Cloud Data Management Interface (CDMI)

Version 2.0.0

SNIA Technical Position

Mar 05, 2018

Table of Contents:

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:

Table 1: Overview of this document

Clause 1	Scope	Defines the scope of this document
Clause 2	Normative references	Lists the normative references for this document
Clause 3	Terms	Provides terminology used in this document
Clause 4	Conventions	Describes the conventions used in presenting the interfaces and the typographical conventions used in this document
Clause 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
Clause 6	Data Object Resource Operations using HTTP	Provides the normative standard of data object resource operations using HTTP
Clause 7	Container Object Resource Operations using HTTP	Provides the normative standard of container object resource operations using HTTP
Clause 8	Data Object Resource Operations using CDMI	Provides the normative standard of data object resource operations using CDMI
Clause 9	Container Object Resource Operations using CDMI	Provides the normative standard of container object resource operations using CDMI
Clause 10	Domain Object Resource Operations using CDMI	Provides the normative standard of domain object resource operations using CDMI
Clause 11	Queue Object Resource Operations using CDMI	Provides the normative standard of queue object resource operations using CDMI
Clause 9	Capability Object Resource Operations using CDMI	Provides the normative standard of capability object resource operations using CDMI
Clause 13	Exported Protocols	Discusses how virtual machines in the cloud computing environment may use the exported protocols from CDMI containers
Clause 14	Snapshots	Discusses how snapshots are accessed under CDMI containers
Clause 15	Serialization/Deserialization	Discusses serialization and deserialization, including import and export of serialized data under CDMI
Clause 16	Metadata	Provides the normative standard of the metadata used in the interface
Clause 17	Retention and Hold Management	Describes the optional retention management disciplines to be implemented into the system management functions
Clause 18	Scope Specification	Describes the structure of the scope specification for JSON objects
Clause 19	Results Specification	Provides a standardized mechanism to define subsets of CDMI object contents
Clause 20	Logging	Describes CDMI functional logging for object functions, security events, data management events, and queues
Clause 21	Notification Queues	Describes how CDMI clients may efficiently discover what changes have occurred to the system
Clause 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 Section V	(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

Clause 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.

Clause 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.

Todo: find a better way of keeping a bibliography.

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 14701:2012 Space data and information transfer systems – Open archival information system (OAIS) – Reference model

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

ISO/IEC DIS 27040 Information technology – Security techniques – Storage security

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

RFC 1867 Form-based File Upload in HTML - see **RFC 1867**

RFC 2045 Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies - see **RFC 2045**

RFC 2046 Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types - see **RFC 2046**

RFC 2119 Key Words for Use in RFCs to Indicate Requirement Levels - see **RFC 2119**

RFC 2578 Structure of Management Information Version 2 (SMIv2) - see **RFC 2578**

RFC 2616 Hypertext Transfer Protocol – HTTP/1.1 - see **RFC 2616**

RFC 2617 HTTP Authentication: Basic and Digest Access Authentication - see **RFC 2617**

RFC 3280 Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile - see **RFC 3280**

RFC 3530 Network File System (NFS) Version 4 Protocol - see **RFC 3530**

RFC 3720 Internet Small Computer Systems Interface (iSCSI) - see **RFC 3720**

RFC 3986 Uniform Resource Identifier (URI): Generic Syntax - see **RFC 3986**

RFC 4627 The Application/JSON Media Type for JavaScript Object Notation (JSON) - see **RFC 4627**

RFC 4648 The Base16, Base32, and Base64 Data Encodings - see **RFC 4648**

RFC 4918 HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV) - see **RFC 4918**

RFC 5246 The Transport Layer Security (TLS) Protocol Version 1.2 - see **RFC 5246**

RFC 6208 Cloud Data Management Interface (CDMI) Media Types - see **RFC 6208**

RFC 6839 Additional Media Type Structured Syntax Suffixes - see **RFC 6839**

SNIA TLS TLS Specification for Storage Systems, version 1.0 - https://snia.org/tech_activities/standards/curr_standards/tls

Clause 3

Terms and definitions

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

Access Control List

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

API Application Programming Interface

CDMI™ Cloud Data Management Interface

CDMI capabilities an object that describes what operations are supported for a given cloud or cloud object

the mimetype for this object is application/cdmi-capability.

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

The mimetype for this object is application/cdmi-container.

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

The mimetype for this object is application/cdmi-object.

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

The mimetype for this object is application/cdmi-domain.

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

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

The mimetype for this object is application/cdmi-queue.

CIFS Common Internet File System

cloud storage See Data storage as a Service

CRC cyclic redundancy check

CRUD create, retrieve, update, delete

Data Storage as a Service

DSaaS delivery of virtualized storage and data services on demand over a network, based on a request for a given service level that hides limits to scalability, is either self-provisioned or provisionless, and is billed based on consumption

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

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

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

FC Fibre Channel

FCoE Fibre Channel over Ethernet

HTTP HyperText Transfer Protocol

Infrastructure as a Service

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

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

iSCSI Internet Small Computer Systems Interface (see **RFC 3720**)

JSON JavaScript Object Notation

LDAP Lightweight Directory Access Protocol

LUN Logical Unit Number (see *ISO/IEC 14776-414*)

metadata data about other data (see `ref_iso_14701:2012`)

MIME Multipurpose Internet Mail Extensions (see **RFC 2045**)

NFS Network File System (see **RFC 3530**)

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

Types of CDMI objects include data objects, container objects, capability objects, domain objects, and queue objects.

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

OCCI Open Cloud Computing Interface (see *[OCCI]*)

Platform as a Service

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

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

POSIX Portable Operating System Interface (see *IEEE Std 1003.1*)

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

Private clouds are created due to issues of trust.

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

Representational State Transfer

REST a specific set of principles for defining, addressing, and interacting with resources addressable by URIs (see *[REST]*)

RPO recovery point objective

RTO recovery time objective

service level performance targets for a service

SNMP Simple Network Management Protocol

Software as a Service

SaaS delivery over a network, on demand, of the use of an application

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

Uniform Resource Identifier

URI compact sequence of characters that identifies an abstract or physical resource (see **RFC 3986**)

VIM Vendor Interface Module

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

WebDAV Web Distributed Authoring and Versioning (see **RFC 4918**)

XAM eXtensible Access Method (see [INCITS 464-2010]_)

Clause 4

Conventions

4.1 Interface Format

Each interface description has nine components, as described in Table 4.1.

Table 4.1: 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:

```
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"
}
```

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 condition(s) specified in the Description cell of this row (to the left of the Requirement) is met, the value specified in this row shall be provided. Otherwise, it may 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 4.2 shall be interpreted as described in **RFC 2119**.

Table 4.2: Key Word Requirements

Key Words	Description
shall must required	An action described with any of these key words is unconditionally required.
shall not must not	An action described with either of these key word phrases is unconditionally prohibited.
should recommended	Valid reasons may exist in specific circumstances to ignore a particular action described with either of these key words, but the full implications must be understood and carefully weighed before choosing a different course.
should not not recommended	Valid reasons may exist in specific circumstances to accept a particular action described by either of these key word phrases, but the full implications should be understood and the case carefully weighed before implementing any action described with these key words.
may optional	An action described with either of these key words is truly optional. One vendor may choose to include the option because a particular marketplace requires it or because the vendor feels that it enhances the product, while another vendor may omit the same option. An implementation which does not include a particular option must be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. Likewise, an implementation which does include a particular option must be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides).

Clause 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 may 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 may differentiate their treatment of the data to meet those requirements.

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

5.1.1 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).

5.1.2 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 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 and CIFS/NFS or WebDAV for file network storage, as shown in Fig. 5.1.

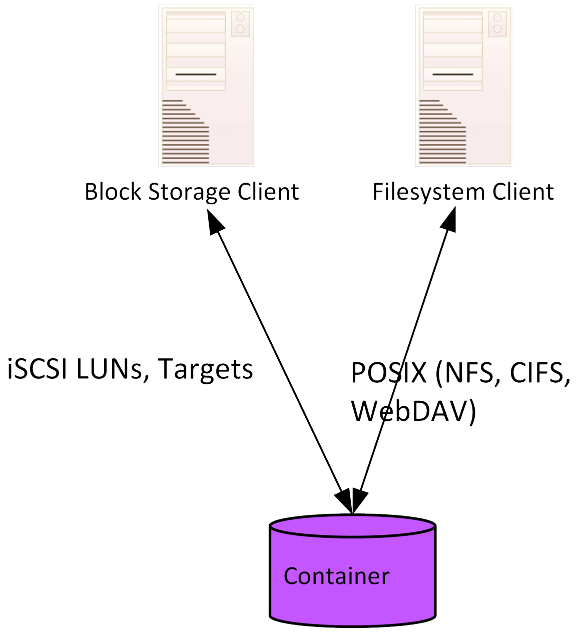

Fig. 5.1: Existing Data Storage Interface Standards

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

Managing this storage is typically done out of band 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 may be used to invoke other data services as well (e.g., snapshot and cloning).

In this model, the underlying storage space 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 Fig. 5.2).

5.1.3 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 may 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).

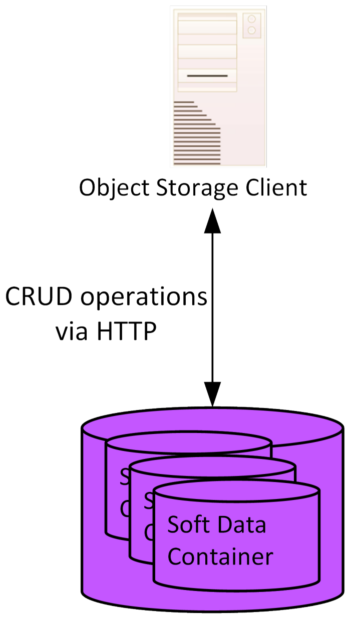

Fig. 5.2: Storage Interfaces for Object Storage Client Data

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 may be maintained while still addressing the requirements of enterprise applications and their data.

User metadata is retained by the cloud and may be used to find the data objects and containers by 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 Section 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 Section 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. It may apply to an aggregation of data objects in a container or to individual data objects, if the cloud service supports this level of granularity. For more information on support for data system metadata, see Section 16.4.

5.1.4 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 may 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 may 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 may be used to specify the data system metadata to override that specified for the container.

5.2 Reference Model for Cloud Storage Interfaces

The cloud storage reference model is shown in Fig. 5.3.

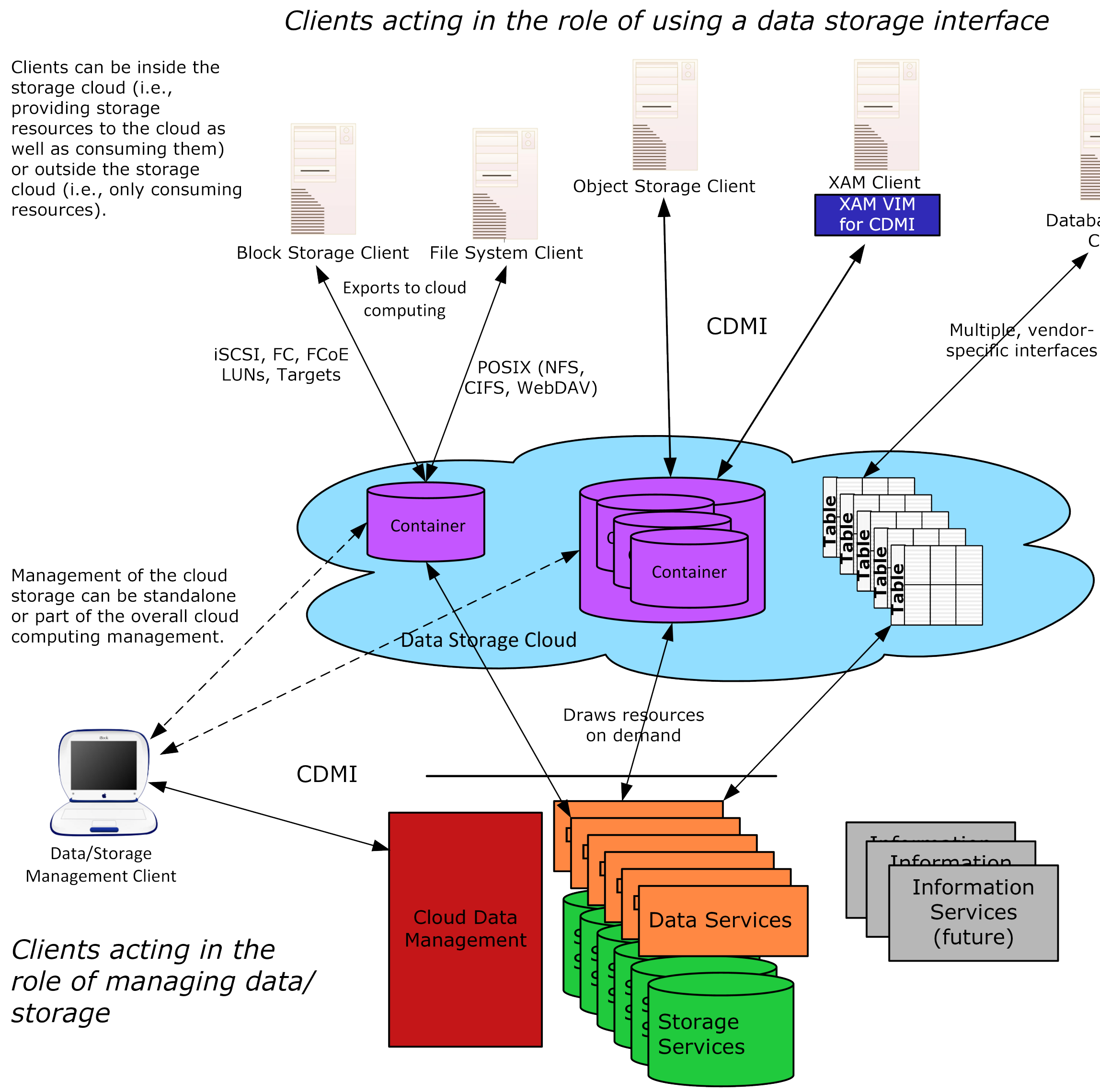
Clients acting in the role of using a data storage interface
Clients can be inside the storage cloud (i.e., providing storage resources to the cloud as well as consuming them) or outside the storage cloud (i.e., only consuming resources).
Block Storage Client
File System Client
Object Storage Client
XAM Client
XAM VIM for CDMI
Database/Table Client
Exports to cloud computing
iSCSI, FC, FCoE LUNs, Targets
POSIX (NFS, CIFS, WebDAV)
CDMI
Multiple, vendor-specific interfaces
Container
Container
Table
Data Storage Cloud
Management of the cloud storage can be standalone or part of the overall cloud computing management.
Draws resources on demand
CDMI
Data/Storage Management Client
Cloud Data Management
Data Services
Storage Services
Information Services (future)
Clients acting in the role of managing data/ storage

Fig. 5.3: 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).

