Cloud Data Management Interface (CDMI)

Version 2.0.0a

SNIA Working Draft

Dec 19, 2018

USAGE

DISCLAIMER

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

Table of Contents:

List of Figures

List of Tables

Section I

CDMI Preamble

345 Introduction

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

348 This document is organized as follows:

349

350 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 12	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 18	Retention and Hold Management	Describes the optional retention management disciplines to be implemented into the system management functions
Clause 19	Scope Specification	Describes the structure of the scope specification for JSON objects
Clause 20	Results Specification	Provides a standardized mechanism to define subsets of CDMI object contents
`Clause %s`	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
Clause 23	Encrypted Objects	Describes how to work with transparently encrypted objects
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**

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

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

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

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

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

394 **RFC 6068** The 'mailto' URI Scheme - see **RFC 6068**

395 **FC 5652** Cryptographic Message Syntac (CMS) - see **RFC 5652**

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

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

398 **FC 7515** JSON Web Signatures - see **RFC 7515**

399 **FC 7516** JSON Web Encryption - see **RFC 7516**

400 **FC 7518** JSON Web Algorithms - see **RFC 7518**

401 **SNIA TLS** TLS Specification for Storage Systems, version 1.0 - https://snia.org/tech_activities/standards/curr_
402 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

current data object version the most recent version of a version-enabled data object

data object version either the current data object version or an historical data object version

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

delegated access control (DAC) the process of delegating an access control decision to a third party

delegated access control provider (DAC provider) a third-party system that is capable of making access control decisions

delegated access control request (DAC request) a request made to a DAC provider for an access control decision

delegated access control response (DAC response) a response from a DAC provider indicating the result of a request for an access control decision

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

historical data object version a non-current state of a version-enabled data object

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.

intermediary CDMI server a CDMI server that is capable of forwarding DAC requests and responses

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

JOSE JavaScript Object Signing and Encryption

JWA JSON Web Algorithm

JWE JSON Web Encryption

JWS JSON Web Signing

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**)

version-enabled data object a CDMI data object with versioning enabled

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 2.

Table 2: 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
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

{
  "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 3 shall be interpreted as described in **RFC 2119**.

Table 3: 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. 1.

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

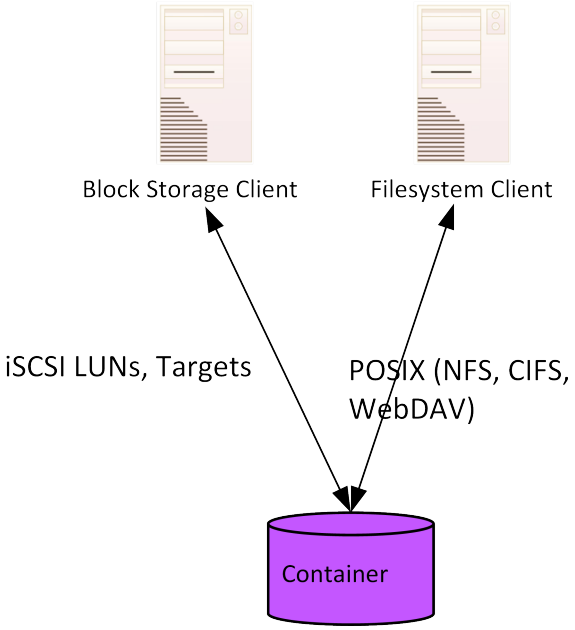


Fig. 1: Existing Data Storage Interface Standards

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. 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).

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 16.4.

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 16.1.

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,

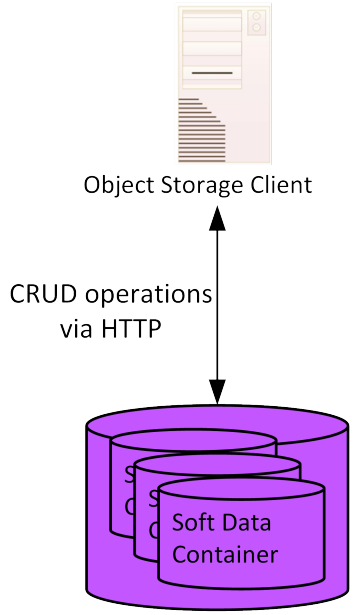


Fig. 2: Storage Interfaces for Object Storage Client Data

if the cloud service supports this level of granularity. For more information on support for data system metadata, see Section 16.2.

