectures, including the following.

- The client creates a CDMI container through the CDMI interface and exports it as an OCCI export protocol type. The CDMI container object ID is returned as a result.
- The client creates a virtual machine through the OCCI interface and attaches a storage volume of type CDMI using the object ID and protocol type. The OCCI virtual machine ID is returned as a result.
- The client 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.
- The client starts the virtual machine through the OCCI interface.

13.2 Exported protocol structure

The export of a container, via data path protocols other than CDMI, is accomplished by creating or updating a container and supplying one or more export protocol structures, one for each such protocol. In this International Standard, all such protocols are referred to as foreign protocols. The implementation of foreign protocols shall be indicated by "true" values for system-wide capabilities in 12.1.1 that shall always begin with "cdmi_export_".

An export protocol structure includes

- the protocol being used;
- the identity of the container as standardized by the protocol;
- the internet domain of the protocol name server for the clients being served;
- the list of who may mount that container via that protocol, identified as standardized by that protocol or optionally by leveraging the name mapping protocol (see 13.2.1) and specifying CDMI user or groupnames;
- required export parameters for the protocol;
- optional export parameters for the protocol; and
- export control parameters.

This International Standard defines JSON export structures for several well known foreign protocols. All depend on the following user and groupname mapping feature in the case that multi-protocol access to the container is desired. However, name mapping is not required if CDMI is used only to provision containers to be used exclusively by foreign protocols.

Implementations that support authenticated and authorized access to CDMI objects via both CDMI and foreign protocols need a way to support the setting of security on a per-object basis. The numerous methods of doing this include the following.

- Defining or adopting a security scheme and mapping all requests into that scheme. CDMI implementations that adopt this scheme shall use a name mapping technique to accomplish it, as (a) this mapping is easier for administrators to manage than straight id-to-id mapping, and (b) it is desired that interoperable CDMI implementations behave similarly in this respect. This means that the name of the principal in an incoming request is mapped to the name of a principal in the security domain, and that principal's id is acquired and used in the authorization procedure.
- Allowing each protocol to set its own security, which implies that an object might be accessible to a given user via one protocol but not another.
- Using the security scheme of the last protocol that was used to set permissions on the object. This method also requires mapping the principal in the incoming request to a principal in the security domain of the object. As in the first case, the server shall use a name mapping procedure to obtain the id that is used to authorize the user against the desired object's ACL.

CDMI does not mandate which method shall be used. It does, however, specify how users and groups shall be mapped between protocols.

13.2.1 Mapping names from CDMI to another protocol

Clients wishing to restrict exports via foreign protocols to mounting only by certain users and groups may be required to provide user and groupname mapping information to the server. This mapping information is also required if access to the container is desired by multiple protocols, e.g., both CDMI and NFS. The mapping is done as follows.

1 When a network share on a CDMI container is created, the server should use the appropriate mechanism, e.g., Powershell WmiClass.Create( ) on the Windows platform or /etc/exports on Unix, to limit permitted mounts of the share from other servers, as specified in the "hosts" line of the "exports" property. The syntax of the hosts line follows the syntax of /etc/exports in the Linux operating system, as encoded in a JSON string. If the CDMI server is unable to limit mounts as specified by the hosts line, an error shall result, but the success or failure of the operation depends on the implementation.

2 When any request requiring the use of a CDMI principal name comes in via a foreign protocol, the foreign domain controller to which the foreign server belongs shall be queried for the principal name corresponding to the user id given in the request. Failure to procure the principal name shall cause the original request to fail.

3 The usermap list for that protocol shall be searched, in order, for an entry matching the username gotten from the foreign domain controller (see 13.2.3 for details on the search). If no match is found, the request shall be denied. The search results may be kept in the same cache entry as the information from the preceding step.

4 The CDMI principal name gotten from the first matching usermap entry during this search is then used to authorize the user request via the security mechanism of the protocol whose security governs access to the object.

13.2.1.1 Capabilities

The following capabilities describe the supported operations that can be performed on an existing container:

- The system-wide capability to export via a given protocol is indicated by the cdmi_<protocol>_export capability in the system-level metadata (e.g., "cdmi_nfs_export", when set to "true", indicates the ability of the system to export containers via NFS). If false or not set, attempts to export containers via the given protocol shall fail.
- Support for the ability to export an existing container object via a given foreign protocol is indicated by the cdmi_<protocol>_export capability in the specified container. The default shall be "true" if this capability is unset.

13.2.1.2 Domains

The internet domain name corresponding to each export shall be given as a JSON-formatted string in the "domain" child of the protocol export specification. If it is not present, it shall be assumed that the domain is the same as that of the server hosting the CDMI implementation.

13.2.1.3 Caching

The lookup to a foreign domain controller can be quite expensive, especially for stateless protocols such as NFS v3, in which it can be theoretically required for nearly every operation. It shall be permissible to cache the results of this lookup. The recommended lifetime of a username cache entry is 30 minutes. Implementations should use this value or less when possible. Servers shall flush this cache whenever a change is made to the exports metadata concerning the protocol being cached. A client may request that the cache be flushed by reading in the usermap data for one or more protocols and writing them back without change. Servers shall flush their username mapping caches, as part of the rewrite operation, for any protocol for which the usermap information has been changed or reset.

For authorization by group to operate via a foreign protocol, a similar mapping exercise must be performed. Multiple lookups to the foreign domain controller may be required to get all the groupnames for a given user (e.g., it is common for an NFS user to be a member of several groups). A groupname cache may be used to mitigate the cost of these lookups. The recommended lifetime of a groupname cache entry is 12 hours. Implementations should use this value or less when possible. Clients may force a flush of the cache by reading in and resetting the group map information. Servers shall immediately flush their groupname mapping cache, as part of the rewrite operation, for any protocol for which the group map information has been changed or reset.