5.3 Cloud Data Management Interface

The Cloud Data Management Interface (CDMI™) shown in Fig. 5.3 may 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 may also be used by administrative and management applications to manage containers, domains, security access, and monitoring/billing information, even for storage that is functionally accessible by legacy or proprietary protocols. The capabilities of the underlying storage and data services are exposed so that clients may understand what services the cloud provides.

Conformant cloud services may 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 Section 5.3.3, 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.

5.3.1 Object Model for CDMI

The model for CDMI is shown in Fig. 5.4.

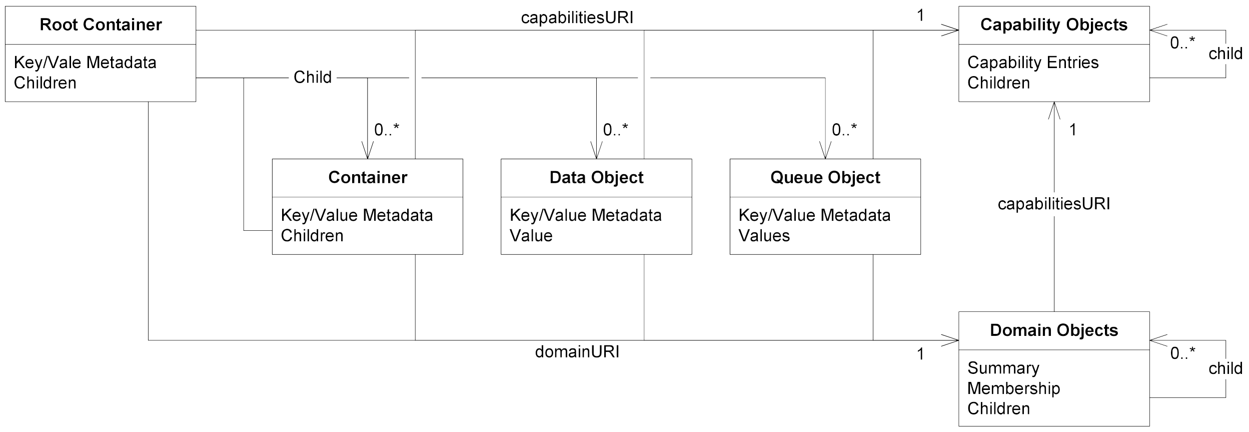

Fig. 5.4: CDMI Object Model

The five types of resources defined are shown in Table 5.1. The content type in any given operation is specific to each type of resource.

Table 5.1: 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 Section 5.1.4). Using the CDMI object model (see Fig. 5.4), the client may send a PUT via CDMI (see Fig. 5.3) 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 may 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 Fig. 5.4) and provide in-order-first in-first-out access to enqueued objects. More information on queues may 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 Section 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 Section 8.4). 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 Section 8.2). Subsequent to creation, objects are modified by performing PUTs using the object path specified by the client (see Section 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 Fig. 5.5.

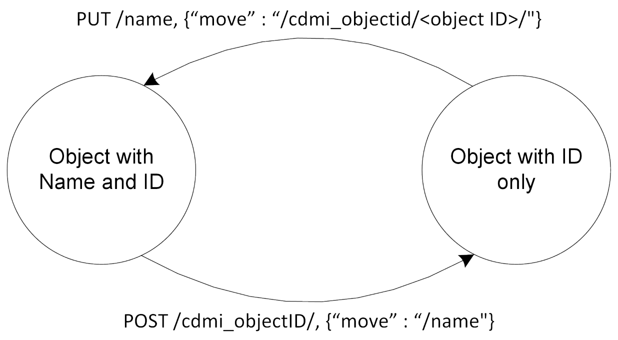
PUT /name, {"move" : "/cdmi_objectid/<object ID>/"}
Object with Name and ID
Object with ID only
POST /cdmi_objectID/, {"move" : "/name"}

Fig. 5.5: Object Transitions between Named and ID-only

5.3.2 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 5.2.

Table 5.2: 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

5.3.3 CDMI Object IDs

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

5.3.4 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 Table 5.3. 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. However, the object ID format is defined such that its integrity may be validated, and independent cloud services may assign unique object ID values independently.

Table 5.3: Object ID Format

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

The fields shown in Table 5.3 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 valida**

    - 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 `crc`

- Opaque data in each object ID shall be unique for a given Enterprise Number.

The native format for an object ID is binary. When necessary, such as when included in 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.

5.4 Security

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

- assures that the communications between a CDMI client and server may not 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 may be summarized as:

- transport security,
- user and entity authentication,
- authorization and access controls,
- data integrity,
- data and media sanitization,
- data retention,
- protections against malware,
- data at-rest encryption, and
- security capabilities.

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

When security is a concern, the CDMI client should begin with a series of security capability lookups (see Section 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).

5.4.1 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 “SNIA TLS Specification for Storage Systems”;
- shall support both HTTP over TLS and HTTP without TLS; and
- shall 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 may be required.

5.4.2 Client Authentication

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

CDMI clients may 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
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 may be used on an insecure network without TLS.
- HTTP status codes of `401 Unauthorized` should not include a choice of authentication.
- HTTP basic authentication and/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 and/or external) to validate client credentials. Differing authentication schemes may be supported, including host-based authentication, Kerberos, PKI, or other; the authentication service is beyond the scope of this international standard.

5.4.3 Use of TLS

Recommendations for using HTTP and TLS include:

- 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.

5.4.4 Further Information

For further information pertaining to storage security techniques, see `iso/iec_dis_27040`

5.5 Required HTTP Support

5.5.1 RFC 2616 Support Requirements

A conformant implementation of CDMI shall also be a conformant implementation of **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.

5.5.2 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 may 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 may contain "/" or a list of media types, or it may 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 may further qualify content negotiation (e.g., in Section 9.3, the absence of a Content-Type header has a specific meaning).

5.5.3 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.

5.5.4 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.

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",
        ...
    }
}
```

5.5.5 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 5.4 shows how relative URIs are resolved against root URIs

Table 5.4: 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 5.4.

- 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 may omit the parentID field and return an empty string (“”) for the value of the parentURI field.

5.5.6 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.

5.6 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 `ref_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.

5.7 Backwards Compatibility

CDMI client and server implementations shall implement the following measures to ensure backwards compability with earlier versions of this Interational Standard.

5.7.1 Specification Version Header

CDMI 2.x clients shall not include the X-CDMI-SPECIFICATION-VERSION custom header. When a CDMI 2.x client connects performs an operation against a CDMI 1.x Server, the absence of this header will result in an error response from the CDMI server. The client may use the presence of a X-CDMI-SPECIFICATION-VERSION header in an error response as an indication to down-negotiate to CDMI 1.x.

CDMI 2.x servers may use the presence of the X-CDMI-SPECIFICATION-VERSION custom header from a CDMI 1.x client to down-negotiate to CDMI 1.x.

See the CDMI 1.1.1 specification for more details on backwards compatiblity.

5.8 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.

- 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.

- 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
```

- 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
```

721 • 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"</P>
}
```

722 The following shows the response.

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

723 • DELETE to delete a reference:

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

724 The following shows the response.

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

Section II

Basic Cloud Storage

Clause 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 9.

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>/<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 may be performed when creating a new data object:

- Support for the ability to create a new data object is indicated by the presence of the `"cdmi_create_dataobject"` capability in the parent container.
- 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 CDMI data object using HTTP are shown in Table 6.1.

Table 6.1: Request Headers - Create a CDMI 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 CDMI 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 CDMI data object shall be set to “utf-8”. Otherwise, the valuetransferencoding field of the CDMI data object shall be set to “base64”. • If not specified, the mimetype field shall be set to “application/octet-stream”.	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 6.2.

Table 6.2: 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 may cause 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 may 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 CDMI data object using HTTP is shown in Table 6.3.

Table 6.3: Request Header - Read a CDMI 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 6.4.

Table 6.4: Response Headers - Read a CDMI 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 6.5.

Table 6.5: HTTP Status Codes - Read a CDMI 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
```

819 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 may be performed when updating an existing data object:

- Support for the ability to modify the value of an existing data object and/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 CDMI data object using HTTP are shown in Table 6.6.

Table 6.6: Request Headers - Update a CDMI 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 CDMI 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 6.7.

Table 6.7: Response Header - Update a CDMI 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 may 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 6.8.

Table 6.8: HTTP Status Codes - Update a CDMI 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 may cause 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
```

(continues on next page)

(continued from previous page)

```
Content-Length: 4

that
```

857 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 operations delete an existing data object at the specified URI:

- `DELETE <root URI>/<ContainerName>/<DataObjectName>`
- `DELETE <root URI>/cdmi_objectid/<DataObjectID>`

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.
- `<DataObjectID>` is the ID of the data object to be deleted.

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 may be provided as per **RFC 2616**.

6.5.4 Request Message Body

A request body may be provided as per **RFC 2616**.

6.5.5 Response Headers

Response headers may be provided as per **RFC 2616**.

6.5.6 Response Message Body

A response body may be provided as per **RFC 2616**.

6.5.7 Response Status

Table 6.9 describes the HTTP status codes that occur when deleting a data object using HTTP.

Table 6.9: HTTP Status Codes - Delete a CDMI 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 may cause a state transition error on the server or the data object may not be deleted.

6.5.8 Example

EXAMPLE 1: 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
```

Clause 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>/<ContainerObjectName>/`

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.
- `<ContainerObjectName>` 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 may 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 may be provided as per **RFC 2616**.

7.2.6 Response Message Body

A response body may be provided as per **RFC 2616**.

7.2.7 Response Status

Table 7.1 describes the HTTP status codes that occur when creating a container object using HTTP.

Table 7.1: 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 may cause a state transition error on the server.

7.2.8 Example

EXAMPLE 1: 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. Clause Section 9.3 describes how to read a container object using CDMI.

7.4 Update a Container Object using HTTP

Updating a container object using HTTP is not defined by this version of this international standard. Clause Section 9.4 describes how to update a container object using CDMI.

7.5 Delete a Container Object using HTTP

7.5.1 Synopsis

The following HTTP DELETE operations delete an existing container object at the specified URI, including all contained children and snapshots:

- `DELETE <root URI>/<ContainerName>/<ContainerObjectName>/`
- `DELETE <root URI>/cdmi_objectid/<ContainerObjectID>`

Where:

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

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 may be provided as per **RFC 2616**.

7.5.4 Request Message Body

A request body may be provided as per **RFC 2616**.

7.5.5 Response Headers

Response headers may be provided as per **RFC 2616**.

7.5.6 Response Message Body

A response body may be provided as per **RFC 2616**.

7.5.7 Response Status

Table 7.2 describes the HTTP status codes that occur when deleting a container object using HTTP.

Table 7.2: HTTP Status Codes - Delete a CDMI 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 may cause a state transition error on the server.

7.5.8 Example

EXAMPLE 1: 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 may be the user-provided file name.

7.6.2 Capabilities

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

- Support for the ability to create data objects through this operation is indicated by 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 Header

The HTTP request header for creating a new CDMI data object using HTTP is shown in Table 7.3.

Table 7.3: Request Header - 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 CDMI 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 CDMI data object shall be set to "utf-8". Otherwise, the valuetransferencoding field of the CDMI 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 CDMI data object using HTTP is shown in Table 7.4.

Table 7.4: Response Header - Create a New Data Object using HTTP

Heac	Type	Description	Re-quire-ment
Lo-ca-tion	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>.	Manda-tory

7.6.6 Response Message Body

A response body may be provided as per **RFC 2616**.

7.6.7 Response Status

Table 6.2 describes the HTTP status codes that occur when creating a new data object using HTTP.

Table 7.5: 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

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

Clause 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.

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

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 may be accessed by specifying a byte range after the value field name.

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 ";" may be specified, allowing multiple fields to be accessed in a single request.

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 may 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 may 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 may 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 may 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 may 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 may 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 Section 9.7.

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 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 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 may be performed when creating a new data object:

- Support for the ability to create a new data object is indicated by the presence of the cdmi_create_dataobject capability in the parent container.
- If the object being created in the parent container is a reference, support for that ability is indicated by the presence of the cdmi_create_reference capability in the parent container.

- If the new data object is a copy of an existing data object, support for the ability to copy is indicated by the presence of the cdmi_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 CDMI data object using CDMI are shown in Table 8.1.

Table 8.1: Request Headers for Creating a CDMI Data Object using CDMI

Header	Type	Description	Requirement
Accept	Header String	"application/cdmi-object" or a consistent value as per clause Section 5.5.2	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 `ref_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 8.2.

Table 8.2: Request Message Body - Create a Data Object using CDMI

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

Continued on next page

Table 8.2 – continued from previous page

Field Name	Type	Description	Requirement
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 10.3 . * If not specified, the domain of the parent container shall be used.	Optional
deserialize	JSON String	URI of a serialized CDMI data object that shall be deserialized to create the new data object	Optional[1]
serialize	JSON String	URI of a CDMI object that shall be serialized into the new data object	Optional[1]

Continued on next page

Table 8.2 – continued from previous page

Field Name	Type	Description	Requirement
copy	JSON String	URI of a source CDMI 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[1]

Continued on next page

Table 8.2 – continued from previous page

Field Name	Type	Description	Requirement
move	JSON String	URI of an existing local or remote CDMI 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[1]
reference	JSON String	URI of a CDMI 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[1]
deserializevalue	JSON String	A data object serialized as specified in RFC 4648.	Optional[1]

Continued on next page

Table 8.2 – continued from previous page

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[1]

Continued on next page

Table 8.2 – continued from previous page

Field Name	Type	Description	Requirement
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[1]

8.2.6 Response Headers

The HTTP response headers for creating a data object using CDMI are shown in Table 8.3.

Table 8.3: 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 8.4.

[1] 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 8.4: Response Message Body - Create a Data Object using CDMI

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
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 sys-	Mandatory

8.2.8 Response Status

The HTTP status codes that occur when creating a data object using CDMI are described in Table 8.5.

Table 8.5: 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 may cause a state transition error on the server.

8.2.9 Examples

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/",
```

(continues on next page)

(continued from previous page)

```
    "completionStatus" : "Complete",
    "mimetype" : "text/plain",
    "metadata" : {
        "cdmi_size" : "37"
    }
}
```

1177 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=="
}
```

1178 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"
   }
}
```

1179 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": {
```

(continues on next page)

(continued from previous page)

```
        "colour": "blue"
    }
}

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

<37 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p--
```

1180 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",
    ...
    }
}
```

1181 1. PUT to the container URI the data object name and binary contents using multi-part MIME with optional
1182 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
```

(continues on next page)

(continued from previous page)

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

<37 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p--
```

1183 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 may be performed when reading an existing data object:

- Support for the ability to read the metadata of an existing data object is indicated by the presence of the cdmi_read_metadata capability in the specified object.
- Support for the ability to read the value of an existing data object is indicated by the presence of the cdmi_read_value capability in the specified object.
- 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 CDMI data object using CDMI are shown in Table 8.6.