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. 3.

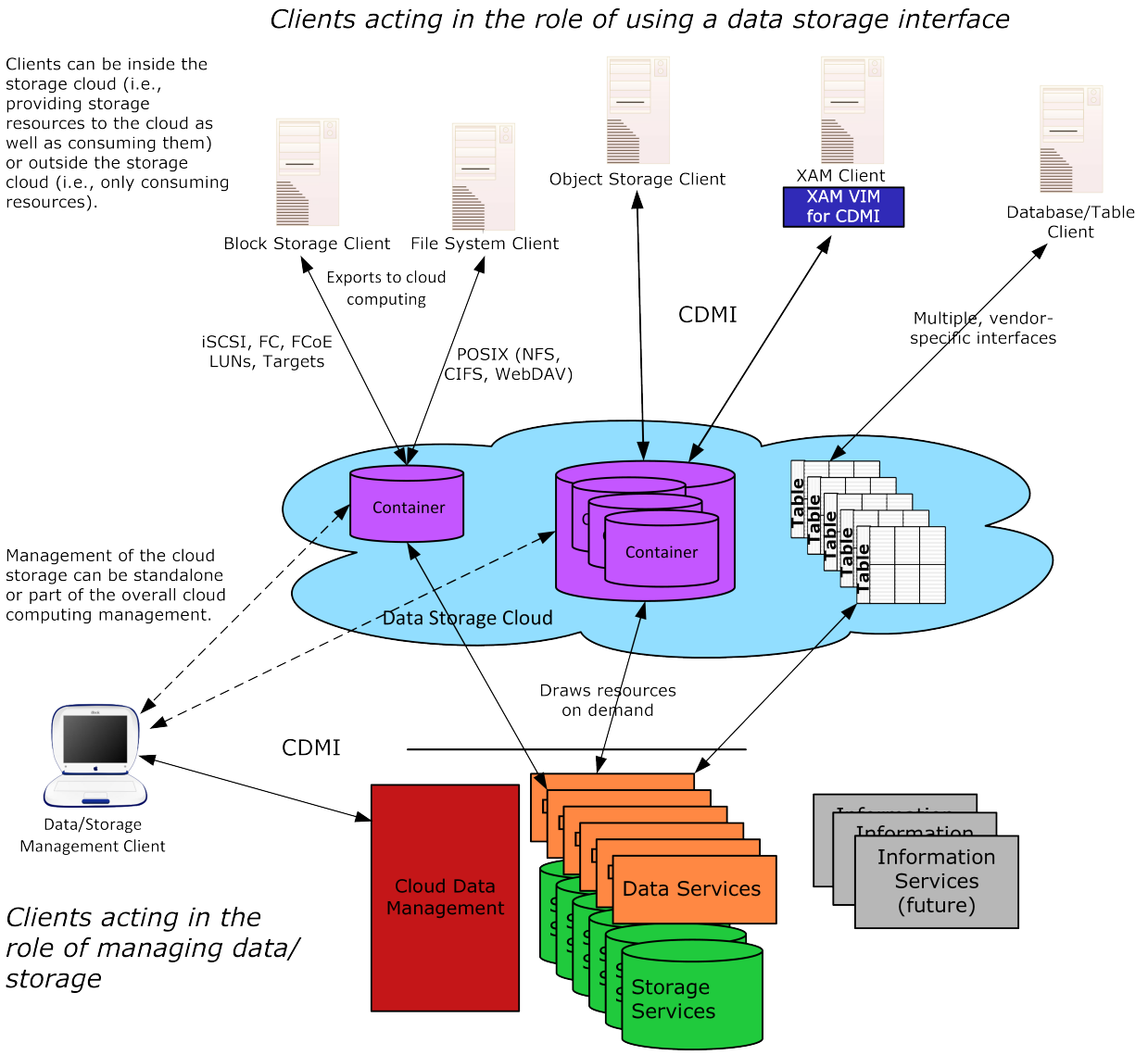


Fig. 3: Cloud Storage Reference Model

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. 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. 4.

The five types of resources defined are shown in Table 4. The content type in any given operation is specific to each type of resource.