13.2.1.4 Groups

Groupname mapping for each foreign protocol shall be specified in a groupname field of the foreign protocol export specification. Its syntax is identical to the syntax for the username field.

Note: The mapping information is only required on the container being exported.

13.2.1.5 Synopsis

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

{
    "exports": {
        "nfs": {
            "hosts": [
                "*.mycollege.edu",
                "derf.cs.myuni.edu"
            ],
            "domain": "lab.mycollege.edu",
            "usermap": [
                [
                    "<cdminame>",
                    "<mapping_operator>",
                    "<nfsname>"
                ],
                [
                    "jimsmith",
                    "<->",
                    "jims"
                ],
                [
                    "*",
                    "<-->",
                    "*"
                ]
            ],
            "groupmap": [
                [
                    "<cdminame>",
                    "<mapping_operator>",
                    "<nfsname>"
                ],
                [
                    "admins",
                    "<-",
                    "wheel"
                ],
                [
                    "everyone",
                    "<-",
                    "*"
                ]
            ],
```

```
            "cifs": {
                "hosts": "*",
                "domain": "lab.mycollege.edu",
                "usermap": [
                    [
                        "<cdminame>",
                        "<mapping_operator>",
                        "<cifsname>"
                    ],
                    [
                        "jimsmith",
                        "<-->",
                        "james.smith"
                    ],
                    [
                        "*",
                        "<-->",
                        "*"
                    ]
                ],
                "groupmap": [
                    [
                        "<cdminame>",
                        "<mapping_operator>",
                        "<cifsname>"
                    ],
                    [
                        "admins",
                        "<-",
                        "Administrators"
                    ],
                    [
                        "everyone",
                        "<-",
                        "*"
                    ]
                ]
            }
        }
    }
}
```

The following shows the response.

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

{
    "objectURI" : "/Containers/MyContainer/",
    "objectID" : "00007E7F00100C435125A61B4C289455",
    "objectName" : "MyContainer/",
    "parentURI" : "/Containers/",
    "parentID" : "00007E7F0010D538DEEE8E38399E2815",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/container/",
    "completionStatus" : "Complete",
    "metadata" : {
     ...
   },
    "exports" : { <exports as listed in request> }
}
```


13.2.2 Administrative users

By default, the following users shall be considered "root", or administrative users, and equivalent to each other:

- root (Unix/NFS/LDAP),
- Administrator (Windows/AD/CIFS), and
- the domain owner (CDMI).

Servers shall automatically map these users to the root user of the target protocol unless otherwise instructed by the usermaps.

As an automatic mapping does not meet strict security standards, servers shall override these built-in entries with any usermap entries that apply to one or more root users.

EXAMPLE In the following example, root gets mapped to nobody, and everyone else is mapped to a user of the same name in the NFS domain and the CDMI domain.

```
PUT /MyContainer/ HTTP/1.1
Host: cloud.example.com
Accept: application/vnd.org.snia.cdmi.container+json
Content-Type: application/vnd.org.snia.cdmi.container+json
X-CDMI-Specification-Version: 1.1

{
    "exports": {
        "nfs": {
            "usermap": [
                [
                    "nobody",
                    "<-",
                    "root"
                ],
                [
                    "*",
                    "<-->",
                    "*"
                ]
            ]
        }
    }
}
```

Permissions mapping

The permissions sets of file-serving protocols, unfortunately, do not map on a one-to-one basis to each other. NFSv4 ACLs, Windows ACLs, POSIX ACLs, NFSv3 perms and object-based capabilities all are capable of representing security conditions that the others are not, except NFSv3, which is the least expressive. The primary area of concern is in representing the possibly rich set of permissions in a CDMI ACL in a more restricted perms-based system, such as NFSv3, for display to users.

As there are a number of possible ways to coordinate the permissions/ACLs and CDMI ACLs, this international specification does not mandate a particular method. However, all mappings of user and groupnames between domains shall use the name mapping mechanism specified in 13.2.3.

13.2.3 User and groupname mapping syntax and evaluation rules

A BNF-style grammar for name mapping is as follows:

```
name_mapping_list = protocol protocol mapping_list
protocol = "cdmi" | "nfs" | "cifs" | "ldap"
mapping_list = name mapping_operator name
name = pattern | utf8_name | quoted_utf8_name
quoted_utf8_name = " utf8_name "
utf8_name = <any legal utf8 character sequence not including the characters ",',\,/
   ,:,*,?>
pattern =  <utf8_name> * | *
mapping_operator = "<--" | "<-->" | "-->"
```

To restate this in English, a mapping entry consists of two names separated by a directional indicator. As most environments use the same usernames and groupnames across administrative domains, the most common mapping is " * <--> * ", which maps any name to the same name in the foreign protocol domain, and vice versa. It is highly recommended that this be both the default map and the last entry on all more complex maps.

CDMI specifies pattern matching on names in the name map, but only prefix matching is required. The symbol " * " at the end of a character string shall match zero or more occurrences of any non-whitespace character.

Evaluation of the name mapping list shall proceed in order; once a match is made, evaluation shall cease and the result of the match shall be returned.

If no matches are found on the match list, the result is system dependent. However, it is recommended that servers either deny access altogether or map the user in question to the equivalent of "anonymous" on the destination protocol. It is also recommended that an entry be devoted to the special user "EVERYONE@".

13.3 Discovering and mounting containers via foreign protocols

Clients need a way to discover exported containers that may be available for mounting. Discovering containers is done via a GET operation to the "exports" member of a container.