Table 8.6: Request Headers - Read a CDMI Data Object using CDMI

Header	Type	Description	Requirement
Accept	Header String	"application/cdmi-object", "multipart/mixed", or a consistent value as per clause Section 5.5.2	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 8.7.

Table 8.7: Response Headers - Read a CDMI 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 `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 CDMI data object using CDMI are shown in Table 8.8.

Table 8.8: Response Message Body - Read a Data Object using CDMI

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	Optional

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

8.3.7 Response Status

The HTTP status codes that occur when reading a data object using CDMI are described in Table 8.9.

Table 8.9: HTTP Status Codes - Read a CDMI 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

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",
```

(continues on next page)

(continued from previous page)

```
    "valuetransferencoding" : "utf-8",
    "value" : "This is the Value of this Data Object"
}
```

1233 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
```

1234 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"
}
```

1235 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
```

1236 The following shows the response.

1237 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
```

1238 The following shows the response.

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

```

(continues on next page)

(continued from previous page)

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

1239 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
```

1240 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--
```

1241 6. GET to the data object URI to read the data object using multi-part MIME, with optional content-lengths for the
1242 parts:

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

1243 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--
```

(a) 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.

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 may be performed when updating an existing data object:

- Support for the ability to modify the metadata of an existing data object is indicated by the presence of the cdmi_modify_metadata capability in the specified object.
- Support for the ability to modify the value of an existing data object and/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 CDMI data object using CDMI are shown in Table 8.10.

Table 8.10: Request Headers - Update a CDMI 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 8.11.

Table 8.11: Request Message Body - Update a CDMI Data Object using CDMI

Field Nam	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 10.3). * If not specified, the existing domain shall be preserved.	Optional
deserialize	JSON String	URI of a serialized CDMI 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[1]
copy	JSON String	URI of a source CDMI 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[1]
deserializevalue	JSON String	A data object serialized as specified in **RFC 4648**. The object ID of the serialized data object shall match the object ID of the destination data object.	Optional[1]
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. *	Optional[1]

8.4.5 Response Header

The HTTP response header for updating a data object using CDMI is shown in Table 8.12.

Table 8.12: Response Header - Update a CDMI Data Object using CDMI

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

8.4.6 Response Message Body

A response body may 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 8.13.

Table 8.13: HTTP Status Codes - Update a CDMI 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 may cause a state transition error on the server.

8.4.8 Examples

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"
    },
```

(continues on next page)

[1] 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`.

(continued from previous page)

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

The following shows the response.

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

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

```
PUT /MyContainer/MyDataObject.txt?mimetype 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
```

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.

1. PUT to the data object URI to replace all metadata with new metadata:

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

{
    "metadata" : {
        "colour" : "red",
```

(continues on next page)

(continued from previous page)

```
        "number" : "7"
    }
}
```

1304 The following shows the response.

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

1305 2. 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"
    }
}
```

1306 The following shows the response.

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

1307 3. 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"
    }
}
```

1308 The following shows the response.

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

1309 4. 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": {}
}
```

1310 The following shows the response.

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

5. 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
```

6. 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
```

7. 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
```

(continues on next page)

(continued from previous page)

```
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 CDMI data object using CDMI is shown in Table 8.14.

Table 8.14: Request Header - Delete a CDMI 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 may be provided as per **RFC 2616**.

8.5.5 Response Headers

Response headers may be provided as per **RFC 2616**.

8.5.6 Response Message Body

A response body may be provided as per **RFC 2616**.

8.5.7 Response Status

Table 8.15 describes the HTTP status codes that occur when deleting a data object using CDMI.

Table 8.15: HTTP Status Codes - Delete a CDMI 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-CDMIP access protocol lock or may cause a state transition error on the server or the data object may not be deleted.

8.5.8 Example

1. 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
```

Clause 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. 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.

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

http://cloud.example.com/container/?children</P>

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

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

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.

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 9.1).

Table 9.1: 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 `create_a_container_object_using_cdmi` .

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 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.
- 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 CDMI container object using CDMI are shown in Table 9.2.

Table 9.2: 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 Section 5.5.2	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 9.3.

1483 Table 9.3: Request Message Body - Create a Container Object using CDMI

1484

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	Optional

9.2.6 Response Headers

The HTTP response headers for creating a CDMI container object using CDMI are shown in Table 9.4.

Table 9.4: 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 CDMI container object using CDMI are shown in Table 9.5.

[1] 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 9.5: Response Message Body - Create a Container Object using CDMI

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
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	Mandatory

9.2.8 Response Status

Table 9.6 describes the HTTP status codes that occur when creating a container object using CDMI.

Table 9.6: HTTP Status Codes - Create a CDMI 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 may cause a state transition error on the server.

9.2.9 Example

1. PUT to the URI the container object name and metadata:

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

{
    "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.

[2] Returned only if present.

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

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 may 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 CDMI container object using CDMI are shown in Table 9.7.

Table 9.7: 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 Section 5.5.2	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 CDMI container object using CDMI are shown in *Response Headers - Read a Container Object using CDMI*.

Table 9.8: 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 CDMI container object using CDMI are shown in Table 9.9

1539 Table 9.9: Response Message Body - Read a Container Object using CDMI

1540

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 * 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	Optional

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

9.3.7 Response Status

`http_status_codes_read_a_container_object_using_cdmi` describes the HTTP status codes that occur when reading a container object using CDMI.

Table 9.10: 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

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/",
```

(continues on next page)

[1] Returned only if present.

(continued from previous page)

```
                "http://example.com/compute/1/"
            ]
        },
            "Network/NFSv4" : {
                "identifier" : "/users",
                "permissions" : "domain"
            },
            "childrenrange" : "0-4",
            "children" : [
                "red",
                "green",
                "yellow",
                "orange/",
                "purple/"
            ]
        }
}
```

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/"
    ]
}
```

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

{
```

(continues on next page)

(continued from previous page)

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

1555 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
```

1556 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 `ref_cdmi_snapshots`) 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 may 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 CDMI container object using CDMI are shown in Table 9.11.

Table 9.11: 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 9.12.

1604

Table 9.12: Request Message Body - Update a Container Object using CDMI

1605

Field Nam	Type	Description	Re-quire-ment
meta-data	JSON Ob-ject	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.	Op-tional
do-main-URI	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 10.3). * If not specified, the parent domain shall be used.	Op-tional
snap-shot	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.	Op-tional
de-se-ri-al-ize	JSON String	URI of a CDMI 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.	Op-tional[1]
copy	JSON String	URI of a CDMI container object that shall be copied into the existing container object. Only the contents of the container object itself shall be copied, not any children of the 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. 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.	Op-tional[1]
de-se-ri-al-ize-value	JSON Sting	A container object serialized as specified 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.	Op-tional[1]
ex-ports	JSON Ob-ject	A structure for each protocol that is enabled for this container object (see `exported_protocols` . If an exported protocol is added or changed, the PUT operation only returns after the export operation has completed.	Op-tional

[1] 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 CDMI container object using CDMI is shown in Table 9.13.

Table 9.13: 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 may be provided as per **RFC 2616**.

9.4.8 Response Status

Table 9.14 describes the HTTP status codes that occur when updating a container object using CDMI.

Table 9.14: 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 may cause a state transition error on the server.

9.4.9 Examples

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

{
```

(continues on next page)

(continued from previous page)

```
    "metadata" : {

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

1618 The following shows the response.

```

```

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

1619 2. PUT to the container object URI to set a new exported protocol value:

```
PUT /MyContainer/?exports 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"
        }
    }
}
```

1620 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 CDMI container object using CDMI is shown in Table 9.15.

Table 9.15: 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 may be provided as per **RFC 2616**.

9.5.5 Response Headers

Response headers may be provided as per **RFC 2616**.

9.5.6 Response Message Body

A response body may be provided as per **RFC 2616**.

9.5.7 Response Status

Table 9.16 describes the HTTP status codes that occur when deleting a container object using CDMI.

Table 9.16: 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 may cause a state transition error on the server.

9.5.8 Example

1. 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 `ref_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 may 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 may 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 CDMI data object using CDMI are shown in Table 9.17.

Table 9.17: 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 Section 5.5.2	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 Section 8.3. * 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 9.18.

Table 9.18: Request Message Body - Create a New Data Object Using CDMI

Field Nam	Type	Description	Re-quire-ment
mime-type	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.	Op-tional
meta-data	JSON Ob-ject	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.	Op-tional
do-main-URI	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_required_settings_for_domain_member_user_objects` . * If not specified, the root domain “/cdmi_domains/” shall be used.	Op-tional
de-se-ri-al-ize	JSON String	URI of a CDMI data object that shall be deserialized to create the new data object	Op-tional[1]
se-ri-al-ize	JSON String	URI of a CDMI object that shall be serialized into the new data object	Op-tional[1]
copy	JSON String	URI of a CDMI data object or queue object that shall be copied into the new data object	Op-tional[1]
move	JSON String	URI of a CDMI 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.	Op-tional[1]
ref-er-ence	JSON String	URI of a CDMI 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`.	Op-tional[1]
de-se-ri-al-ize-value	JSON String	A data object serialized as specified 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.	Op-tional[1]
val-ue-trans-fer-en-cod-ing	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”	Op-tional

9.6.6 Response Headers

The HTTP response headers for creating a new CDMI data object using CDMI are shown in Table 9.19.

Table 9.19: 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 CDMI data object using CDMI are shown in Table 9.20.

[1] 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 9.20: Response Message Body - Create a New Data Object Using CDMI

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	Optional

9.6.8 Response Status

Table 9.21 describes the HTTP status codes that occur when creating a new data object using CDMI.

Table 9.21: 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 may cause a state transition error on the server.

9.6.9 Examples

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/",
```

(continues on next page)

(continued from previous page)

```
    "capabilitiesURI" : "/cdmi_capabilities/dataobject/",
    "completionStatus" : "Complete",
    "mimetype" : "text/plain",
    "metadata" : {
        ...
    }
}
```

1748 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"
}
```

1749 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"
            }
        ],
                ...
    }
}
```

1750 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

```

(continues on next page)

(continued from previous page)

```
    --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.

.. code-block:: http

    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 `ref_queue_object_resource_operations_using_cdmi`) 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 `ref_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 may 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 may 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 CDMI queue object using CDMI are shown in Table 9.22.

Table 9.22: 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 Section 5.5.2	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 9.23.

Table 9.23: Request Message Body - Create a New Queue Object Using CDMI

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_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 CDMI data object that will be deserialized to create the new queue object	Optional[1]
copy	JSON String	URI of a CDMI queue object that will be copied into the new queue object	Optional[1]
move	JSON String	URI of a CDMI queue object that will be copied into the new queue object. When the copy is successfully completed, the queue object at the source URI is removed.	Optional[1]
reference	JSON String	URI of a CDMI 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[1]
deserializevalue	JSON String	A queue object serialized as specified in **RFC 4648**	Optional[1]

9.7.6 Response Headers

The response headers for creating a new CDMI queue object using CDMI are shown in Table 9.24.

Table 9.24: 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 CDMI queue object using CDMI are shown in Table 9.25.

[1] 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`.

1836

Table 9.25: Response Message Body - Create a New Queue Object Using CDMI

1837

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 * 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	Optional

9.7.8 Response Status

Table 9.26 describes the HTTP status codes that occur when creating a new queue object using CDMI.

Table 9.26: 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

1. 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" : {
                ...
```

(continues on next page)

(continued from previous page)

```
    },
    "queueValues" : ""
}
```

Section IV

CDMI Advanced

Clause 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.

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.

1. 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.

1. 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 10.1).

Table 10.1: .*

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".	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 Section 16.4 and Section 16.5 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 Section 16.1) 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.

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.

1. 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.

1. 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 10.2 shall be present within the contents of each domain summary object, which are in JSON representation.

Table 10.2: .*

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
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_objects	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_bytes	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 `ref_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

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

```
{
    "cdmi_domainURI" : "/cdmi_domains/MyDomain/",
    "cdmi_summary_start" : "2009-12-10T00:00:00",
```

(continues on next page)

(continued from previous page)

```
    "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 Section 16.1), 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 Section 10.1.4). 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 Section 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 10.3 lists the domain settings that shall be present within each domain member user object.

Table 10.3: .*

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 contains a JSON list of special privileges associated with the user or "group". 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 10.4 lists the domain settings that shall be present within each domain member delegation object.

Table 10.4: .*

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

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

```
{
    "cdmi_member_enabled" : "true",
```

(continues on next page)

(continued from previous page)

```
    "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"
    ]
}
```

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 may be performed when creating a new domain:

- Support for the ability to create a new domain object is indicated by the presence of the cdmi_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 CDMI domain object using CDMI are shown in Table 10.5

Table 10.5: .*

Header	Type	Description	Requirement
Accept	Header String	"application/cdmi-domain" or a consistent value as per clause Section 5.5.2	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 `ref_request_message_body_create_a_domain_object_using_cdmi`.

Table 10.6: Request Message Body Create a Domain Object using CDMI

Field Nam	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[1]
move	JSON String	URI of an existing local CDMI 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[1]
deserialize	JSON String	URI of a serialized CDMI data object that shall be deserialized to create the new domain, including all child objects inside the source serialized data object	Optional[1]
deserialize-value	JSON String	A domain object serialized as specified in **RFC 4648**.	Optional[1]

10.2.5 Response Headers

The HTTP response headers for creating a domain object using CDMI are shown in Table 10.7

Table 10.7: .*

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

[1] 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`.