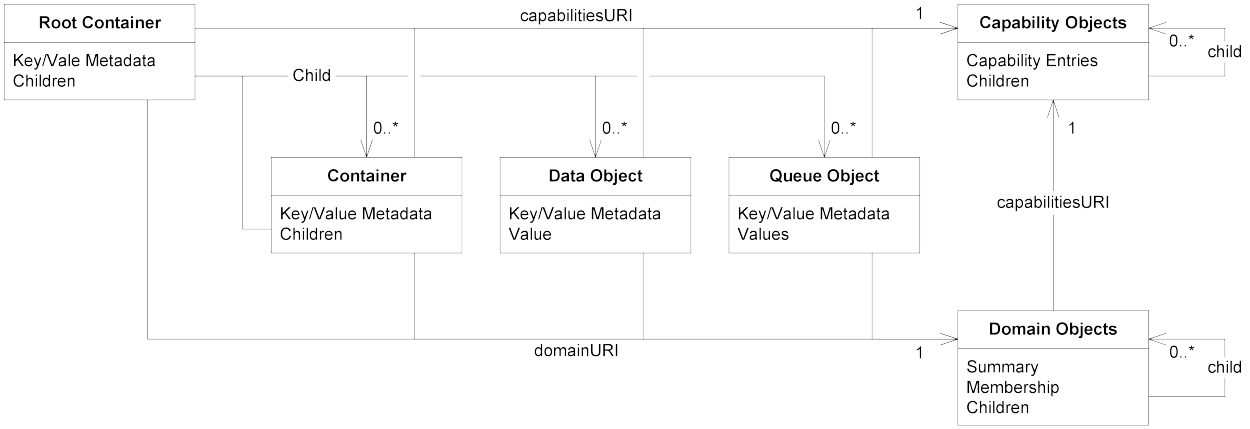


Fig. 4: CDMI Object Model

Table 4: 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. 4), the client may send a PUT via CDMI (see Fig. 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. 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.

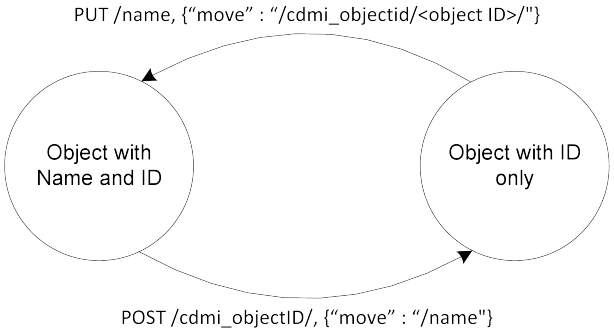


Fig. 5: Object Transitions between Named and ID-only

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.

Table 5: 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 6. The native format of an object ID is a variable-length byte sequence and shall be a maximum length of 40 bytes. A client should treat object IDs as opaque byte strings. 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 6: 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 6 are defined as follows:

- The reserved bytes shall be set to zero.
- The Enterprise Number field shall be the SNMP enterprise number of the offering organization that developed the system that created the object ID, in network byte order. See **RFC 2578** and http://www.iana.org/assignments/enterprise-numbers. 0 is a reserved value.
- The byte at offset 5 shall contain the full length of the object ID, in bytes.
- The CRC field shall contain a 2-byte (16-bit) CRC in network byte order. The CRC field enables the object ID to be validated for integrity. The CRC field shall be generated by running the algorithm (see ISO 14701:2012) across all bytes of the object ID, as defined by the Length field, with the CRC field set to zero. The CRC function shall have the following fields:
  - Name : “CRC-16”,
  - Width : 16,
  - Poly : `0x8005`,
  - Init : `0x0000`,
  - RefIn : True,
  - RefOut : True,
  - XorOut : `0x0000`, and
  - Check : `0xBB3D`.

  This function defines a 16-bit CRC with polynomial `0x8005`, reflected input, and reflected output. This CRC-16 is specified in ISO 14701:2012.
- Opaque data in each object ID shall be unique for a given Enterprise Number.

The native format for an object ID is binary. When necessary, such as when included in URIs and JSON strings, the object ID textual representation shall be encoded using Base16 encoding rules described in **RFC 4648** and shall be case insensitive.

5.4 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 12.1.7 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 the client request does not include an "Authorization" header and authentication is required, the server responds with an HTTP status code of `401 Unauthorized` and a WWW-Authenticate response header. The HTTP client shall then respond with the appropriate Authorization header in a subsequent request. The format of the WWW-Authenticate and Authorization headers varies depending on the type of authentication required.

- HTTP basic authentication involves sending the user name and password in the clear, and it should only be used on a secure network or in conjunction with TLS.
- HTTP digest authentication sends a secure digest of the user name and password (and other information such as a nonce value), and 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 14701:2012.