Synopsis:

To read all exports for an existing container object, the following request shall be performed:

```
GET <root URI>/<ContainerName>/<TheContainerName>/?exports
```

To read selected exports for an existing container object, the following request shall be performed:

```
GET <root URI>/<ContainerName>/<TheContainerName>/
   ?exports:protocol=<protocol>,user=<user>,verbose="false"
```

Where:

- <root URI> is the path to the CDMI cloud.
- <ContainerName> is zero or more intermediate containers.
- <TheContainerName> is the name specified for the topmost container for which exports are available.
- <protocol> is the name of a protocol to which query results should be restricted. This parameter is optional; if it is omitted or a value of "all" is given, information about all protocols shall be returned, subject to additional filtering.
- <user> is the login name of a CDMI user who wishes to mount the share. This parameter is optional and defaults to the owner of the container. When non-empty, servers shall filter the returned export list to include only exports which may be mounted given the restrictions in the protocol export structures.
- <verbose> is an optional parameter indicating a desire for maximum information about the exports. When present, it shall have the values "true" or "false". The default is "false". When true, the server should return additional information about the container, as contained in its "exports" member. The amount of said information that is returned is implementation dependent, as server implementors need to be able to balance the needs of their clients against various security considerations.

13.4 NFS exported protocol

To export a container via NFS, the information required is exactly what the server implementation will use to do the export. Normally, this information is contained in the /etc/exports file on a server or the equivalent. Administrators should be aware that lines may be automatically added to that file for each CDMI container that is exported.

Required members of the protocol structure for NFS are described in Table 111.

Table 111 — Required members of the NFS protocol structure

Member	Description
protocol	The protocol being requested. This value shall be "NFSv3", "NFSv4", "NFSv4.1", or any subsequent NFS version enshrined in a major IETF RFC. Version 2 of NFS is not supported by CDMI.
exportpath	The pathname to which the export should be surfaced. This value shall be a UTF8 string of the form [<server>]:/<path>, where the <server> component is optional, (e.g., "eeserver:/lessons/number1"). The <server> component of the path must be obtained from an administrator of the service running the CDMI implementation.
exportdomain	The internet domain of the protocol name server for the clients being served. This value is normally the name of the LDAP domain for the organization, e.g., "iti.edu". A value of "." shall be interpreted to be the DNS name of the domain occupied by the CDMI server.
mode	This value shall be "ro", "rw", "root" or "rpc_gsssec" and becomes the default export mode. Hosts requiring different access shall be specified in the optional "rw_mode", "ro_mode", and "root_mode" structure members. However, the "rpc_gsssec" mode overrides all other modes, and all other mode members and their contents shall be ignored if it is specified.
control	Export control for the container. This value shall be "immediate", "off", "on", or <n> (a number). Servers may set the value to on, but clients shall not. A numeric value (<n>) indicates that the export should be shut down in <n> seconds, possibly after a message has been sent to clients mounting the export. If a client specifies a value for <n> but the server does not support delayed shutdown of exports, then <n> shall be interpreted to mean off.

Optional export parameters for NFS are described in Table 112.

Table 112 — Optional NFS export parameters (Sheet 1 of 2)

Parameter	Description
domain_servers	A list of server names or IP addresses that function as name servers for the domain given in "domain". If given, this list shall override the names obtainable by the CDMI server via other programmatic means.
mount_name	The name the client should use to surface the export. This name replaces the last name in the path string, (e.g., mounting "eeserver:/lessons/number1" with a mountname of "1" over the directory /somepath/lessons/num1 should result in a /somepath/lessons/1 directory on the client).

Table 112 — Optional NFS export parameters (Sheet 2 of 2)

Parameter	Description
hosts	A list of hosts that can access the container in the mode given in "mode". The default shall be "*"; other values restrict the possibilities.
root_hosts	A list of hosts that can access the container in superuser mode. The default shall be an empty list.
rw_hosts	A list of hosts that can access the container in r/w mode. The default shall be an empty list.
ro_hosts	A list of hosts that can access the container in r/o mode only. The default shall be an empty list.
mount_type	One of the two strings "hard" or "soft". Clients hang when a server serving a hard mount becomes unresponsive. Clients with soft mounts generate error messages. The default is implementation dependent.
recurse	This value shall be either "true" or "false". The default shall be "true". When true, recurse indicates that mounts within the CDMI directory structure (presumably put there by other NFS operations) shall be followed and the mounted directory exposed as though it were part of the CDMI container actually being exported. This parameter is equivalent to the Linux "crossmnt" parameter.

Other export parameters for NFS are not specified by the CDMI protocol but may be included in the export structure. These parameters include Linuxisms, such as "sync", "no_wdelay", "insecure_locks", and "no_acl", as well as any other parameters used by a given server operating system. In all such cases, the parameter shall be specified as a JSON tuple in which "true" and "false" are explicitly called out for binary flags, and a JSON-formatted string or list is used for other parameters.

EXAMPLE

```
{
    "exports": {
        "nfs": {
            ...
            "no_wdelay": "true",
            "refer": "otherserver://path/leaf"
            ...
        }
    }
}
```

Export control

Export control is accomplished with the use of a single member, named "control."

- The value "immediate" shall indicate to the server that the export shall be made successfully before the PUT operation returns. Servers shall reset the value to "on" and place that in the reply.
- The value "off" shall indicate to the server that the export, if new, shall not be enabled, and if existing, shall be shut down and all client connections forcibly broken.
- A numeric value <n> shall indicate that the server shall wait <n> seconds before forcibly shutting down the export and breaking client connections. Whether the server sends a warning message to clients, giving them a chance to exit from the connection gracefully, is recommended but implementation dependent. Once the export has been shut down, the server shall also change the value of "control" to "off" in the export structure.

Servers shall support wildcard matching on the " * " and " ? " characters in the hosts lists (this is standard practice), so that **.cs.uscs.edu" matches all servers in the cs.ucsc.edu department.

Servers may support netgroup names in the various hosts lists. When this functionality is supported, these names shall resolve to ordinary lists of hostnames via queries to the domain nameserver.

Servers may also support IP address ranges in the various lists of hosts. These IP addresses shall beaugmented by the same wildcard matching as is used for ordinary host names (e.g., "192.168.1.*" exports to all the machines on a default home network). Client-side developers should note that "exporting to" only means making a container available for export. The client must still mount the exported container before there is a connection with the server.