10.2.6 Response Message Body

The response message body fields for creating a domain object using CDMI are shown in Table 10.8

Table 10.8: .*

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

`ref_response_status_create_a_domain_object_using_cdmi` describes the HTTP status codes that occur when creating a domain object using CDMI.

Table 10.9: .*

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 may cause a state transition error on the server.

10.2.8 Example

1. PUT to the domain URI the domain name and metadata:

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

"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 may be performed when reading an existing domain:

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

10.3.3 Request Headers

The HTTP request headers for reading a CDMI domain object using CDMI are shown in `tbl_cdmi_domain_object_read_request_headers`

.._list-table:: Request Headers - Read a Domain Object using CDMI

header-rows 1

widths auto

align center

- – Header
  – Type
  – Description
  – Requirement

- – Accept
  – Header String
  – "application/cdmi-domain" or a consistent value as per clause Section 5.5.2
  – 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 CDMI domain object using CDMI are shown in `ref_response_headers_read_a_domain_object_using_cdmi`.

Table 10.10: .*

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 CDMI domain object using CDMI are shown in Table 10.11

Table 10.11: .*

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

`ref_response_status_read_a_domain_object_using_cdmi` describes the HTTP status codes that occur when reading a domain object using CDMI.

Table 10.12: 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

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/"
    ]
}
```

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/"
    ]
}
```

2103 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
```

2104 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 CDMI domain object using CDMI are shown in Table 10.13.

Table 10.13: .*

Header	Type	Description	Require-ment
Content-Type	Header String	“application/cdmi-domain”	Manda-tory
X-CDMI-Specification-Version	Header String	A comma-separated list of versions that the client supports, e.g., “1.1, 1.5, 2.0”	Manda-tory

10.4.4 Request Message Body

The request message body fields for updating a domain object using CDMI are shown in Table 10.14.

Table 10.14: Request Message Body - Update a domain object using CDMI

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 CDMI 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[2]
deserialize	JSON String	URI of a serialized CDMI 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[2]
deserializevalue	JSON String	A domain object serialized as specified 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[2]

10.4.5 Response Header

The HTTP response header for updating a CDMI domain object using CDMI is shown in `response_header_update_a_domain_object_using_cdmi`

Table 10.15: .*

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 may be provided as per **RFC 2616**.

10.4.7 Response Status

`http_status_codes_update_a_domain_object_using_cdmi` describes the HTTP status codes that occur when updating a domain object using CDMI.

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

Table 10.16: .*

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 may cause a state transition error on the server.

10.4.8 Example

1. 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 Headers

The HTTP request header for deleting a CDMI domain object using CDMI is shown in `request_headers_delete_a_domain_object_using_cdmi`

Table 10.17: .*

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 may be provided as per **RFC 2616**.

10.5.5 Response Headers

Response headers may be provided as per **RFC 2616**.

10.5.6 Response Message Body

A response body may be provided as per **RFC 2616**.

10.5.7 Response Status

`http_status_codes_delete_a_domain_object_using_cdmi` describes the HTTP status codes that occur when deleting a domain object using CDMI.

Table 10.18: .*

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 may cause a state transition error on the server.

10.5.8 Example

1. 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
```

Clause 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 value(s) from the queue object and subsequently deletes the value(s) to acknowledge receipt of the value(s) 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 `ref_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.

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.

1. 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.

1. 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 `ref_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 may be performed when creating a new queue object:

- Support for the ability to create a new queue object is indicated by the presence of the cdmi_create_queue capability in the parent container.
- If the object being created in the parent container is a reference, support for that ability is indicated by the presence of the cdmi_create_reference capability in the parent container.
- If the new queue object is a copy of an existing queue object, support for the ability to copy is indicated by the presence of the cdmi_copy_queue capability in the 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 CDMI queue object using CDMI are shown in Table 9.22

.._list-table:: Request Headers - Create A Queue Object Using CDMI

header-rows 1

widths auto

align center

- - Header
  - Type
  - Description
  - Requirement
- - Accept
  - Header String
  - “application/cdmi-queue” or a consistent value as per per clause Section 5.5.2
  - 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 9.23.

2307 Table 11.1: Request Message Body - Create A Queue Object Using CDMI

2308

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 10.3). • If not specified, the parent domain shall be used.	Optional

11.2.6 Response Status

The HTTP response headers for creating a CDMI queue object using CDMI are shown in Table 9.24

Table 11.2: 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 CDMI queue object using CDMI are shown in Table 9.25

[1] 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 11.3: Response Message Body - Create A Queue Object Using CDMI

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
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	Mandatory

11.2.8 Response Status

Table 9.26 describes the HTTP status codes that occur when creating a queue object using CDMI.

Table 11.4: 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 may cause a state transition error on the server.

11.2.9 Examples

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" : {
```

(continues on next page)

(continued from previous page)

```
        ...
    },
    "queueValues" : ""
}
```

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 queue-Values range is empty.

11.3.2 Capabilities

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

- Support for the ability to read the metadata of an existing queue object is indicated by the presence of the cdmi_read_metadata capability in the specified queue 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 CDMI queue object using CDMI are shown in Table 11.5

Table 11.5: 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 Section 5.5.2 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 Section 8.4.	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 Status

The HTTP response headers for reading a CDMI queue object using CDMI are shown in Table 11.6.

Table 11.6: 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 CDMI queue object using CDMI are shown in Table 11.7

Table 11.7: .*

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 * 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,	Optional

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

11.3.7 Response Status

Table 11.8 describes the HTTP status codes that occur when reading a queue object using CDMI.

Table 11.8: 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

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"
    ],
```

(continues on next page)

(continued from previous page)

```
    "value": [
        "First Enqueued Value"
    ]
}
```

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"
    ]
}
```

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"
    ]
}
```

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
```

(continues on next page)

(continued from previous page)

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

2388 1. 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
```

2389 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"
    ]
```

(continues on next page)

(continued from previous page)

```
}

--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 CDMI queue object using CDMI are shown in Table 11.9

Table 11.9: 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 9.23.

Table 11.10: Request Message Body - Update A Queue Object Using CDMI

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 10.3). * If not specified, the existing domain shall be preserved.	Optional
deserialize	JSON String	URI of a serialized CDMI 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[2]
copy	JSON String	URI of a source CDMI 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 `ref_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[2]
deserializevalue	JSON String	A queue object serialized as specified 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[2]

11.4.5 Response Header

The HTTP response header for updating a CDMI queue object using CDMI is shown in Table 11.11

Table 11.11: 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

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

11.4.6 Response Message Body

A response body may be provided as per RFC 2616.

11.4.7 Response Status

Table 11.12 describes the HTTP status codes that occur when updating a queue object using CDMI.

Table 11.12: 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 may cause a state transition error on the server.

11.4.8 Examples

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
```

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 CDMI queue object using CDMI is shown in Table 11.13

Table 11.13: 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 may be provided as per **RFC 2616**.

11.5.5 Response Headers

Response headers may be provided as per **RFC 2616**.

11.5.6 Response Message Body

A response body may be provided as per **RFC 2616**.

11.5.7 Response Status

Table 11.14 describes the HTTP status codes that occur when deleting a queue object using CDMI.

Table 11.14: 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 may cause a state transition error on the server.

11.5.8 Example

1. 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 may be performed when enqueuing a new value into an existing queue object:

- Support for the ability to modify the value of an existing queue object is indicated by the presence of the cdmi_modify_value capability in the specified queue 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 CDMI queue object value using CDMI are shown in Table 11.15

Table 11.15: 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 Section 8.3.	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 11.16.

Table 11.16: Request Message Body - Enqueue A New Queue Object Value Using CDMI

Field Name	Type	Description	Requirement
mimetype	JSON Array of JSON Strings	MIME type(s) of the data value(s) 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 multipart 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
copy	JSON String	URI of a source CDMI 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[3]
move	JSON String	URI of a source CDMI data object or queue object from which the value shall be moved and	Optional[3]

11.6.5 Response Headers

Response headers may be provided as per RFC 2616.

11.6.6 Response Message Body

A response body may be provided as per RFC 2616.

11.6.7 Response Status

tbl_cdmi_queue_object_value_enqueue_response_status describes the HTTP status codes that occur when enqueuing a new queue object using CDMI.

Table 11.17: 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 may cause a state transition error on the server.

11.6.8 Examples

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
```

[3] 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.

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
```

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
```

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
```

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
```

(continues on next page)

(continued from previous page)

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

{
   "mimetype": [
       "text/plain",
       "text/plain"
   ],
   "valuetransferencoding": [
       "utf-8",
       "base64"
   ],
   "value": [
       "First",
       "U2Vjb25k"
   ]
}
```

2508 The following shows the response.

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

2509 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--
```

2510 The following shows the response.

```
HTTP/1.1 204 No content
```

2511 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
```

(continues on next page)

(continued from previous page)

```
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--
```

2512 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 `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 CDMI queue object value using CDMI is shown in Table 11.18.

Table 11.18: 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 may be provided as per RFC 2616.

11.7.5 Response Headers

Response headers may be provided as per RFC 2616.

11.7.6 Response Message Body

A response body may be provided as per RFC 2616.

11.7.7 Response Status

Table 11.19 describes the HTTP status codes that occur when deleting a queue object value using CDMI.

Table 11.19: 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 may cause a state transition error on the server.

11.7.8 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
```

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
```

2560 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
```

2561 The following shows the response.

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

Clause 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 Section 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 Section 12.1.1 Vendor-defined capabilities not specified in this international standard shall not start with "cdmi_".

Fig. 12.1 shows the hierarchy of capabilities and shows how the capabilitiesURI links data objects and container objects into the capabilities tree.

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.

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.

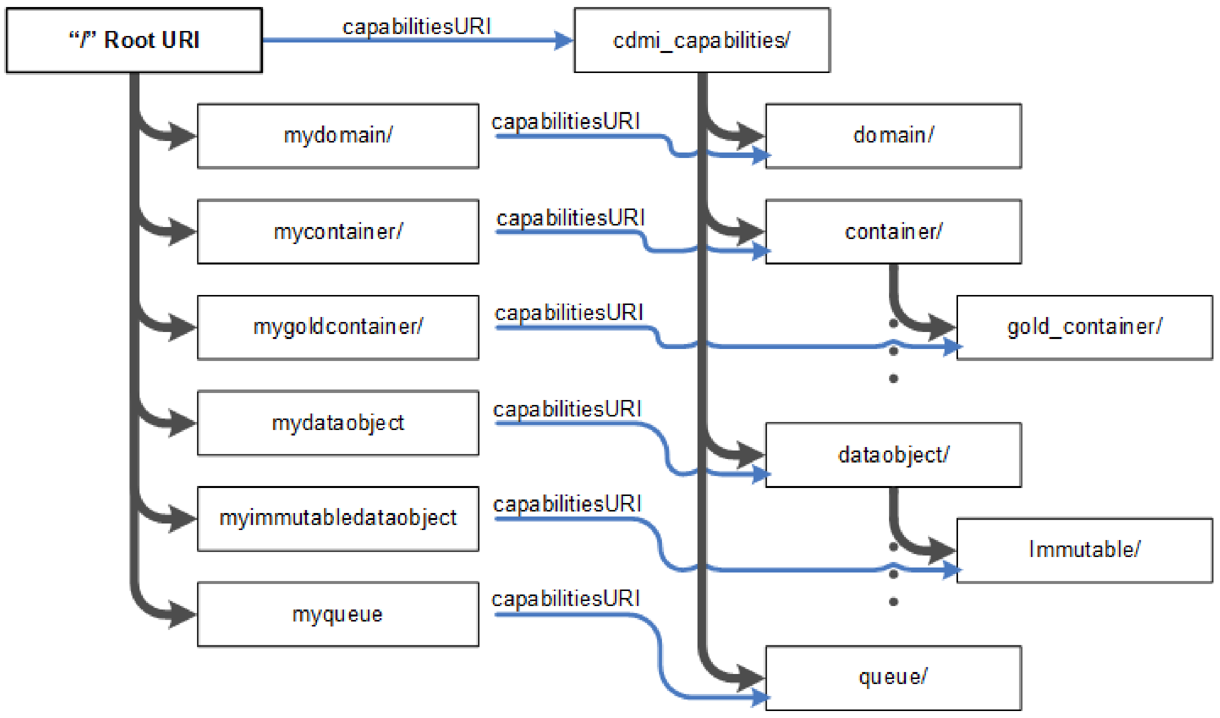

Fig. 12.1: Hierarchy of Capabilities

2592 ### 12.1.1 Cloud Storage System-Wide Capabilities

2593 Table 12.1 defines the system-wide capabilities in a cloud storage system. These capabilities, which are found in the
2594 capabilities object, are referred to by the root URI (root capabilities).

Capability Name	Type	Definition
cdmi_domains	JSON String	If present and "true", indicates that the cloud storage system supports domains.
cdmi_export_cifs	JSON String	If present and "true", this capability indicates that the cloud storage system sup
cdmi_dataobjects	JSON String	If present and "true", this capability indicates that the cloud storage system sup
cdmi_export_iscsi	JSON String	If present and "true", this capability indicates that the cloud storage system sup
cdmi_export_nfs	JSON String	If present and "true", this capability indicates that the cloud storage system sup
cdmi_export_occi_iscsi	JSON String	If present and "true", this capability indicates that the cloud storage system sup
cdmi_export_webdav	JSON String	If present and "true", this capability indicates that the cloud storage system sup
cdmi_metadata_maxitems	JSON String	If present, this capability indicates the maximum number of user-defined metac
cdmi_metadata_maxsize	JSON String	If present, this capability indicates the maximum size, in bytes, of each user-de
cdmi_metadata_maxtotalsize	JSON String	If present, this capability indicates the maximum size, in bytes, of user-defined
cdmi_notification	JSON String	If present and "true", this capability indicates that the cloud storage system sup
cdmi_logging	JSON String	If present and "true", this capability indicates that the cloud storage system sup
cdmi_query	JSON String	If present and "true", this capability indicates that the cloud storage system sup
cdmi_query_regex	JSON String	If present and "true", this capability indicates that the cloud storage system sup
cdmi_query_contains	JSON String	If present and "true", this capability indicates that the cloud storage system sup
cdmi_query_tags	JSON String	If present and "true", this capability indicates that the cloud storage system sup
cdmi_query_value	JSON String	If present and "true", this capability indicates that the cloud storage system sup

Capability Name	Type	Definition
cdmi_queues	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_security_access_control	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_security_audit	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_security_data_integrity	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_security_encryption	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_security_immutability	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_security_sanitization	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_serialization_json	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_snapshots	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_references	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_object_move_from_local	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_object_move_from_remote	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_object_move_from_ID	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_object_move_to_ID	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_object_copy_from_local	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_object_copy_from_remote	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_object_access_by_ID	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_post_dataobject_by_ID	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_post_queue_by_ID	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_deserialize_dataobject_by_ID	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_deserialize_queue_by_ID	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_serialize_dataobject_to_ID	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_serialize_domain_to_ID	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_serialize_container_to_ID	JSON String	If present and “true”, this capability indicates that the cloud storage system allo
cdmi_serialize_queue_to_ID	JSON String	If present and “true”, this capability indicates that the cloud storage system allo
cdmi_copy_dataobject_by_ID	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_copy_queue_by_ID	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_create_reference_by_ID	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_copy_dataobject_from_queue	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_multipart_mime	JSON String	If present and “true”, this capability indicates that the cloud storage system sup
cdmi_create_value_range_by_ID	JSON String	If present and “true”, this capability indicates that the system allows a new data

12.1.2 Storage System Metadata Capabilities

Table 12.2 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. See Section 16.3 for a description of these storage system metadata items.