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
```

The response shall be:

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-container

{
    "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 7 shows how relative URIs are resolved against root URIs

Table 7: Relative URIs Resolved Against Root URIs

Root URI	+ Relative URI	=> Resolved URI
`http://cloud.example.com/`	`container/testObject`	`http://cloud.example.com/container/testObject`
`http://cloud.example.com/`	`/container/testObject`	`http://cloud.example.com/container/testObject`
`http://cloud.example.com/p1/`	`container/testObject`	`http://cloud.example.com/p1/ccontainer/testObject`
`http://cloud.example.com/p1/`	`/container/testObject`	`http://cloud.example.com/container/testObject`
`http://cloud.example.com/p1/p2/`	`container/testObject`	`http://cloud.example.com/p1/p2/container/testObject`
`http://cloud.example.com/p1/p2/`	`/container/testObject`	`http://cloud.example.com/container/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 7.

- 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 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.

CDMI 2.x servers may support the “json” value transfer encoding. Data objects with a value transfer encoding of json shall be accessible to CDMI 1.x clients with a value transfer encoding of UTF-8.

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.

EXAMPLE 1: GET to a URI, where the URI is a reference:

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

The following shows the response.

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

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

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

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

The following shows the response.

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

EXAMPLE 3: PUT to create a reference:

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

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

The following shows the response.

```
HTTP/1.1 201 Created
```

906 EXAMPLE 4: POST to create a reference:

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

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

907 The following shows the response.

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

908 EXAMPLE 5: DELETE to delete a reference:

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

909 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.

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.

Ciphertext representation of encrypted objects are created, accessed, and updated by explicitelyspecifying a MIME type “`application/cms`” or “`application/jose+json`”. Otherwise, a plaintext representation is created, accessed, and updated. For more details on encrypted updates, see clause 23.

6.2 Create a Data Object using HTTP

6.2.1 Synopsis

The following HTTP PUT operation creates a new data object in the specified container:

- `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 8.

Table 8: 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
X-CDMI-Partial	Header String	Indicates that the newly created object is part of a series of writes and has not yet been fully created. When set to `true`, the completionStatus field shall be set to `Processing`. `X-CDMI-Partial` works across CDMI and non-CDMI operations.	Optional
Content-Range	Header String	A valid ranges-specifier (see **RFC 2616** Section 14.35.1)	Optional

6.2.4 Request Message Body

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

6.2.5 Response Headers

No response headers are specified.

6.2.6 Response Message Body

No response message body fields are specified.

6.2.7 Response Status

The HTTP status codes that occur when creating a data object using HTTP are described in Table 9.

Table 9: 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
```

EXAMPLE 2: Put to the container URI to create an encrypted object:

```
PUT /MyContainer/MyEncryptedObject.txt HTTP/1.1
Host: cloud.example.com
Content-Type: application/cms
Content-Length: 1425

<CMS Encrypted Object>
```

The following shows the response:

```
HTTP/1.1 201 Created
```

EXAMPLE 3: PUT to the container URI to create an encrypted object:

967 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 operations read from an existing data object at the specified URI:

- `GET <root URI>/<ContainerName>/<DataObjectName>`
- `GET <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 read from.
- `<DataObjectID>` is the ID of the data object to be read from.

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 10.

Table 10: 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
Accept	Header String	"*/*" or a value as per `clause %s` "Content-type negotiation". If the object has a mimetype of "application/cms" or "application/jose+json", and the mimetype "application/cms" or "application/jose+json" is included in the Accept mimetype, the CDMI server shall return the CMS or JOSE value in the response message body. Otherwise, the decrypted plaintext shall be returned in the response message body, along with the encapsulated mimetype in the Content-Type response header. If decryption is not possible, an error result code shall be returned. (See:numref:*clause %s <clause_cdmi_encrypted_objects>* – Encrypted Objects) If the Accept mimetype list includes "*/*" before "application/cms" and/or "application/jose+json", the server will first try to return the decrypted plaintext, and shall return the CMS or JOSE value when decryption fails. If the Accept mimetype list excludes "*/*", decrypted plaintext shall only be returned if the encapsulated mimetype is included in the Accept mimetype list.	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 11.

Table 11: 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 12.