Users wishing to use optional and vendor-specific settings are responsible for determining from the CDMI product vendor the legal settings and their format. Servers shall return an HTTP status code of `400 Bad Request` when an export setting does not conform to an allowable setting on the server.

13.5 CIFS exported protocol

To export a container via CIFS, the information required is exactly what the server implementation will use to do the export. Where this information is contained on a server is implementation dependent. The server may add or delete lines automatically to and from that file for each CDMI container that is exported or unexported.

Required members of the protocol structure for CIFS are described in Table 113.

Table 113 — Required members of the CIFS protocol structure

Member	Description
share_name	The name that CIFS shall use to discover the share.
exportdomain	The domain of the protocol name server for the clients being served. This value is normally the name of the Active Directory LDAP domain for the organization, e.g. "iti.edu". A value of "." shall be interpreted to be the domain occupied by the CDMI server.
mode	This value shall be either "ro" or "rw".
control	Export control for the container. This value shall be "immediate", "off", or <n> (a number). Servers may set the value to on, but clients shall not. The semantics and normative requirements are exactly the same as for NFS, as documented in the paragraph "Export control" in the subclause on NFS Exports (see 13.4).

There is no protocol specification; CDMI assumes that normal SMB protocol negotiation will take place.

An optional export parameter is "comment," which is often used as a user-friendly share name on the client.

Other export parameters for CIFS are not specified by the CDMI protocol but may be included in the export structure. These parameters include vendor settings such as "forcegroup", "umask", "caching", and "oplocks", as well as any other parameters used by a given server operating system. In all such cases, the parameter shall be specified as a JSON tuple in which "true" and "false" are explicitly called out for binary flags, and a JSON-formatted string or list is used for other parameters.

EXAMPLE

```
{
    "exports": {
        "cifs": {
            "caching": [
                "manual",
                "document",
                "program"
            ],
            "oplocks": "true"
        }
    }
}
```

Users wishing to manipulate vendor-specific settings are responsible for determining from the CDMI product vendor the legal settings and their format. Servers shall return an HTTP status code of `400 Bad Request` when an export setting does not conform to an allowable setting on the server.

For more detail on the use of the OCCI export protocol structure attributes, see 13.1 "Overview". Because the actual networking and access control is under the control of a hidden, common infrastructure implementing both OCCI and CDMI, the normal permission structure shall not be provided.

13.6 OCCI exported protocol

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

- The protocol is "OCCI/<protocol standard>" (e.g., "OCCI/NFSv4").
- The identifier is the CDMI object ID.
- A JSON array of URIs to OCCI compute resources shall have access (permissions) to the exported container.

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

```
"OCCI/iSCSI": {
        "identifier": "00007E7F00104BE66AB53A9572F9F51E",
        "permissions": [
            "http://example.com/compute/0/",
            "http://example.com/compute/1/"
        ]
    }
```

For more detail on using the OCCI export protocol structure attributes, see 13.1 "Overview". Because the actual networking and access control is under the control of a hidden, common infrastructure that implements both OCCI and CDMI, the normal permission structure shall not be provided.

13.7 iSCSI export modifications

CDMI defines the export of a container using the iSCSI protocol (see RFC 3720). Each container is exported as a single SCSI Logical Unit as a Logical Unit Number (LUN). One or more iSCSI initiators import the LUN through an iSCSI target node and port using one or more iSCSI network portals (IP addresses).

The export is described by the presence of an export field structure on the container that specifies the

- export protocol ("Network/iSCSI");
- iSCSI target information (IP addresses or fully qualified domain names, target identifier, and LUN);
- logical unit world-wide name; and
- 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.

13.7.1 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"
            ]
        }
    }
```

13.7.2 Create and update containers

The following export field contents, when included in a container create or update, indicates that the container shall be exported via iSCSI. 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.

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

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.

13.7.3 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:host2"
        ]
    }
}
```

13.8 WebDAV exported protocol

CDMI defines an export protocol structure for the WebDAV standard as follows (see RFC 4918):

- The protocol is "Network/WebDAV".
- The path of the WebDAV mount point is 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.

EXAMPLE 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 CDMI as a result, and the interaction between the two protocols is purposely not described in this International Standard.

14 CDMI snapshots

A snapshot is a point-in-time copy (image) of a container and all of its contents, including subcontainers and all data objects and queue objects. The client names a snapshot of a container at the time the snapshot is requested. A snapshot operation creates a new container to contain the point-in-time image. The first processing of a snapshot operation also adds a cdmi_snapshots child container to the source container. Each new snapshot container is added as a child of the cdmi_snapshots container. The snapshot does not include the cdmi_snapshots child container or its contents (see Figure 9).

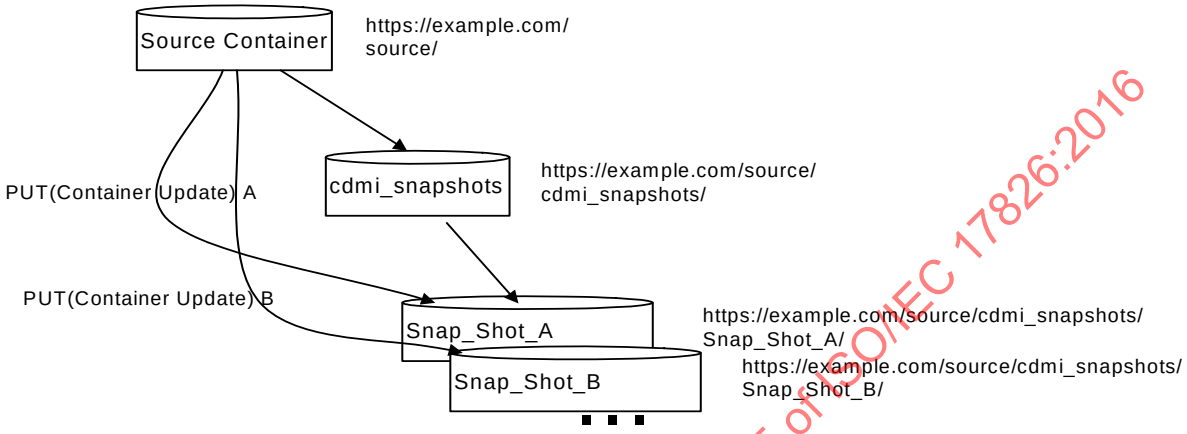

Figure 9 — Snapshot container structure

A snapshot operation is requested using the container update operation (see 9.4), in which the snapshot field specifies the requested name of the snapshot.