Table 12.2: 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 12.2 of Section 12.1.1 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 12.3 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 Section 16.4 for a description of the meaning of the corresponding data system metadata items.

Table 12.3: Capabilities for Data System Metadata

Capability Name	Type	Definition
cdmi_assignedsize	JSON String	**When the cloud storage system supports the cdmi_as** 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 Section 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 Section 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 Section 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 Section 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 sup-

12.1.4 Data Object Capabilities

Table 12.4 defines the capabilities for data objects in a cloud storage system.

Table 12.4: Capabilities for Data 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 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.
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 12.5 defines the capabilities for containers in a cloud storage system.

Table 12.5: Capabilitie

Capability Name	Type	Definition
cdmi_list_children	JSON String	If present and "true", this capability indicates that the cloud storage system sha
cdmi_list_children_range	JSON String	If present and "true", this capability indicates that the cloud storage system sha
cdmi_read_metadata	JSON String	If present and "true", this capability indicates that the cloud storage system sha
cdmi_modify_metadata	JSON String	If present and "true", this capability indicates that the cloud storage system sha
cdmi_modify_deserialize_container	JSON String	If present and "true", this capability indicates that the cloud storage system sha
cdmi_snapshot	JSON String	If present and "true", this capability indicates that the cloud storage system sha
cdmi_serialize_dataobject	JSON String	If present and "true", this capability indicates that the cloud storage system sha
cdmi_serialize_container	JSON String	If present and "true", this capability indicates that the cloud storage system sha
cdmi_serialize_queue	JSON String	If present and "true", this capability indicates that the cloud storage system sha
cdmi_serialize_domain	JSON String	If present and "true", this capability indicates that the cloud storage system sha
cdmi_deserialize_container	JSON String	If present and "true", this capability indicates that the cloud storage system sha
cdmi_deserialize_queue	JSON String	If present and "true", this capability indicates that the cloud storage system sha
cdmi_deserialize_dataobject	JSON String	If present and "true", this capability indicates that the cloud storage system sha
cdmi_create_dataobject	JSON String	If present and "true", this capability indicates that the cloud storage system sha
cdmi_post_dataobject	JSON String	If present and "true", this capability indicates that the cloud storage system sha
cdmi_post_queue	JSON String	If present and "true", this capability indicates that the cloud storage system sha
cdmi_create_container	JSON String	If present and "true", this capability indicates that the cloud storage system sha
cdmi_create_queue	JSON String	If present and "true", this capability indicates that the cloud storage system sha
cdmi_create_reference	JSON String	If present and "true", this capability indicates that the cloud storage system sha

Table 12.5 – continued f

Capability Name	Type	Definition
cdmi_export_container_cifs	JSON String	If present and “true”, this capability indicates that the cloud storage system sha
cdmi_export_container_nfs	JSON String	If present and “true”, this capability indicates that the cloud storage system sha
cdmi_export_container_iscsi	JSON String	If present and “true”, this capability indicates that the cloud storage system sha
cdmi_export_container_occi	JSON String	If present and “true”, this capability indicates that the cloud storage system sha
cdmi_export_container_webdav	JSON String	If present and “true”, this capability indicates that the cloud storage system sha
cdmi_delete_container	JSON String	If present and “true”, this capability indicates that the cloud storage system sha
cdmi_move_container	JSON String	If present and “true”, this capability indicates that the cloud storage system sha
cdmi_copy_container	JSON String	If present and “true”, this capability indicates that the cloud storage system sha
cdmi_move_dataobject	JSON String	If present and “true”, this capability indicates that the cloud storage system sha
cdmi_copy_dataobject	JSON String	If present and “true”, this capability indicates that the cloud storage system sha
cdmi_create_value_range	JSON String	If present and “true”, this capability indicates that the container allows a new da

12.1.6 Domain Object Capabilities

Table 12.6 defines the capabilities for domains in a cloud storage system. (All capabilities refer to what may be done via CDMI content-type operations.

Table 12.6: Capabilities for Domain Objects

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.
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 12.7 defines the capabilities for queue objects in a cloud storage system.

Table 12.7: 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 CDMI capabilities object using CDMI are shown in Table 12.8.

Table 12.8: 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 Section 5.5.2	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 CDMI capabilities object using CDMI are shown in Table 12.9.

Table 12.9: 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 CDMI capabilities object using CDMI are shown in Table 12.10.

Table 12.10: 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 12.11 describes the HTTP status codes that occur when reading a capabilities object using CDMI.

Table 12.11: 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

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/",
```

(continues on next page)

(continued from previous page)

```
        "queue/"
    ]
}
```

2669 1. 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
```

2670 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/"
    ]
}
```

2671 2. 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
```

2672 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/"
    ]
}
```

Clause 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 Fig. 13.1 shows.

Fig. 13.1: 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 `ref_cloud_storage_systemwide_capabilities` 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 `ref_mapping_names_from_cdmi_to_another_protocol`) 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:

- 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 `ref_user_and_groupname_mapping_syntax_and_evaluation_rules` 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.

13.2.2 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.

13.2.3 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.

13.2.4 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.

13.2.5 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.

The mapping information is only required on the container being exported.

13.2.6 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" : {<BR>
    "hosts" : { "*.mycollege.edu", "derf.cs.myuni.edu" },
            "domain" : "lab.mycollege.edu",
            "usermap" : {
                { <cdminame>, <map>, <nfsname> },
                { "jimsmith", "<-->", "jims" },
                { [ordered list of CDMIname/operator/NFSname triples] },
                { "*", "<-->", "*" }
            }
            "groupmap" : {
                { "admins", "<-", "wheel" },
                { "everyone", "<-", "*" }
            }
        }
        "cifs" : {
            "hosts" : "*",
            "domain" : "lab.mycollege.edu",
            "usermap" : {
                { "jimsmith", "<-->", "james.smith" }
                { [ordered list of CDMIname/operator/NFSname triples] },
                { "*", "<-->", "*" }
            }
            "groupmap" : {
                { "admins", "<-", "Administrators" },
                { "everyone", "<-", "*" }
            }
        }
    }
}
```

The following shows the response.

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

(continues on next page)

(continued from previous page)

```
{
    "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> }
}
```

13.2.7 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.

1. 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"
                ],
                [
                    "*",
                    "<-->",
                    "*"
                ]
            ]
        }
```

(continues on next page)

(continued from previous page)

```
    }
}
```

13.2.8 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 `ref_user_and_groupname_mapping_syntax_and_evaluation_rules`.

13.2.9 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.

13.3.1 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 13.1.

Table 13.1: .*

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 13.2.

Table 13.2: .*

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).
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.

1. Example

```
"exports":
    { "nfs":
        {
            ...
            {"no_wdelay": "true" },
            {"refer": "otherserver://path/leaf"},
            ...
        }
    }
```

13.4.1 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 13.3

Table 13.3: Required Members of the CIFS protocol structure

Member	Description
share_name	The name that CIFS shall use to discover the share.
export-domain	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 “`ref_export_control`” in the subclause on NFS Exports (see `ref_nfs_exported_protocol`).

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.

1. 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 `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 (`ref_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.

1. 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 `ref_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.

1. 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.

Clause 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 Fig. 14.1).

Fig. 14.1: Snapshot Container Structure

A snapshot operation is requested using the container update operation (see Section 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.

Clause 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 CDMI 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.

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

1. 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",
```

(continues on next page)

(continued from previous page)

```
            "mimetype": [
                "text/plain",
                "text/plain"
            ],
            "valuetransferencoding": [
                "utf-8",
                "utf-8"
            ],
            "valuerange": [
                "0-2",
                "0-3"
            ],
            "value": [
                "red",
                "blue"
            ]
        }
    ]
}
```

3029 To allow efficient deserialization in stream mode when serializing container objects to JSON, the children array should
3030 be the last item in the canonical serialized JSON format.

Clause 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 `ace_types` `ref_ace_flags`, and `ref_ace_bit_masks`.

16.1.2 ACE Types

Table 16.1 defines the following ACE types, following NFSv4.

Table 16.1: 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

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 :ref'tbl_who_identifiers'). 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 16.2: 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 16.3 govern this behavior.

Table 16.3: ACE Flags

String Form	Description	Constant	Bit Mask
"NO_FLAGS"	No flags are set	CDMI_ACE_FLAGS_NONE	0x00000
"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_INHER	0x00001
"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_IN	0x00002
"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	0x00004
"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_	0x00008
"IDENTIFIER_GROUP"	An ACE on which IDENTIFIER_GROUP is set indicates that the "who" refers to a group identifier.	CDMI_ACE_FLAGS_IDENTIFIER_GR	0x00040
"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_ACI	0x00080

16.1.5 ACE Mask Bits

3072 The mask field of an ACE contains 32 bits. RFC 3530.

Table 16.4: ACE Bit Masks

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 and/or childrenrange field shall return an HTTP status code of 403 Forbidden.	CDMI_ACE_LIST_CONTAINE	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_SUBCONTA	0x00000004
“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_METADAT	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_METADA	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_CON	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_SUBCO	0x00000040
“READ_ATTRIBUTES”	Permission to read the attribute fields[#a]_ 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_ATTRIBUT	0x00000080
“WRITE_ATTRIBUTES”	Permission to change attribute fields[#a]_ 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_ATTRIBU	0x00000100

[#a]_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 th**

    - 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 `ace_types` `ref_ace_flags`, and `ref_ace_bit_masks`, 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

1. Example

   ```
   "READ_ALL" | 0x02
   ```

   evaluates to 0x09 | 0x02 == 0x0

2. Example

   ```
   0x001F07FF
   ```

   evaluates to 0x001F07FF == “ALL_PERMS”

3. Example

   ```
   "RW_ALL" | DELETE
   ```

   evaluates to 0x000601DF | 0x00100000 == 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:

Table 16.5: ACE Bit Masks

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.
4. 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.
```

1. 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 `storage_system_metadata` shall be generated by the cloud storage system and shall immediately be made available to a CDMI client in the metadata that is returned as a result of the create operation and any subsequent retrievals.

Table 16.6: .*

Metadata Nam	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
cdmi_ctime	JSON String	The time when the object was created, in ISO-8601 point-in-time format, as described in `ref_time_representations`. 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 `ref_time_representations`. 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 `ref_time_representations`. 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
cdmi_owner	JSON String	The name of the principal that has owner privileges for the object.	Mandatory
cdmi_acl	JSON Array of JSON Objects	Standard ACL metadata. If not specified when the object is created, this metadata shall be filled in by the system.	Optional

16.4 Support for Data System Metadata

When specified, data system metadata provides guidelines to the cloud storage system on how to provide storage data services for data managed through the CDMI interface.

Data system metadata (see Table 16.7 is inherited from parent objects to any children. If a child explicitly contains data system metadata, the metadata value of the child data system metadata shall override the metadata value of the parent data system metadata.

Table 16.7: Data System Metadata

Metadata Name	Type	Description	Requirement
cdmi_data_redundancy	JSON String	If this data system metadata item is present and set to a positive numeric string, it indicates that the client is requesting a desired number of complete copies. Additional copies may be made to satisfy demand for the value. When this data system metadata item is absent, or is present and is not set to a positive numeric string, this data system metadata item shall not be used.	Optional
cdmi_immediate_redundancy	JSON String	If this data system metadata item is present and set to "true", it indicates that the client is requesting that at least the number of copies indicated in cdmi_data_redundancy contain the newly written value before the operation completes. This metadata is used to make sure that multiple copies of the data are written to permanent storage to prevent possible data loss. When this data system metadata item is absent, or is present and is not set to "true", this data system metadata item shall not be used. If the requested number of copies cannot be created within the HTTP timeout period, the transaction shall complete, but the cdmi_immediate_redundancy_provided data system metadata shall be set to "false".	Optional
cdmi_assignedsize	JSON String	If this data system metadata item is present and set to a positive numeric string, it indicates that the client is specifying the size in bytes that is desired to be reported for a container object exported via other protocols (see `ref_container_metadata`). The system is not required to reserve this space and may thin-provision the requested space. Thus,	Optional

16.5 Support for Provided Data System Metadata

For each metadata item in a data system, there is an actual value that the cloud service is able to achieve at this time, as shown in Table 16.8 Data system-provided metadata items are read only. Updates of these metadata items shall be ignored.

Table 16.8: .*

Metadata Name	Type	Description	Requirement
cdmi_data_redundancy_provided	JSON String	Contains the current number of complete copies of the data object at this time	Optional
cdmi_immediate_redundancy_provided	JSON String	If present and set to "true", indicates if immediate redundancy is provided for the object	Optional
cdmi_infrastructure_redundancy_provided	JSON String	Contains the current number of independent storage infrastructures supporting the data currently operating	Optional
cdmi_data_dispersion_provided	JSON String	Contains the current lowest distance (km) between any two infrastructures hosting the data	Optional
cdmi_geographic_placement_provided	JSON Array of JSON Strings	Contains an ISO-3166 identifier that corresponds to a geopolitical region where the object is stored	Optional
cdmi_retention_period_provided	JSON String	Contains an `ref_iso_8601:2004` time interval (as described in `ref_time_representations`) specifying the period the object is protected by retention	Optional
cdmi_retention_autodelete_provided	JSON String	Contains "true" if the object will automatically be deleted when retention expires	Optional
cdmi_hold_id_provided	JSON Array of JSON Strings	Contains the user-specified hold identifiers for active holds	Optional
cdmi_encryption_provided	JSON String	Contains the algorithm used for encryption, the mode of operation, and the key size. (See `ref_cdmi_encryption` in `ref_data_system_metadata` for the format.)	Optional
cdmi_value_hash_provided	JSON String	Contains the algorithm and length being used to hash the object value. (See `ref_cdmi_value_hash` in `ref_data_system_metadata` for the format.)	Optional
cdmi_latency_provided	JSON String	Contains the provided maximum time to first byte	Optional
cdmi_throughput_provided	JSON String	Contains the provided maximum data rate on retrieve	Optional
cdmi_sanitization_method_provided	JSON String	Contains the sanitization method used. (See `ref_cdmi_sanitization_method` in `ref_data_system_metadata` for the format.)	Optional
cdmi_RPO_provided	JSON String	Contains the provided duration, in seconds, between an update and when the update may be recovered	Optional
cdmi_RTO_provided	JSON String	Contains the provided duration, in seconds, to restore data	Optional
cdmi_authentication_methods_provided	JSON Array of JSON Strings	Contains a list of authentication methods enabled for the domain. (See `ref_cdmi_authentication_methods` in `ref_data_system_metadata` for the format.)	Optional

16.6 Metadata Update Operations

CDMI permits a client to replace all metadata items or to perform operations against one or more individual metadata items.

Replacing all metadata items is accomplished by including the metadata field in the update request body JSON and not specifying specific metadata items in the update URI.

Adding, updating, and removing specific metadata items is accomplished by specifying the specific metadata item names in the update URI:

- To add a new metadata item to an existing object, the metadata item name shall be included in the update request URI, and the metadata item shall be included in the metadata field in the update request body JSON.
- To update the value of an existing metadata item, the metadata item name shall be included in the update request URI, and the metadata item shall be included in the metadata field in the update request body JSON.
- To remove an existing metadata item, the metadata item name shall be included in the update request URI, and the metadata item shall not be included in the metadata field in the update request body JSON.

When individual metadata items are specified in the update URI, metadata items included in the metadata field in the request body JSON that are not referred to in the update URI shall be ignored.

Clause 17

Retention and Hold Management

17.1 Introduction

A cloud storage system may optionally implement retention management disciplines into the system management functionality of the cloud-based storage system. The implementation of retention and hold capabilities is indicated by the presence of the cloud storage system-wide capabilities for retention and hold capabilities.

Retention management includes implementing a retention policy, defining a hold policy to enable objects to be held for specific purposes (e.g., litigation), and defining how the rules for deleting objects are affected by placing either a retention policy and/or a hold on an object. CDMI™ object deletion is not a capability of retention management, per se, but rather is a general system capability. However, this clause describes what happens when placing either a retention policy and/or a hold on an object.

Retention management may be applied to the following object types:

- data objects,
- queue objects, and
- container objects.

17.2 Retention Management Disciplines

CDMI retention, deletion, and hold management affect any CDMI client that creates or deletes CDMI objects, as these disciplines mandate how a cloud storage system manages CDMI objects when they are created and until they are deleted.

CDMI retention management is comprised of three management disciplines: retention, hold, and deletion:

- CDMI retention uses retention time criteria to determine the time period during which object deletion from the CDMI-based system is prohibited. No changes to the object are allowed, even after the retention period has expired, except as specified below.
- CDMI hold prohibits object deletion and modification until all holds on the object have been released.
- A CDMI-based system shall not allow the deletion of a CDMI object before the CDMI retention time criteria are met or while holds exist. Any deletion attempts (e.g., by a CDMI application) shall return an error.
- After the CDMI retention time criteria have been met and all holds have been released, CDMI retention and holds shall no longer be a reason to prohibit object deletion.
- Once the retention period has started or if holds exist, changes to the object data and metadata shall not be allowed, with the exception of extensions to the retention and hold data system metadata. The retention data system metadata may be added or the retention period extended, and the hold data system metadata may be added or extended with additional holds. Any other attempt to modify the object shall return an error.

17.3 CDMI Retention

17.3.1 Overview

CDMI retention only allows one retention policy to be applied to an object at a time.

Retention management uses time criteria to determine the time period during which CDMI object deletion from the CDMI-based system shall be prohibited. CDMI retention criteria shall be specified by the following data system metadata:

- a retention criteria identifier—a CDMI client-specified string that shall identify the retention records class (cdmi_retention_id); and
- a retention start time and retention period time—the start time, when used together with period, indicating when retention shall no longer be enforced (cdmi_retention_period).

When a CDMI client attempts to delete an object, the cloud storage system shall evaluate all such retention criteria and return an error, if any retention criteria have not been met.

When copying objects with a retention policy, retention properties shall not be transferred from the source CDMI object to the destination object, and the destination object shall not have a retention policy.

Fig. 17.1 shows how to establish time-based retention with a retention identifier. The value of the object data system metadata for the retention period shall not be reduced.

Fig. 17.1: Object Retention

A specific HTTP error code (403) shall be returned on operations to objects that are under retention period when the cloud storage system attempts to change or delete the object before the retention period criteria are met.

A cloud storage system shall not prevent metadata changes that increase the retention period, as there are valid business reasons to change a retention period for an object.

17.3.2 Examples

1. Place an existing object under retention:

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

```

(continues on next page)

(continued from previous page)

```
{
    "metadata" : {
        "cdmi_retention_id" : "1",
        "cdmi_retention_period" : "2010-04-28T00:00:00.000000Z/2012-04-
↪27T00:00:00.000000Z"
    }
}
```

3239 The following shows the response.

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

3240 2. Increase the duration of retention on an existing object under retention:

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

{
    "metadata" : {
        "cdmi_retention_period" : "2011-04-28T00:00:00.000000Z/2013-04-
↪27T00:00:00.000000Z"
    }
}
```

3241 The following shows the response.

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

3242 3. Decrease the duration of retention on an existing object under retention:

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

{
    "metadata" : {
        "cdmi_retention_period" : "2011-04-28T00:00:00.000000Z/2012-01-
↪27T00:00:00.000000Z"
    }
}
```

3243 The following shows the response.

```
HTTP/1.1 403 Forbidden
```

17.4 CDMI Hold

17.4.1 Overview

CDMI hold enforces read-only data object access and prohibition of object deletion. A cloud storage system shall allow multiple holds to be applied to a single object to satisfy multiple hold orders. While an object is on hold, a cloud storage system shall strictly enforce read-only access to the object and prohibit object deletion.

When copying objects that are on hold, hold properties shall not be transferred from the source CDMI object to the destination object, and the destination object shall not be on hold.

Hold management uses a hold indicator to determine the time period(s) during which CDMI object revision (data and metadata) and deletion from the CDMI-based system shall be prohibited. CDMI hold criteria shall be specified by data system metadata, specifically, a hold criteria identifier that is a client-specified string that shall identify the holds and their order.

A CDMI client may place an object on hold by adding a hold identifier to the cdmi_hold_id data system metadata item. When an object is on hold, CDMI clients shall be subject to failures or unexpected state changes on operations, which would otherwise be successful if the object was not on hold.

Fig. 17.2 shows how placing a hold on an object affects its read-only and deletion capability.

Fig. 17.2: Object Hold

Fig. 17.3 shows how to establish time-based retention with a retention identifier that has a hold placed on the object. The value of the object data system metadata for the retention period shall not be reduced, and the value of the object data system metadata for hold identifiers shall not permit holds to be removed. Removing holds is outside the scope of the CDMI international standard.

Fig. 17.4 shows how placing multiple holds on an object affects its read-only and deletion capability.

A cloud storage system shall maintain an on-hold object in read-only mode with respect to the application access to data and metadata and shall prohibit deletion, either automated or explicit.

- CDMI clients shall tolerate these object on-hold failures or state changes.
- Releases from hold are not part of this international standard and are typically performed out of band using an additionally secured non-CDMI mechanism provided by the implementation.

A specific HTTP error code (403) shall be returned on operations to objects that are under a hold when the system attempts to change the object or attempts to delete the object before the hold is removed. This failure should be a an error to the application.

Fig. 17.3: Object Hold on Object with Retention

Fig. 17.4: Object with Multiple Holds

17.4.2 Examples

1. Place an existing object under hold:

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

{
  "metadata": {
    "cdmi_hold_id": {
       "case_7": ""
    }
  }
}
```

The following shows the response.

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

2. Attempt to remove a hold for an object under hold:

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

{
  "metadata": {
    "cdmi_hold_id": {}
  }
}
```

The following shows the response.

```
HTTP/1.1 403 Forbidden
```

3. Add a second hold to an object under hold:

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

{
  "metadata":{
    "cdmi_hold_id": {
       "case_7": "",
       "case_15": ""
   }
    }
}
```

The following shows the response.

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

17.5 CDMI Auto-deletion

17.5.1 Overview

CDMI deletion controls cloud storage system actions with respect to object deletion. A cloud storage system may automatically delete a CDMI object after the retention time and hold criteria have been met. (See `ref_cdmi_retention_autodelete` in `ref_data_system_metadata`.)

CDMI objects shall be automatically deleted by the system at the retention period expiration by setting the data system metadata flag cdmi_retention_autodelete. The cdmi_retention_autodelete flag indicates to the system that the object shall be made unavailable for access after the retention criteria have been satisfied. The system shall ensure that the object is no longer available through the CDMI interface. If the system has satisfied the retention requirement and a hold is established for the object, the object shall not be made unavailable or deleted. When a hold and retention have been applied to an object, both need to be satisfied (retention period expired and no holds existing) for objects to be automatically deleted from the system.

1. Place an object under retention with autodelete:

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

{
    "metadata":{
        "cdmi_retention_period": "2011-04-28T00:00:00.000000Z/2013-04-27T00:00:00.
↪000000Z",
        "cdmi_retention_autodelete": "true"
    }
}
```

   The following shows the response.

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

17.6 Retention Security Considerations

The accuracy and integrity of the retention start and elapsed times depend on the accuracy and integrity of the clock that is used to set their values. Equally important is the relative accuracy and security of the clock that determines if the retention period has elapsed when compared to the clock that sets the start time property. Relative time differences between these two clocks may lead to undesirable retention and deletion management behavior.

It is important to have a reliable source from which the system clock is set. A stratum 1 time is directly connected to a reference clock and is at the top of the time server hierarchy. Relative time differences between the system clock and the reference clock may lead to undesirable retention timestamps and difficulties with time action events.

1. An object is created in a cloud storage system at time 0 with a period of 8 years and autodelete of TRUE. At time 1 year, the system clock is adjusted forward to 9 years. Now, because the system time is 9 years, the retention time criterion is satisfied, even though only 1 year has actually elapsed. And, since autodelete is TRUE, the system automatically deletes the object.

The specification for accuracy and integrity of timekeeping is not within the scope of CDMI. However, to prevent undesirable retention and deletion management consequences, systems should maintain accurate clock time, with zero or minimal deviation to clock integrity.

Clause 18

Scope Specification

18.1 Introduction

CDMI™ provides a standardized mechanism to define sets of objects that match certain characteristics. This mechanism is known as a CDMI scope specification. Scope specifications are typically used to provide a CDMI client with a way to indicate in what set of CDMI objects it is interested.

Each JSON object within the scope specification represents a set of conditions that shall all be true in order for an object to be considered to match against the scope (a logical AND relationship). For queries, a matching object would be returned in the query results. An empty scope specification is considered to evaluate to true. Multiple JSON objects are used to express logical OR relationships, where if any JSON object in the scope evaluates to true, then the object shall be considered to have matched against the scope.

Each JSON object is constructed using the same structure that CDMI objects use. To show this structure, assume the following result from a CDMI GET for a data object:

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

{
    "objectType" : "application/cdmi-object",
    "objectID" : "00007E7F0010EB9092B29F6CD6AD6824",
    "objectName" : "MyDataObject.txt",
    "parentURI" : "/MyContainer/",
     "parentID" : "00007E7F00102E230ED82694DAA975D2",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/dataobject/",
    "completionStatus" : "Complete",
    "mimetype" : "text/plain",
    "metadata" : {
        "cdmi_size" : "108263",
        ...
    },
    "valuerange" : "0-108262",
    "value" : "..."
}
```

18.2 Examples

Each field inside a scope specification JSON object represents a condition that shall be met for a field.

1. A query to find all objects belonging to the domain /cdmi_domains/MyDomain/ is structured as follows:

```
[
    {
        "domainURI" : "== /cdmi_domains/MyDomain/"
    }
]
```

2. To query for all objects belonging to the domain /cdmi_domains/MyDomain/ AND are also located within the container MyContainer, the scope specification is structured as follows:

```
[
    {
        "parentURI" : "== /MyContainer/",
        "domainURI" : "== /cdmi_domains/MyDomain/"
    }
]
```

3. To query for all objects created within a certain time range, the scope specification is structured as follows:

```
[
    {
        "metadata": {
            "cdmi_ctime": [
                ">=2012-01-01T00:00:00",
                "<=2013-01-01T00:00:00"
            ]
        }
    }
]
```

When multiple matching expressions are specified for a given field or metadata item, all matching expression must evaluate true for an object to be considered a query result.

1. To query for all objects that belong to the domain MyDomain OR are located within the container MyContainer, the query is structured as follows:

```
[
    {
        "parentURI" : "== /MyContainer/",
    },
    {
        "domainURI" : "== /cdmi_domains/MyDomain/"
    }
]
```

Queries may match on any field within an object that a cloud storage system is capable of returning as a result of an object GET.

1. To query metadata items, the metadata object is included as an object within the query request. This query is shown as follows:

```
[
    {
```

(continues on next page)

(continued from previous page)

```
        "metadata" : {
            "colour" : "== blue"
        }
    }
]
```

This approach allows matching against arbitrarily nested metadata structures. When a JSON object is included in the scope specification, matches are performed within that object, and when a JSON array is included in the scope specification, matches are performed within that array. Matching against the contents of arrays of objects is indicated by having an object within the array, as illustrated in Example 5.

1. To query all objects with an ACE associated with the user “jdoe”:

```
[
    {
        "metadata" : {
            "cdmi_acl" : [
                {
                    "identifier" : "== jdoe"
                }
            ]
        }
    }
]
```

To query the value of objects, the value field is included within the query request. Values are always represented using base 64 encoding in queries.

1. This query is shown as follows:

```
{
   [
       {
           "value": "== Ymx1ZQ=="
       }
   ]
}
```

Query against the value of objects is optional and is indicated by the presence of the cdmi_query_value capability.

18.3 Query Matching Expressions

Table 18.1 defines the query matching expressions.

Table 18.1: .*

Matching Expression	Description
“field” : “*”	The exists matching expression tests for the existence of the field. If the field is present, even if empty, the condition shall be considered to be met.
“field” : “!*”	The not exists matching expression tests for the non-existence of the field. If the field is absent, the condition shall be considered to be met.
“field” : “== constant”	The equals matching expression tests for the equality of the value of the field and a specified constant value. The equality test is case sensitive. The leading space after the “==” and before the constant value is not included in the comparison. If the constant value matches the value of the field, the condition shall be considered to be met.
“field” : “#== constant”	The numeric equals matching expression tests for the numeric equality of the value of the field and a specified constant value. Numeric constant strings shall be processed according to the JSON number representation described in RFC 4627. A numeric matching expression shall be considered to be non-matching against a non-numeric field value.
“field” : “!= constant”	The not equals matching expression tests for the non-equality of the value of the field and a specified constant value. The not-equals test is case sensitive. The leading space character after the “!=” and before the constant value is not included in the comparison. If the constant value does not match the value of the field, the condition shall be considered to be met. If the matching expression starts with a “#” character (e.g., “#!=”), the value of the field is considered to be numeric for the purposes of comparison. Numeric constant strings shall be processed according to the JSON number representation described in RFC 4627. A numeric matching expression shall be considered to be non-matching against a non-numeric field value.
“field” : “> constant”	The greater than matching expression tests if the value of the field is lexicographically greater than a specified constant value. The greater than test is case sensitive. The leading space character after the “>” and before the constant value is not included in the comparison. If the constant value is greater than the value of the field, the condition shall be considered to be met.
“field” : “#> constant”	The numeric greater than matching expression tests if the numeric value of the field is greater than a specified constant value. Numeric constant strings shall be processed according to the JSON number representation described in RFC 4627. A numeric matching expression shall be considered to be non-matching against a non-numeric field value.
“field” : “>= constant”	The greater than or equals to matching expression tests if the value of the field is lexicographically greater than or equal to a specified constant value. The greater than or equals to test is case sensitive. The leading space character after the “>=” and before the constant value is not included in the comparison. If the constant value is greater than or equal to the value of the field, the condition shall be considered to be met.
“field” : “#>= constant”	The numeric greater than or equals to matching expression tests if the numeric value of the field is greater than or equal to a specified constant value. Numeric constant strings shall be processed according to the JSON number representation described in RFC 4627. A numeric matching expression shall be considered to be non-matching against a non-numeric field value.
“field” : “< constant”	The less than operator tests if the value of the field is lexicographically less than a specified constant value. The less than test is case sensitive. The leading space character after the “<” and before the constant value is not included in the comparison. If the constant value is less than the value of the field, the condition shall be considered to be met.
“field” : “#< constant”	The numeric less than operator tests if the numeric value of the field is less than a specified constant value. Numeric constant strings shall be processed according to the JSON number representation described in RFC 4627. A numeric matching expression shall be considered to be non-matching against a non-numeric field value.
“field”	The less than or equals to matching expression tests if the value of the field is lexicographically less than or

All fields in objects that are not included in the scope specification shall be ignored for the purpose of matching objects.

When a URI is used as the constant for the equals and not equals operators against the parentURI, domainURI, and capabilitiesURI, either a URI by path or URI by object ID can be specified and are considered interchangeable.

1. In a query to find all objects belonging to a specific domain, the following two query scopes are considered identical:

```
[
    {
        "domainURI" : "== /cdmi_domains/MyDomain/"
    }
]
```

and

```
[
    {
        "domainURI" : "== /cdmi_objectid/00007E7F001074C86AD256DA5C67180D/"
    }
]
```

2. Likewise, a query to find all objects with a given parent container would have two equivalent forms:

```
[
    {
        "parentURI" : "== /MyContainer/"
    }
]
```

and

```
[
    {
        "parentURI" : "== /cdmi_objectid/00007ED900100E358C3B312DB652C201/"
    }
]
```

If an object ID is used in a query scope in the objectID field or the parentID field, all object IDs shall be processed such that they are case insensitive.

Clause 19

Results Specification

19.1 Introduction

CDMI™ provides a standardized mechanism to define subsets of object contents. This mechanism is known as a CDMI results specification. Results specifications are typically used to provide a CDMI client with a way to indicate on what subset of the contents of CDMI objects it intends to retrieve or operate.

Each JSON object within the results specification represents a set of fields that are returned for each matching object.

The results JSON object shall be constructed using the same structure as is used for CDMI objects. To show this, assume the following result from a CDMI GET for a data object:

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

{
    "objectType" : "application/cdmi-object",
    "objectID" : "00007E7F0010EB9092B29F6CD6AD6824",
    "objectName" : "MyDataObject.txt",
    "parentURI" : "/MyContainer/",
     "parentID" : "00007E7F00102E230ED82694DAA975D2",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/dataobject/",
    "completionStatus" : "Complete",
    "mimetype" : "text/plain",
    "metadata" : {
        "cdmi_size" : "108263",
            ...
    },
    "valuerange" : "0-108262",
    "value" : "..."
}
```

19.2 Examples

Each field inside a results specification JSON object indicates that the field shall be included in the results.

1. The following results specification requests that the objectID and cdmi_size metadata fields be returned in the results:

```
{
    "cdmi_results_specification" : {
        "objectID" : "",
        "metadata" : {
            "cdmi_size" : ""
        }
    }
}
```

2. If an object is matched, the result JSON is enqueued as follows:

```
{
    "objectID" : "00007E7F0010EB9092B29F6CD6AD6824",
    "metadata" : {
        "cdmi_size" : "108263"
    }
}
```

For most common use cases, clients request either the objectID, the objectName and parentURI, or all three fields in the cdmi_results_specification. If the parentURI or objectName is requested, the field shall only be returned for objects existing in a container object.

1. To request all metadata items be returned for each matching object, the following cdmi_results_specification shall be used:

```
{
    "cdmi_results_specification" : {
        "metadata" : ""
    }
}
```

2. To request all fields and all metadata items be returned for each matching object, the following cdmi_results_specification shall be used:

```
{
    "cdmi_results_specification" : ""
}
```

The value field is always returned in base 64 encoding when included in a query result, where the valuetransferencoding field indicates the encoding that should be expected if a GET to read the object is performed.

Clause 20

Logging

20.1 Overview

CDMI™ logging is divided into functional areas, each with differing levels of detail. These areas are:

- object logging,
- security logging, and
- data management logging.

This international standard does not define the format of log messages. It is anticipated that future logging standards will address this area.

A CDMI client may access log data by creating a logging queue that indicates the scope of log messages that the client wishes to receive, as described in `ref_logging_queues`. If the user has sufficient permissions to create a logging queue, all log messages to which he or she has subscribed shall be enqueued into the queue, which may be accessed for processing and archival storage.

If multiple logging queues are defined, each logging queue shall get the log entry for a subscribed event. If no logging queues are defined that subscribe to a given log message or class of log messages, these messages do not have to be retained by the cloud storage system.

20.2 Object Logging

If the cloud storage system supports logging, then all operations performed on CDMI objects (data objects, container objects, domain objects, queue objects, and capability objects) shall be persistently stored into all defined logging queues.

Log messages shall contain a minimum of the following information, in a format specified by the implementor:

- a timestamp in ISO-8601 format (see `ref_time_representations`);
- the domain in which the operation was performed;
- the operation being performed;
- the URI of the object against which the operation was performed;
- the principal of the entity by which the operation was performed; and
- the result of the operation.

Operations logged should include operations performed to a CDMI-exported file system.

20.3 Security Logging

All security-sensitive events, including establishing sessions, authenticating and authorizing users, and modifying and delegating domains, shall be logged as security events. Security logging includes managing credentials (i.e., validating revocation lists) and managing users and domains. Security logging should also include out-of-band operations that affect the security of a cloud storage system (e.g., modifying security properties of a CDMI domain via an administrative GUI).

If the cloud storage system supports a queue type of cdmi_logging_queue and a cdmi_logging_class of cdmi_security_logging as shown in `ref_logging_queues`, this metadata indicates that the system supports audit logging. Consequently, the system-wide capability of cdmi_security_audit specified in `ref_systemwide_capabilities` of Section 12.1.3 shall be set to “true”. Otherwise, cdmi_security_audit shall not be present.

20.4 Data Management Logging

In addition to log messages associated with changing metadata when changing data system metadata, logging should also include all conditions where the specified or actual data system metadata for objects change. For example, if the number of requested replicas was changed by a client, this change shall generate a log message indicating this change. A corresponding change in the actual number of replicas by the system shall also generate a log message.

This class of logging shall also contain object holds and retention policy log messages.

20.5 Logging Queues

Logging queues allow CDMI clients to get detailed logging information about the actions related to the operation of a cloud storage system. As queue data is persistent, no session state needs to be retained by the client. If different logging queues are used for different clients, then each client operates independently from the others (e.g., an analysis application may retrieve information about actions performed in a specific domain or set of objects using a logging queue that is uniquely configured to its specific needs).

Logging queues differ from notification queues (see `ref_notification_queues`) in that the information provided is at a much more detailed level than notifications and is typically restricted to a smaller, privileged subset of clients.

When a client wishes to receive logging information, it may first check if the system is capable of providing logging by checking for the presence of the cdmi_logging capability in the root container capabilities. If this capability is not present, creating a logging queue shall be successful, but no logging entries shall be enqueued into the logging queue.

When creating a logging queue, the metadata described in Table 20.1 shall be provided. Attempts to change metadata in this table shall result in an HTTP status code of 403 Forbidden . Once a logging queue has been created, with the exception of cdmi_queue_type, the metadata items in this table cannot be changed. cdmi_queue_type can only be removed, indicating to the system that the logging queue shall no longer receive log messages and shall be treated as a regular CDMI queue object.

Table 20.1: .*

Metadata Name	Type	Description	Requirement
cdmi_queue_type	JSON String	The queue type indicates how the cloud storage system shall manage the queue object. The type of cdmi_logging_queue is defined for logging queues.	Mandatory
cdmi_logging_class	JSON Array of JSON Strings	Contains a JSON array that indicates which log messages are to be enqueued. Defined values are: • cdmi_object_logging - Receive logging messages related to object operations; • cdmi_datasystem_logging - Receive logging messages related to data system metadata state changes; and • cdmi_security_logging - Receive logging messages related to security events. Clients may include the desired classes of log messages in the cdmi_logging_class JSON array. If all log messages are desired, an empty JSON array shall be used.	Mandatory
cdmi_scope_specification	JSON Array of JSON Objects	The scope specification determines the set of objects for which associated log messages shall be enqueued. If logging is desired for all objects, include an empty JSON array. For security logging, the scope specification is ignored. See `scope_specification` for how to construct a scope specification.	Mandatory

1. An example of the metadata associated with a logging queue is as follows:

```
{
    "metadata" : {
        "cdmi_queue_type" : "cdmi_logging_queue",
        "cdmi_logging_class" : [
            "cdmi_object_logging",
            "cdmi_security_logging"
        ],
        "cdmi_scope_specification" : [
            {
                "domainURI" : "== /cdmi_domains/MyDomain/"
            }
        ]
    }
}
```

When logging messages are dequeued from a logging queue, the contents of each queue value shall contain a JSON object and have a mimetype field value of “application/json”. This JSON object contains one or more JSON strings or objects, each representing a single log message.

Log messages are only included in a logging queue if the user who created the logging queue is able to access the object associated with the log message, (i.e., user has any ACE from `ref_ace_mask_bits`).

1. If the administrator created the logging queue, then all matching objects, without restriction, are included in the results. If user “jdoe” created the logging queue, then only logging messages for objects that “jdoe” is allowed to access are included in the results.

`ref_logging_status_metadata` describes the system-created metadata that provides details on the status of the logging queue.

Table 20.2: .*

Metadata Name	Type	Description	Requirement
cdmi_logging_status	JSON String	A string indicating the state of the logging queue. Defined values are: • Processing - Indicates that the logging queue is scanning for results; • Halted - Indicates that new log messages will no longer be enqueued; • Current - Indicates that the logging queue contained all log messages that can be found at this time; and • Error - Indicates that the logging queue metadata is not valid, or other errors were encountered that prevented logging messages from being enqueued. Arbitrary vendor-defined text may follow the string “Error”.	Mandatory

20.6 Logging Security Considerations

The timestamp accuracy and integrity of the log entries depend on the accuracy and integrity of the clock that is used to set their timestamp values. Accurate timestamps are essential to troubleshooting, forensic analysis of distributed attacks, dispute resolution, and proof of time-sensitive transactions. In essence, debugging, security, audit, and authentication are founded on the basis of event correlation (i.e., what happened when and whether the action occurred on the client or server side), and these security considerations depend on good time synchronization.

While specifying the accuracy and integrity of timekeeping is not within the scope of this international standard, to demonstrate that log timestamps are trustworthy, timestamps should be traceable to a standard time, and it should be demonstrated that system time may not be arbitrarily changed.

&nbsp;

Clause 21

Notification Queues

A cloud storage system may optionally implement notification functionality. The implementation of notification is indicated by the presence of the cloud storage system-wide capabilities for notification and requires support for CDMI™ queues.

Notification queues allow CDMI clients to efficiently discover what changes have occurred to the system. As queue data is persistent, no session state needs to be retained by the client. If different notification queues are used for different clients, then each client operates independently from the others (e.g., a storage management application may use a notification queue to keep its database current without having to do full scans of a container to discover what data objects have been added, modified, or removed).

When a client wishes to receive notifications, it may first check if the system is capable of providing notifications by checking for the presence of the cdmi_notification capability in the root container capabilities. If this capability is not present, creating a notification queue shall be successful, but no notifications shall be enqueued into the notification queue.

To create a notification queue, the client creates a regular CDMI queue and adds metadata instructing the storage system to treat the queue as a notification queue. This added metadata also instructs the system about what types of notifications shall be generated and what information shall be included with each notification.

After the notification queue is created, all subsequent matching events after the queue creation time shall result in notification results being enqueued into the queue. CDMI does not mandate any specific ordering of events, and clients must be able to handle events that arrive out of order.