Table 12: 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
```

1010 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
```

1011 EXAMPLE 3: GET to the data object URI to always return the ciphertext of an encrypted object:

```
GET /MyContainer/MyEncryptedObject.txt HTTP/1.1
Host: cloud.example.com
Accept: application/cms, application/jose+json
```

1012 The following shows the response:

```
HTTP/1.1 200 OK
Content-Type: application/cms
Content-Length: 1425

<CMS Encrypted Object>
```

1013 EXAMPLE 4: GET to the data object URI to read the plaintext of an encrypted object, if possible; otherwise, get the
1014 ciphertext:

```
GET /MyContainer/MyEncryptedObject.txt HTTP/1.1
Host: cloud.example.com
Accept: */*, application/cms, application/jose+json
<Header credentials used to authenticate and access the decryptionkey>
```

1015 The following shows the response:

```
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 252

<Decrypted contents of Encrypted Value>
```

1016 EXAMPLE 5: GET to the data object URI to read the plaintext of an encrypted object:

```
GET /MyContainer/MyEncryptedObject.txt HTTP/1.1
Host: cloud.example.com
<Header credentials used to authenticate and access the decryption key>
```

1017 The following shows the response:

```
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 252

<Decrypted contents of Encrypted Value>
```

6.4 Update a Data Object using HTTP

6.4.1 Synopsis

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

- `PUT <root URI>/<ContainerName>/<DataObjectName>`
- `PUT <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 updated.
- `<DataObjectID>` is the ID of the data object to be read from.

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 13.

Table 13: 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 14.

Table 14: 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 15.

Table 15: 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
Content-Length: 4

that
```

The following shows the response.

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

6.5 Delete a Data Object using HTTP

6.5.1 Synopsis

The following HTTP DELETE 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 16 describes the HTTP status codes that occur when deleting a data object using HTTP.

Table 16: 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 CDMI 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.

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

Table 17: 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 18 describes the HTTP status codes that occur when deleting a container object using HTTP.

1169 Table 18: HTTP Status Codes - Delete a CDMI Container Object using HTTP

1170

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.

1171 ### 7.5.8 Example

1172 EXAMPLE 1: DELETE to the container object URI:

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

1173 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 19.

Table 19: 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 2047** 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	Indicates that the newly created object is part of a series of writes and has not yet been fully created. When set to "`true`", 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 20.

Table 20: Response Header - Create a New Data Object using HTTP

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

7.6.6 Response Message Body

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

7.6.7 Response Status

Table 9 describes the HTTP status codes that occur when creating a new data object using HTTP.

Table 21: HTTP Status Codes - Create a New Data Object using HTTP

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

7.6.8 Examples

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

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

<object contents>
```

The following shows the response.

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

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 mandatory value,
- mandatory fields generated by the cloud storage system,
- mandatory metadata items generated by the cloud storage system,
- optional metadata generated by the cloud storage system; and
- optional metadata specified by the cloud user.

All cloud storage systems shall support data objects, but the ability to create a data object is determiend by the presence or absence of the `cdmi_create_dataobject` and `cdmi_post_dataobject` capabilities in the parent container, and by the `cdmi_post_dataobject_by_ID` system-wide capability for creation by ID.

Each CDMI data object is represented as a JSON object, containing one or more "fields". For example, the "`metadata`" field contains metadata items.

EXAMPLE 1: CDMI Data 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"
    },
    "valuetransferencoding" : "utf-8",
    "valuerange" : "0-36",
    "value" : "This is the Value of this Data Object"
}
```

The meaning, use, and permitted values of each field is described in each operation that creates, modifies or retreives CDMI data objects.

8.1.1 Data Object Addressing

Data objects are addressed in CDMI in two ways:

- by name (e.g. `http://cloud.example.com/dataobject`); and
- by 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.

8.1.2 Data Object Fields

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.

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

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

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

EXAMPLE 3: The following URI returns the value and metadata fields in the response body:

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

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.

8.1.3 Data Object 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.
- If the value transfer encoding of the object is set to "`json`", the data stored in the value of the data object shall contain a valid JSON object, and the value field shall contain a valid JSON object. The JSON stored and returned shall be semantically equivalent but may not be syntactically identical. For example, whitespace outside of JSON-quoted strings may be removed or added by either client libraries or by the server. This means that the number of bytes sent may not be the same as the number of bytes stored.

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

EXAMPLE 4: 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**.

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.

8.1.4 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.

Every data object has a parent object from which the data object inherits data system metadata that is not explicitly specified in the data object itself.

EXAMPLE 5: 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
```

8.1.5 Data Object Access Control

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.