A snapshot may be accessed in the same way that any other CDMI™ object is accessed. An important use of a snapshot is to allow the contents of the source container to be restored to their values at a previous point in time using a CDMI copy operation.

15 Serialization/deserialization

15.1 Overview

Occasionally, bulk data movement is needed between, into, or out of clouds. When moving bulk data, cloud serialization operations provide a means to normalize data to a canonical, self-describing format, which includes:

- data migration between clouds,
- data migration during upgrades (or replacements) of cloud implementations, and
- robust backup.

The canonical format of serialized data describes how the data is to be represented in a byte stream. As long as this byte stream is not changed during the transfer from source to destination, the data may be reconstituted on the destination system.

15.2 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 object, or queue object. On a successful serialization, the result shall be a data object that is created with the serialized data as its value. If a container object has an exported block protocol, the serialized data may contain the block-by-block contents of that container object along with its metadata.

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

- If the source specified is a data object, the canonical format shall contain all data object fields, including the value, valuetransferencoding, and metadata fields.
- If the source being specified is a queue object, the canonical format shall contain all queue object fields, including the value and valuetransferencoding fields of enqueued items, along with the metadata of the queue object itself.
- If the source being specified is a container object, the canonical format shall contain all container object fields, recursively, including all children of the container object. If a user attempts to serialize a container object that includes children that the user, who is performing the serialization operation, does not have permission to read, these objects shall not be included in the resulting serialized object.

When performing a serialization operation, objects shall only be included if the principal initiating the serialization has sufficient permissions to read those objects.

15.3 Importing serialized data

Canonical data may be deserialized back into the cloud by creating a new data object, container object, or queue object and by specifying that the source for the creation is to deserialize a given data object or by specifying the serialized data in base 64 encoding in the deserializevalue field.

The destination may or may not exist previously. If not, a create operation is performed. If a container object already exists, an update operation with serialized children shall update the container object and all children. If the serialized container object does not contain children, only the container object is updated. Data objects are recreated as specified in the canonical format, including all metadata and the data object ID.

- 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. To specify a domainURI other than the domainURI of the parent, the user shall have the cross_domain privilege. If the user does not have the cross_domain privilege and specifies a domainURI other than the domainURI of the parent, an HTTP status code of `400 Bad Request` shall be returned.
- 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.

15.3.1 Canonical format

The canonical format shall represent specified data objects and container objects 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 container objects, 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 object of the specified object and from the parent of each higher-level container object.

Support for CDMI serialization using JSON as the canonical format requires the presence of the cdmi_serialization_json capability.

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 object;
- data stream contents for each data object and queue object;
- binary data represented using escaped JSON strings; and
- typing of data values consistent with CDMI JSON representations.

15.3.2 Example JSON canonical serialized format

EXAMPLE In this example, a data object and a queue object in a container object have been selected for serialization:

```
{
    "objectType": "application/cdmi-container",
    "objectID": "00007E7F00102E230ED82694DAA975D2",
    "objectName": "MyContainer/",
    "parentURI": "/",
    "parentID": "00007E7F0010128E42D87EE34F5A6560",
    "domainURI": "/cdmi_domains/MyDomain/",
    "capabilitiesURI": "/cdmi_capabilities/container/",
    "completionStatus": "Complete",
    "metadata": {
        ...
    },
    "exports": {
        "OCCI/iSCSI": {
```

```
                "identifier": "00007E7F00104BE66AB53A9572F9F51E",
                "permissions": [
                    "http://example.com/compute/0/",
                    "http://example.com/compute/1/"
                ]
            },
            "Network/NFSv4": {
                "identifier": "/users",
                "permissions": "domain"
            }
        },
        "childrenrange": "0-1",
        "children": [
            {
                "objectType": "application/cdmi-object",
                "objectID": "00007ED900104F67307652BAC9A37C93",
                "objectName": "MyDataObject.txt",
                "parentURI": "/MyContainer/",
                "parentID": "00007E7F00102E230ED82694DAA975D2",
                "domainURI": "/cdmi_domains/MyDomain/",
                "capabilitiesURI": "/cdmi_capabilities/dataobject/",
                "completionStatus": "Complete",
                "mimetype": "text/plain",
                "metadata": {
                   ...
                },
                "valuerange": "0-36",
                "valuetransferencoding": "utf-8",
                "value": "This is the Value of this Data Object"
            },
            {
                "objectType": "application/cdmi-queue",
                "objectID": "00007E7F00104BE66AB53A9572F9F51E",
                "objectName": "MyQueue",
                "parentURI": "/MyContainer/",
                "parentID": "00007E7F00102E230ED82694DAA975D2",
                "domainURI": "/cdmi_domains/MyDomain/",
                "capabilitiesURI": "/cdmi_capabilities/queue/",
                "completionStatus": "Complete",
                "metadata": {
                   ...
                },
                "queueValues": "0-1",
                "mimetype": [
                    "text/plain",
                    "text/plain"
                ],
                "valuetransferencoding": [
                    "utf-8",
                    "utf-8"
                ],
                "valuerange": [
                    "0-2",
                    "0-3"
                ],
                "value": [
                    "red",
                    "blue"
                ]
            }
        ]
    }
```


To allow efficient deserialization in stream mode when serializing container objects to JSON, the children array should be the last item in the canonical serialized JSON format.