When creating a notification queue, the metadata described in Table 21.1 shall be provided. Attempts to change metadata in this table shall result in an HTTP status code of `403 Forbidden`. After a notification queue has been created, with the exception of cdmi_queue_type, the metadata items in this table cannot be changed. cdmi_queue_type can only be removed, indicating to the system that the notification queue shall no longer receive notifications and shall be treated as a regular CDMI queue object.

3493

Table 21.1: .*

3494

Metadata Name	Type	Description	Requirement
cdmi_queue_type	JSON String	The queue type indicates how the cloud storage system shall manage the queue object. The type of cdmi_notification_queue is defined for notification queues.	Mandatory
cdmi_notification_events	JSON Array of JSON Strings	The notification events metadata contains a JSON array that indicates which events generate notifications. Defined values are: • cdmi_create_processing - Notifications are generated when a new object is created but is still in the "Processing" completion status. • cdmi_create_complete - Notifications are generated when a new object is created immediately or when a new object in the process of being created transitions from the "Processing" completion status. When an object transitions from "Processing" completion status, the "cdmi_event_result" is the HTTP result code that would have been returned if the create operation was not delayed. • cdmi_read - Notifications are generated when an object is read. • cdmi_modify_processing - Notifications are generated when an existing object is modified but is still in the "Processing" completion status.	Mandatory

1. The metadata associated with a notification queue is as follows:

```
{
    "metadata" : {
        "cdmi_queue_type" : "cdmi_notification_queue",
        "cdmi_notification_events" : [
            "cdmi_create_complete",
            "cdmi_read",
            "cdmi_modify_complete",
            "cdmi_delete"
        ],
        "cdmi_scope_specification" : [
            {
                "domainURI" : "== /cdmi_domains/MyDomain/",
                "parentURI" : "starts /sandbox",
                "metadata" : {
                    "cdmi_size" : ">+100000"
                }
            }
        ],
        "cdmi_results_specification" : {
            "cdmi_event" : "",
            "cdmi_event_result" : "",
            "cdmi_event_time" : "",
            "objectID" : "",
            "metadata" : {
                "cdmi_size" : ""
            }
        }
    }
}
```

   When notification results are stored in a notification queue, each enqueued value shall consist of a JSON object of MIME type “application/json”. This JSON object contains the specified values requested in the cdmi_results_specification of the notification queue metadata.

2. A notification result JSON object is as follows:

```
{
    "cdmi_event" : "cdmi_read",
    "cdmi_event_result" : "200 OK",
    "cdmi_event_time" : "2010-11-15T13:12:52.342324Z",
    "objectID" : "00007E7F0010EB9092B29F6CD6AD6824",
    "metadata" : {
        "cdmi_size" : "108263"
    }
}
```

Objects shall only be included in the notification results if the user who created the notification queue is able to read the matching object.

If the administrator created the notification queue, then all matching objects that the administrator is allowed to read are included in the results. If user “jdoe” created the notification queue, then only matching objects that “jdoe” is allowed to read are included in the results.

Table 21.2 describes the system-created metadata that provides details on the status of the notification queue.

Table 21.2: Notification Status Metadata

Metadata Name	Type	Description	Requirement
cdmi_notification_status	JSON String	A string indicating the state of the notification queue. Defined values are: • Processing - Indicates that the notification queue is scanning for results; • Halted - Indicates that new notifications will no longer be enqueued; • Current - Indicates that the notification queue contained all notifications that can be found at this time; and • Error - Indicates that the notification queue metadata is not valid, or other errors were encountered that prevented notification messages from being enqueued. Arbitrary vendor-defined text may follow the string "Error". If this metadata item does not exist, then notifications have not yet started being enqueued.	Mandatory

Clause 22

Query Queues

22.1 Overview

A cloud storage system may optionally implement metadata and/or full-text query functionality. The implementation of query is indicated by the presence of the cloud storage system-wide capabilities for query and requires support for CDMI™ queues.

Query queues allow CDMI clients to efficiently discover what content matches a given set of metadata query criteria or full-content search criteria. Clients create or update a query queue by specifying metadata that defines the matching criteria (known as the query scope), along with what results should be returned for matching objects (known as the query results). The cloud service shall then perform the query using the content existing at the time the query is being processed, storing the query results in the query queue. As query results are found, they are added to the queue, and when the query is complete, the cdmi_query_status metadata of the queue is changed to indicate that the query has completed. Any matching objects created or modified while the query is being performed may or may not be included in the query results (e.g., as a consequence of eventual consistency).

When a client wishes to perform queries, it may first check if the system is capable of providing query functionality by checking for the presence of the cdmi_query capability in the root container capabilities. If this capability is not present, creating a query queue shall be successful, but no query results shall be enqueued into the query queue.

When creating a query queue, the metadata described in Table 22.1 shall be provided. Attempts to change metadata in this table shall result in an HTTP status code of `403 Forbidden`. After a query queue has been created, with the exception of cdmi_queue_type, the metadata items in this table cannot be changed. If the value of cdmi_queue_type is changed from "cdmi_query_queue", this change indicates to the system that an in-process query shall be stopped, the query queue shall no longer receive query results, and the query queue shall be treated as a regular CDMI queue object. To start a new query with an existing queue, the value of the cdmi_queue_type shall be changed back to "cdmi_query_queue". This international standard does not define a mechanism to pause a running query or resume a stopped query.

Table 22.1: .*

Metadata Name	Type	Description	Requirement
cdmi_queue_type	JSON String	The queue type indicates how the cloud storage system shall manage the queue object. The type of cdmi_query_queue is defined for query queues.	Mandatory
cdmi_scope_specification	JSON Array of JSON Objects	The scope specification determines which objects are included in the query results. This scope specification is similar to a "WHERE" clause in SQL-like languages. To query all objects, specify an empty JSON array. See `scope_specification` for how to construct a scope specification.	Mandatory
cdmi_results_specification	JSON Object	The results specification contains the JSON fields to be returned for each object that matches the query. This results specification is similar to a "SELECT" clause in SQL-like languages. See `ref_results_specification` for how to construct a results specification.	Mandatory

1. An example of the metadata associated with a query queue is as follows:

```
{
    "metadata" : {
        "cdmi_queue_type" : "cdmi_query_queue",
        "cdmi_scope_specification" : [
            {
                "domainURI" : "== /cdmi_domains/MyDomain/",
                "parentURI" : "starts /sandbox",
                "metadata" : {
                    "cdmi_size" : "#> 100000"
                }
            }
        ],
        "cdmi_results_specification" : {
            "objectID" : "",
            "metadata" : {
                "cdmi_size" : ""
            }
        }
    }
}
```

When results are stored in a query queue, each enqueued value shall consist of a JSON object of MIME type "application/json". This JSON object contains the specified values requested in the cdmi_results_specification of the query queue metadata.

1. An example of a query result JSON object is as follows:

```
{
    "objectID" : "00007E7F0010EB9092B29F6CD6AD6824",
    "metadata" : {
        "cdmi_size" : "108263"
    }
}
```

Table 22.2 describes the system-created metadata that provides details on the status of the query queue.

Table 22.2: Query Status Metadata

Metadata Name	Type	Description	Requirement
cdmi_query_status	JSON String	When present, this metadata item indicates the state of the query queue. Defined values are: * Processing - Indicates that the query queue is scanning for results; * Halted - Indicates that new query results will no longer be enqueued; * Current - Indicates that the query queue contained all query results that can be found at this time; and * Error - Indicates that the query queue metadata was not valid, or other errors were encountered that prevented all query results from being enqueued. Arbitrary vendor-defined text may follow the string "Error".	Mandatory

Objects shall only be included in the query results if the user who created the query queue is able to read the matching objects or metadata.

1. If the administrator created the query queue, then all matching objects that the administrator is allowed to read are included in the results. If user "jdoe" created the query queue, then only matching objects that "jdoe" is allowed to read are included in the results.

22.2 Extending CDMI Query

An implementor of a CDMI server may extend CDMI query by adding vendor-specific matching expressions. When an implementor adds vendor-specific metadata fields, these fields shall be queried using the standard query queue functionality.

An implementor of a CDMI server may extend CDMI query by allowing the creation of vendor-specific query queues with a type other than cdmi_query_queue.

Section V

CDMI Annexes

Clause 23

(Informative) Extensions

23.1 Overview

CDMI extensions describe additional functionality for extending the CDMI International Standard. Each extension is first written as a standalone document that describes the changes that are required to implement the functionality being added into this International Standard.

When one or more vendors have implemented a CDMI extension, it is eligible to be added to this annex. When multiple vendors have implemented a CDMI extension and demonstrated interoperability, the extension is eligible to be merged into the CDMI International Standard itself.

CDMI extensions shall not break or modify existing functionality, and thus do not result in compatibility problems with existing clients. Compatibility is typically accomplished by relaxing restrictions imposed in the current CDMI International Standard, adding new fields, or using reserved names for metadata. Theclients that are using CDMI capabilities can identify the functionality that is associated with these CDMI extensions.

23.2 Summary Metadata for Bandwidth

23.2.1 Overview

Domain summaries provide summary measurement information about domain usage and billing. Some systems may track additional usage and billing information related to network bandwidth. This extension proposes a set of additional, optional contents for domain summary objects.

23.2.2 Changes to CDMI 1.1

The changes proposed are a set of additional, optional contents for domain summary objects.

1. Insert into Clause 3.

   private network segment a single IP address or range of IP addresses that are considered internal (e.g., LAN)

   public network segment a single IP address or range of IP addresses that are considered external (e.g., WAN)

2. Add table entries to the end of Table 10.2 in `ref_domain_object_summaries` as follows:

3580

Metadata Name	Type	Description	Requirement
cdmi_summary_network_bytes	JSON String	Total number of bytes read/written to/from public/private network segments	Optional
cdmi_summary_reads_private	JSON String	Total number of bytes read from private network segment	Optional
cdmi_summary_reads_private_min	JSON String	Minimum number of bytes read from private network segment for the given interval	Optional
cdmi_summary_reads_private_max	JSON String	Maximum number of bytes read from private network segment for the given interval	Optional
cdmi_summary_reads_private_avg	JSON String	Average number of bytes read from private network segment for the given interval	Optional
cdmi_summary_writes_private	JSON String	Total number of bytes written to private network segment	Optional
cdmi_summary_writes_private_min	JSON String	Minimum number of bytes written to private network segment for the given interval	Optional
cdmi_summary_writes_private_max	JSON String	Maximum number of bytes written to private network segment for the given interval	Optional
cdmi_summary_writes_private_avg	JSON String	Average number of bytes written to private network segment for the given interval	Optional
cdmi_summary_reads_public	JSON String	Total number of bytes read from public network segment	Optional
cdmi_summary_reads_public_min	JSON String	Minimum number of bytes read from public network segment for the given interval	Optional
cdmi_summary_reads_public_max	JSON String	Maximum number of bytes read from public network segment for the given interval	Optional
cdmi_summary_reads_public_avg	JSON String	Average number of bytes read from public network segment for the given interval	Optional
cdmi_summary_writes_public	JSON String	Total number of bytes written to public network segment	Optional
cdmi_summary_writes_public_min	JSON String	Minimum number of bytes written to public network segment for the given interval	Optional
cdmi_summary_writes_public_max	JSON String	Maximum number of bytes written to public network segment for the given interval	Optional
cdmi_summary_writes_public_avg	JSON String	Average number of bytes written to public network segment for the given interval	Optional
cdmi_summary_reads_total	JSON String	Total number of bytes read from both public and private network segments	Optional
cdmi_summary_writes_total	JSON String	Total number of bytes written to both public and private network segments	Optional

23.3 Expiring Access Control Entries (ACEs)

23.3.1 Overview

A common trait of cloud storage services is the ability to share an object with other clients for a limited time. This extension adds an attribute of ACEs used in ACLs that imposes a time limit (expiration) on the ACE. Once the ACE expires, the ACE is no longer valid or included in the authorization calculation for the object.

23.3.2 Changes to CDMI 1.1

1. Insert into `ref_acl_evaluation`:

   After the bullet item:

   - ACEs that do not refer to the principal P requesting the operation are ignored.

   Insert bullet:</P>

   - ACEs that have an expiration value less than the current time are ignored.

2. Change `ref_acl_evaluation`:

   Original text:

   ACE = { acetype , identifier , aceflags , acemask , acetime }

   Revised text:

   ACE = { acetype , identifier , aceflags , acemask , acetime, expiration }

3. Insert into `ref_acl_evaluation` after "`acemask = uint_t | acemaskstring`":

   expiration = uint_t

4. Insert into `ref_acl_evaluation` after "When ACE masks…":

   When ACE expiration is presented in string format, it shall be specified in ISO-8601 point-in-time format as described in `ref_time_representations`.

5. Insert a new subclause 16.1.x - ACE Expiration.

   An ACE may have an optional expiration associated with it. The expiration is a point-in-time value, in ISO-8601 point-in-time format, as described in `ref_time_representations`, which specifies that the ACE is no longer valid and shall be ignored after the time specified.

23.4 Group Storage System Metadata

23.4.1 Overview

ACLs in CDMI can refer to the owner of an object by specifying an ACE Who of “OWNER@”. This reference corresponds to the contents of the cdmi_owner storage system metadata. However, no cdmi_group storage system metadata corresponds to an ACE Who of “GROUP@”.

This extension defines a new storage system metadata item, cdmi_group, that allows an object to be associated with a group for ACL evaluation purposes.

23.4.2 Changes to CDMI 1.1

1. Add a table entry to the end of Table 12.3 in Section 12.1.3.

Capability Name	Type	Definition
cdmi_group	JSON String	If present and “true”, this capability indicates that the cloud storage system supports group storage system metadata to indicate a group associated with the object.

2. Add a table entry below “cdmi_owner” in Table 16.6 of Section 16.3.

Metadata Name	Type	Description	Requirement
cdmi_group	JSON String	The name of the group that is associated with the object.	Optional

Section VI

References

References

Todo: find a better way to include these references.

Bibliography

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

[OCCI] “Open Cloud Computing Interface”, Version 1.1, June 2011. Specification - http://occi-wg.org/about/specification/

[PKS12] RSA Laboratories, PKCS #12: Personal Information Exchange Syntax, Version 1.0, June 1999. Specification and Technical Corrigendum - http://www.rsa.com/rsalabs/node.asp?id=2138

[REST] “Representational State Transfer” - http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

Index