8.1.6 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.

8.1.7 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.

8.1.8 Encrypted Objects

CDMI data object operations only permit management operations and access to the ciphertext of encrypted objects. For more details on encrypted objects, see clause 23.

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 9.6.

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 22.

Table 22: 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 Section 6.2. • 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-Partial	Header String	Indicates that the newly created object is part of a series of writes and has not yet been fully created. When set to "`true`", the `completionStatus` field 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 23.

Table 23: 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
metadata	JSON Object	Metadata for the data object • If this field is included when deserializing, serializing, copying, or moving a data object, the value provided in this field shall replace the metadata from the source URI. • If this field is not included when deserializing, serializing, copying, or moving a data object, the metadata from the source URI shall be used. • If this field is included when creating a new data object by specifying a value, the value provided in this field shall be used as the metadata. • If this field is not included when creating a new data object by specifying a value, an empty JSON object (i.e., "`{}`") shall be assigned as the field value. • This field shall not be included when referencing a data object.	Optional
domainURI	JSON String	URI of the owning domain • If different from the parent domain, the user shall have the "`cross-domain`" privilege (see `cdmi_member_privileges` in Table 63 . • 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 23 – 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]
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**. • If multi-part MIME is being used and this field contains the value of the MIME boundary parameter, the contents of the second MIME part shall be assigned as the field value. • If the serialized data object in the second MIME part does not include a value field, the contents of the third MIME part shall be assigned as the field value of the value field.	Optional[1]

Continued on next page

Table 23 – continued from previous page

Field Name	Type	Description	Requirement
valuetransferencoding	JSON String	The value transfer encoding used for the data object value. Three 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. • "`json`" indicates that the data object contains a valid JSON object, and the `value` field shall be a JSON object containing valid JSON data. If the contents of the `value` field are set to any value other than a valid JSON object, an HTTP status code of `400 Bad Request` shall be 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 CDMI; 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]
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**. • If the `valuetransferencoding` field indicates JSON encoding, the value shall contain a valid JSON object.	Optional[1]

8.2.6 Response Headers

The HTTP response headers for creating a data object using CDMI are shown in Table 24.

Table 24: Response Headers - Create a Data Object using CDMI

Header	Type	Description	Requirement
Content-Type	Header String	"application/cdmi-object"	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 25.

Table 25: 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	A string indicating the percentage of completion if the object is still in the process of bewing created or updated by another operation. • 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

Continued 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.

Table 25 – continued from previous page

Field Name	Type	Description	Requirement
mimetype	JSON String	MIME type of the value of the data object	Mandatory
metadata	JSON Object	Metadata for the data object. This field includes any user and data system metadata specified in the request body metadata field, along with storage system metadata generated by the cloud storage system. See Clause 16 for a further description of metadata.	Mandatory

8.2.8 Response Status

The HTTP status codes that occur when creating a data object using CDMI are described in Table 26.

Table 26: 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

EXAMPLE 1: PUT to the container URI the data object name and contents:

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

{
    "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

{
    "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",
```

(continues on next page)

(continued from previous page)

```
    "metadata" : {
        "cdmi_size" : "37"
    }
}
```

1392 EXAMPLE 2: PUT to the container URI the data object name and binary contents:

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

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

1393 The following shows the response.

```
HTTP/1.1 201 Created
Content-Type: application/cdmi-object

{
   "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"
   }
}
```

1394 EXAMPLE 3: PUT to the container URI the data object name and binary contents using multi-part MIME:

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

--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--
```

1395 The following shows the response.

```
HTTP/1.1 201 Created
Content-Type: application/cdmi-object

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

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

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

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

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

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

<37 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p--
```

The following shows the response.

```
 HTTP/1.1 201 Created
 Content-Type: application/cdmi-object

 {
    "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",
    }
}
```

1399 EXAMPLE 5: PUT to the container URI the data object name and JSON 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" : "json"
  "value" : {
      "test" : "value"
  }
}
```

1400 The following shows the response.