16 Metadata

16.1 Access control

Access control comprises the mechanisms by which various types of access to objects are authorized and permitted or denied. CDMI™ uses the well-known mechanism of an Access Control List (ACL) as defined in the NFSv4 standard (see RFC 3530). 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 four fields: type, who, flags and access_mask, as per RFC 3530. The type, flags, and access_mask shall be specified as either unsigned integers in hex string representation or as a comma-delimited list of bit mask string form values taken from Table 114, Table 116, and Table 117.

16.1.2 ACE types

Table 114 defines the following ACE types, following NFSv4.

Table 114 — ACE types

String form	Description	Constant	Bit mask
"ALLOW"	Allow access rights for a principal	CDMI_ACE_ACCESS_ALLOW	0x00000000
"DENY"	Deny access rights for a principal	CDMI_ACE_ACCESS_DENY	0x00000001
"AUDIT"	Generate an audit record when the principal attempts to exercise the specified access rights	CDMI_ACE_SYSTEM_AUDIT	0x00000002

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.

The client is responsible for ordering the ACEs in an ACL. 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 external security domain (see Table 115). Some of these identifiers may not be understood when a CDMI client accesses the server, but they may have meaning when a local process accesses the file. The ability

to display and modify these permissions is permitted over CDMI, even if none of the access methods on the server understands the identifiers.

Table 115 — Who identifiers

Who	Description
"OWNER@"	The owner of the file
"GROUP@"	The group associated with the file
"EVERYONE@"	The world
"ANONYMOUS@"	Access without 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 flags in Table 116 govern this behavior.

Table 116 — ACE flags

String form	Description	Constant	Bit mask
"NO_FLAGS"	No flags are set	CDMI_ACE_FLAGS_NONE	0x00000000
"OBJECT_INHERIT"	An ACE on which OBJECT_INHERIT is set is inherited by objects as an effective ACE: OBJECT_INHERIT is cleared on the child object. When the ACE is inherited by a container, OBJECT_INHERIT is retained for the purpose of inheritance, and additionally, INHERIT_ONLY is set.	CDMI_ACE_FLAGS_OBJECT_INHERIT_ACE	0x00000001
"CONTAINER_INHERIT"	An ACE on which CONTAINER_INHERIT is set is inherited by a subcontainer as an effective ACE. Both INHERIT_ONLY and CONTAINER_INHERIT are cleared on the child container.	CDMI_ACE_FLAGS_CONTAINER_INHERIT_ACE	0x00000002
"NO_PROPAGATE"	An ACE on which NO_PROPAGATE is set is not inherited by any objects or subcontainers. It applies only to the container on which it is set.	CDMI_ACE_FLAGS_NO_PROPAGATE_ACE	0x00000004
"INHERIT_ONLY"	An ACE on which INHERIT_ONLY is set is propagated to children during ACL inheritance as specified by OBJECT_INHERIT and CONTAINER_INHERIT. The ACE is ignored when evaluating access to the container on which it is set and is always ignored when set on objects.	CDMI_ACE_FLAGS_INHERIT_ONLY_ACE	0x00000008
"IDENTIFIER_GROUP"	An ACE on which IDENTIFIER_GROUP is set indicates that the "who" refers to a group identifier.	CDMI_ACE_FLAGS_IDENTIFIER_GROUP	0x00000040
"INHERITED"	An ACE on which INHERITED is set indicates that this ACE is inherited from a parent directory. A server that supports automatic inheritance will place this flag on any ACEs inherited from the parent directory when creating a new object.	CDMI_ACE_FLAGS_INHERITED_ACE	0x00000080

16.1.5 ACE bit masks

The mask field of an ACE contains 32 bits. Table 117 defines the ACE bit masks in CDMI; their values are taken from the IETF NFSv4 RFC 3530.

Table 117 — ACE bit masks (Sheet 1 of 3)

String form	Description	Constant	Bit mask
"READ_OBJECT"	Permission to read the value of an object. If "READ_OBJECT" is not permitted: • A CDMI GET that requests all fields shall return all fields with the exception of the value field. • A CDMI GET that requests specific fields shall return the requested fields with the exception of the value field. • A CDMI GET for only the value field shall return an HTTP status code of 403 Forbidden. • A non-CDMI GET shall return an HTTP status code of 403 Forbidden.	CDMI_ACE_READ_OBJECT	0x00000001
"LIST_CONTAINER"	Permission to list the children of an object. If "LIST_CONTAINER" is not permitted: • A CDMI GET that requests all fields shall return all fields with the exception of the children field and childrenrange field. • A CDMI GET that requests specific fields shall return the requested fields with the exception of the children field and childrenrange field. • A CDMI GET for only the children field or childrenrange field shall return an HTTP status code of 403 Forbidden.	CDMI_ACE_LIST_CONTAINER	0x00000001
"WRITE_OBJECT"	Permission to modify the value of an object. If "WRITE_OBJECT" is not permitted, a PUT that requests modification of the value of an object shall return an HTTP status code of 403 Forbidden.	CDMI_ACE_WRITE_OBJECT	0x00000002
"ADD_OBJECT"	Permission to add a new child data object or queue object. If "ADD_OBJECT" is not permitted, a PUT or POST that requests creation of a new child data object or new queue object shall return an HTTP status code of 403 Forbidden.	CDMI_ACE_ADD_OBJECT	0x00000002
"APPEND_DATA"	Permission to append data to the value of a data object. If "APPEND_DATA" is permitted and "WRITE_OBJECT" is not permitted, a PUT that requests modification of any existing part of the value of an object shall return an HTTP status code of 403 Forbidden.	CDMI_ACE_APPEND_DATA	0x00000004
"ADD_SUBCONTAINER"	Permission to create a child container object or domain object. If "ADD_SUBCONTAINER" is not permitted, a PUT that requests creation of a new child container object or new domain object shall return an HTTP status code of 403 Forbidden.	CDMI_ACE_ADD_SUBCONTAINER	0x00000004
[1]The value fields, children fields, and metadata field are considered to be non-attribute fields. All other fields are considered to be attribute fields.			