```
 HTTP/1.1 201 Created
 Content-Type: application/cdmi-object

 {
    "objectType": "application/cdmi-object",
    "objectID": "0000706D0010374085EF1A5C7018D774",
    "objectName": "MyDataObject.txt",
    "parentURI": "/MyContainer/",
    "parentID" : "00007ED90010067404EDED32860C086A",
    "domainURI": "/cdmi_domains/MyDomain/",
    "capabilitiesURI": "/cdmi_capabilities/dataobject/",
    "completionStatus": "Complete",
    "mimetype": "text/plain",
    "metadata": {
         "cdmi_size": "21"
    }
}
```

1401 EXAMPLE 6: PUT to the container URI to create an encrypted object:

```
PUT /MyContainer/MyEncryptedObject.txt HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-object

{
 "mimetype" : "application/cms",
 "metadata" : {
 "cdmi_enc_key_id" : "testkey"
 },
 "valuetransferencoding" : "base64"
 "value" : "<CMS Encrypted Object in Base64>"
}
```

1402 The following shows the response:

```
HTTP/1.1 201 Created
```

1403 EXAMPLE 7: PUT to the container URI to create an encrypted object:

```
PUT /MyContainer/MyEncryptedObject2.txt HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-object

{
 "mimetype" : "application/jose+json",
 "metadata" : {
 "cdmi_enc_key_id" : "77c7e2b8-6e13-45cf-8672-617b5b45243a"
 },
```

(continues on next page)

(continued from previous page)

```
 "valuetransferencoding" : "json",
 "value" : {
 "protected": "eyJhbGciOiJkaXIiLCJraWQiOiI3N2M3ZTJi
                          OC02ZTEzLTQ1Y2YtODY3Mi02MTdiNWI0NTI0
                          M2EiLCJlbmMiOiJBMTI4R0NNIn0",
 "iv": "refa467QzzKx6QAB",
 "ciphertext": "JW_i_f52hww_ELQPGaYyeAB6HYGcR559l9T
                                YnSovc23XJoBcW29rHP8yZOZG7YhLpT1bjF
                                uvZPjQS-m0IFtVcXkZXdH_lr_FrdYt9HRUY
                                kshtrMmIUAyGmUnd9zMDB2n0cRDIHAzFVeJ
                                UDxkUwVAE7_YGRPdcqMyiBoCO-FBdE-Nceb
                                4h3-FtBP-c_BIwCPTjb9o0SbdcdREEMJMyZ
                                BH8ySWMVi1gPD9yxi-aQpGbSv_F9N4IZAxs
                                cj5g-NJsUPbjk29-s7LJAGb15wEBtXphVCg
                                yy53CoIKLHHeJHXex45Uz9aKZSRSInZI-wj
                                sY0yu3cT4_aQ3i1o-tiE-F8Ios61EKgyIQ4
                                CWao8PFMj8TTnp",
 "tag": "vbb32Xvllea2OtmHAdccRQ",
 "cty": "text/plain"
 }
}
```

1404 The following shows the response:

```
HTTP/1.1 201 Created
```

8.3 Read a Data Object using CDMI

8.3.1 Synopsis

To read an existing data object, the following requests shall be performed:

- `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>;...`
- `GET <root URI>/cdmi_objectid/<DataObjectID>`
- `GET <root URI>/cdmi_objectid/<DataObjectID>?<fieldname>;<fieldname>;...`
- `GET <root URI>/cdmi_objectid/<DataObjectID>?value:<range>;...`
- `GET <root URI>/cdmi_objectid/<DataObjectID>?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.
- `<DataObjectID>` is the ID of the data object to be read from.

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 27.

Table 27: 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

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 28.

Table 28: Response Headers - Read a CDMI Data Object using CDMI

Header	Type	Description	Requirement
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. • 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 29.

Table 29: 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

Continued on next page

Table 29 – continued from previous page

Field Name	Type	Description	Requirement
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	A string indicating the percentage of completion if the object is still in the process of bewing created or updated by another operation. • When the value of completionStatus is "`Processing`", this field, if provided, shall indicate the percentage of completion as a numeric integer value from 0 through 100. • When the value of completionStatus is "`Complete`", this field, if provided, shall contain the value "`100`". • When the value of completionStatus is "`Error`", this field, if provided, may contain any integer value from "`0`" through "`100`".	Optional
mimetype	JSON String	MIME type of the value of the data object	Mandatory
metadata	JSON Object	Metadata for the data object. This field includes any user and data system metadata specified in the request body metadata field, along with storage system metadata generated by the cloud storage system. See Clause 16 for a further description of metadata.	Mandatory
valuerange	JSON String	The range of bytes of the data object to be returned in the value field • If a specific value range has been requested, the valuerange field shall correspond to the bytes requested. If the request extends beyond the end of the value, the valuerange field shall indicate the smaller byte range returned. • If the object value has gaps (due to PUTs with non-contiguous value ranges), the value range will indicate the range to the first gap in the object value. • The cdmi_size storage system metadata of the data object shall always indicate the complete size of the object, including zero-filled gaps.	Mandatory

Continued on next page

Table 29 – continued from previous page

Field Name	Type	Description	Requirement
valuetransferencoding	JSON String	The value transfer encoding used for the data object value. Three 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. • "`json`" indicates that the data object contains a valid JSON object, and the value field shall contain a valid JSON object.	Mandatory
value	JSON String	The data object value • If the valuetransferencoding field indicates UTF-8 encoding, the value field shall contain a UTF-8 string using JSON escaping rules described in **RFC 4627**. • If the valuetransferencoding field indicates base 64 encoding, the value field shall contain a base 64-encoded string as described in **RFC 4648**. • If the valuetransferencoding field indicates JSON encoding, the value field shall contain a valid JSON object. • The value field shall not be provided when using multi-part MIME. • The value field shall only be provided when the completionStatus field contains "`Complete`". • When reading a value, zeros shall be returned for any gaps resulting from non-contiguous writes.	Conditional

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

8.3.7 Response Status

The HTTP status codes that occur when reading a data object using CDMI are described in Table 30.

Table 30: 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

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

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

The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-object

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

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

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

The following shows the response.