Table 117 — ACE bit masks (Sheet 2 of 3)

String form	Description	Constant	Bit mask
"READ_METADATA"	Permission to read the metadata of an object. If "READ_METADATA" is not permitted: • A CDMI GET that requests all fields shall return all fields with the exception of the metadata field. • A CDMI GET that requests specific fields shall return the requested fields with the exception of the metadata field. • A CDMI GET for only the metadata field shall return an HTTP status code of 403 Forbidden.	CDMI_ACE_READ_METADATA	0x00000008
"WRITE_METADATA"	Permission to modify the metadata of an object. If "WRITE_METADATA" is not permitted, a CDMI PUT that requests modification of the metadata field of an object shall return an HTTP status code of 403 Forbidden.	CDMI_ACE_WRITE_METADATA	0x00000010
"EXECUTE"	Permission to execute an object.	CDMI_ACE_EXECUTE	0x00000020
"TRAVERSE_CONTAINER"	Permission to traverse a container object or domain object. If "TRAVERSE_CONTAINER" is not permitted for a parent container, all operations against all children below that container shall return an HTTP status code of 403 Forbidden.	CDMI_ACE_TRAVERSE_CONTAINER	0x00000020
"DELETE_OBJECT"	Permission to delete a child data object or child queue object from a container object. If "DELETE_OBJECT" is not permitted, all DELETE operations shall return an HTTP status code of 403 Forbidden.	CDMI_ACE_DELETE_OBJECT	0x00000040
"DELETE_SUBCONTAINER"	Permission to delete a child container object from a container object or to delete a child domain object from a domain object. If "DELETE_SUBCONTAINER" is not permitted, all DELETE operations shall return an HTTP status code of 403 Forbidden.	CDMI_ACE_DELETE_SUBCONTAINER	0x00000040
"READ_ATTRIBUTES"	Permission to read the attribute fields[1] of an object. If "READ_ATTRIBUTES" is not permitted: • A CDMI GET that requests all fields shall return all non-attribute fields and shall not return any attribute fields. • A CDMI GET that requests at least one non-attribute field shall only return the requested non-attribute fields. • A CDMI GET that requests only non-attribute fields shall return an HTTP status code of 403 Forbidden.	CDMI_ACE_READ_ATTRIBUTES	0x00000080
"WRITE_ATTRIBUTES"	Permission to change attribute fields[1] of an object. If "WRITE_ATTRIBUTES" is not permitted, a CDMI PUT that requests modification of any non-attribute field shall return an HTTP status code of 403 Forbidden.	CDMI_ACE_WRITE_ATTRIBUTES	0x00000100
"WRITE_RETENTION"	Permission to change retention attributes of an object. If "WRITE_RETENTION" is not permitted, a CDMI PUT that requests modification of any non-hold retention metadata items shall return an HTTP status code of 403 Forbidden.	CDMI_ACE_WRITE_RETENTION	0x00000200

[1]The value fields, children fields, and metadata field are considered to be non-attribute fields. All other fields are considered to be attribute fields.

Table 117 — ACE bit masks (Sheet 3 of 3)

String form	Description	Constant	Bit mask
"WRITE_RETENTION_HOLD"	Permission to change retention hold attributes of an object. If "WRITE_RETENTION_HOLD" is not permitted, a CDMI PUT that requests modification of any retention hold metadata items shall return an HTTP status code of 403 Forbidden.	CDMI_ACE_WRITE_RETENTION_HOLD	0x00000400
"DELETE"	Permission to delete an object. If "DELETE" is not permitted, all DELETE operations shall return an HTTP status code of 403 Forbidden.	CDMI_ACE_DELETE	0x00010000
"READ_ACL"	Permission to read the ACL of an object. If "READ_ACL" is not permitted: • A CDMI GET that requests all metadata items shall return all metadata items with the exception of the cdmi_acl metadata item. • A CDMI GET that requests specific metadata items shall return the requested metadata items with the exception of the cdmi_acl metadata item. • A CDMI GET for only the cdmi_acl metadata item shall return an HTTP status code of 403 Forbidden. If "READ_ACL" is permitted and "READ_METADATA" is not permitted, then to read the ACL, a client CDMI GET for only the cdmi_acl metadata item shall be permitted.	CDMI_ACE_READ_ACL	0x00020000
"WRITE_ACL"	Permission to write the ACL of an object. • If "WRITE_ACL" is not permitted, a CDMI PUT that requests modification of the cdmi_acl metadata item shall return an HTTP status code of 403 Forbidden. • If "WRITE_ACL" is permitted and "WRITE_METADATA" is not permitted, then to write the ACL, a client CDMI PUT for only the cdmi_acl metadata item shall be permitted.	CDMI_ACE_WRITE_ACL	0x00040000
"WRITE_OWNER"	Permission to change the owner of an object. • If "WRITE_OWNER" is not permitted, a CDMI PUT that requests modification of the cdmi_owner metadata item shall return an HTTP status code of 403 Forbidden. • If "WRITE_OWNER" is permitted and "WRITE_METADATA" is not permitted, then to write the owner, a client CDMI PUT for only the cdmi_owner metadata item shall be permitted.	CDMI_ACE_WRITE_OWNER	0x00080000
"SYNCHRONIZE"	Permission to access an object locally at the server with synchronous reads and writes.	CDMI_ACE_SYNCHRONIZE	0x00100000
[1]The value fields, children fields, and metadata field are considered to be non-attribute fields. All other fields are considered to be attribute fields.			

Implementations shall use the correct string form to display permissions, if the object type is known. If the object type is unknown, the "object" version of the string shall be used.

16.1.6 ACL evaluation

When evaluating whether access to a particular object *O* by a principal *P* is to be granted, the server shall traverse the object's logical ACL (its ACL after processing inheritance from parent containers) in list order, using a temporary permissions bitmask *m*, initially empty (all zeroes).

- 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@; and
  - 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:

```
"cdmi_acl":
[
    {
        "acetype": "ALLOW",
        "identifier": "OWNER@",
        "aceflags": "OBJECT_INHERIT, CONTAINER_INHERIT",
        "acemask": "ALL_PERMS"
    },
    {
        "acetype": "ALLOW",
        "identifier": "AUTHENTICATED@",
        "aceflags": "OBJECT_INHERIT, CONTAINER_INHERIT",
        "acemask": "READ"
    }
]
```

As ACLs are storage system metadata, they are stored and retrieved through the metadata field included in a PUT or GET request. The syntax is as follows, using the constant strings from Table 114, Table 116, and Table 117, above.

```
ACL = { ACE [, ACE ...] }
ACE = { acetype , identifier , aceflags , acemask }
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

aceobinherititem   = "CDMI_ACE_OBJECT_INHERIT_ACE" | 0x01
acecontinherititem = "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 | aceexecuteitem |
                   acereadattritem | acewriteattritem |
                   aceretentionitem
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
aceexecuteitem    = "CDMI_ACE_EXECUTE" |  0x80000
acereadattritem   = "CDMI_ACE_READ_ATTRIBUTES" | 0x00080
acewriteattritem  = "CDMI_ACE_WRITE_ATTRIBUTES" | 0x00100
aceretentionitem  = "CDMI_ACE_SET_RETENTION" | 0x10000000
```

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.

When an object is created, no ACL is supplied, and an ACL is not inherited from the parent container (or there is no parent container), the server shall place an ACL containing the following ACEs on the object:

```
"cdmi_acl":
[
    {
        "acetype": "ALLOW",
        "identifier": "OWNER@",
        "aceflags": "OBJECT_INHERIT, CONTAINER_INHERIT",
        "acemask": "ALL_PERMS"
    }
]
```

16.1.7 Example ACE mask expressions

EXAMPLE 1

```
"READ_ALL" | 0x02
```

evaluates to 0x09 | 0x02 == 0x0

EXAMPLE 2

```
0x001F07FF
```

evaluates to 0x001F07FF == "ALL_PERMS"

EXAMPLE 3

```
"RW_ALL" | DELETE
```

evaluates to 0x000601DF | 0x00010000 == 0x000701DF

16.1.8 Canonical format for ACE hexadecimal quantities

ACE mask expressions may be evaluated and converted to a string hexadecimal value before transmission in a CDMI JSON body. Applications or utilities that display them to users should convert them into a text expression before display and accept user input in text format as well.

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:

0x001F07FF	"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.

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.

EXAMPLE 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: 1.1

{
    "objectType" : "/application/cdmi-object",
    "objectID" : "00007ED9001086A99CC6487FEE373D82",
    "objectName" : "MyDataItem.txt",
    "parentURI" : "/MyContainer/",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/dataobject/",
    "completionStatus" : "Complete",
    "mimetype" : "text/plain",
    "metadata" : {
        "cdmi_size" : "17",
        "cdmi_acl" : [
            {
                "acetype" : "0x00",
                "identifier" : "EVERYONE@",
                "aceflags" : "0x00",
                "acemask" : "0x00020089"
            }
        ],
        ...
    },
    "valuerange" : "0-16",
    "value" : "Hello CDMI World!"
}
```

16.2 Support for user metadata

All CDMI 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.
- The maximum total size of user-defined metadata items for an object is specified by the capability cdmi_metadata_maxtotalsize.

16.3 Support for storage system metadata

After an object has been created, the storage system metadata, as described in Table 118, 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 118 — Storage system metadata (Sheet 1 of 3)

Metadata name	Type	Description	Requirement
cdmi_size	JSON string	The number of bytes consumed by the object. This storage system metadata item is computed by the storage system, and any attempts to set or modify it will be ignored.	Optional

Table 118 — Storage system metadata (Sheet 2 of 3)

Metadata name	Type	Description	Requirement
cdmi_ctime	JSON string	The time when the object was created, in ISO-8601 point-in-time format, as described in 5.14. This metadata value can only be updated by a client if it has the "backup_operator" privilege. If a client does not have the "backup operator privilege, updates of this metadata item shall be ignored.	Optional
cdmi_atime	JSON string	The time when the object was last accessed in ISO-8601 point-in-time format, as described in 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). For a newly created object, this value shall be set to the creation time. This metadata value can only be updated by a client if it has the "backup_operator" privilege. If a client does not have the "backup operator privilege, updates of this metadata item shall be ignored.	Optional
cdmi_mtime	JSON string	The time when the object was last modified, in ISO-8601 point-in-time format, as described in 5.14. The modification of a child is not considered a modification of a container object (modification times do not propagate up the tree). For a newly created object, this value shall be set to the creation time. This metadata value can only be updated by a client if it has the "backup_operator" privilege. If a client does not have the "backup operator privilege, updates of this metadata item shall be ignored.	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. For a newly created object, this value shall be set to the value "0". This metadata value can only be updated by a client if it has the "backup_operator" privilege. If a client does not have the "backup operator privilege, updates of this metadata item shall be ignored.	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. For a newly created object, this value shall be set to the value "0". This metadata value can only be updated by a client if it has the "backup_operator" privilege. If a client does not have the "backup operator privilege, updates of this metadata item shall be ignored.	Optional
cdmi_hash	JSON string	The hash of the value of the object, encoded using Base16 encoding rules described in RFC 4648. 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