```
HTTP/1.1 200 OK
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"
    },
    "valuetransferencoding" : "utf-8",
    "valuerange" : "0-36",
    "value" : "This is the Value of this Data Object"
}
```

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

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

The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-object

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

EXAMPLE 4: GET to the data object URI to read the first 11 bytes of the value of the data object:

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

The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-object

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

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

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

The following shows the response.

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

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

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

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

<37 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p—
```

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

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

The following shows the response.

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

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

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

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

<37 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p—
```

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

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

The following shows the response.

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

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

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

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

(continues on next page)

(continued from previous page)

```
<11 bytes of binary data>

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

<4 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p--
```

EXAMPLE 8: GET to the data object URI to read the value and valuetransferencoding fields of a data object storing JSON data:

```
GET /cdmi_objectid/0000706D0010374085EF1A5C7018D774?valuetransferencoding;value HTTP/
↪1.1
Host: cloud.example.com
Accept: application/cdmi-object
```

The following shows the response.

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

{
   "valuetransferencoding" : "json"
   "value" : {
      "test" : "value"
      }
}
```

EXAMPLE 9: GET to the data object URI to read a newly-created data object with a current version:

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

The following shows the response.

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

{

  "objectType" : "application/cdmi-object",
  "objectID" : "00007ED900100DA32EC94351F8970400",
  "objectName" : "MyVersionedDataObject.txt",
  "parentURI" : "/MyContainer/",
  "parentID" : "00007E7F00102E230ED82694DAA975D2",
  "domainURI" : "/cdmi_domains/MyDomain/",
  "capabilitiesURI" : "/cdmi_capabilities/dataobject/",
  "completionStatus" : "Complete",
  "mimetype" : "text/plain",
  "metadata" : {
    "cdmi_size" : "33",
    "cdmi_versioning" : "user",
    "cdmi_version_object" : "/cdmi_objectid/00007ED900100DA32EC94351F8970400",
    "cdmi_version_current" : "/cdmi_objectid/00007ED90010512EB55A9304EAC5D4AA",
    "cdmi_version_oldest" : [
      "/cdmi_objectid/00007ED90010512EB55A9304EAC5D4AA"
    ],
    ...
  },
  "valuerange" : "0-32",
  "valuetransferencoding" : "utf-8",
  "value" : "First version of this Data Object"
}
```


1475 EXAMPLE 10: GET to the data object URI to read a data object with two historical versions:

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

1476 The following shows the response.

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

{

  "objectType" : "application/cdmi-object",
  "objectID" : "00007ED900100DA32EC94351F8970400",
  "objectName" : "MyDataObject.txt",
  "parentURI" : "/MyContainer/",
  "parentID" : "00007E7F00102E230ED82694DAA975D2",
  "domainURI" : "/cdmi_domains/MyDomain/",
  "capabilitiesURI" : "/cdmi_capabilities/dataobject/",
  "completionStatus" : "Complete",
  "mimetype" : "text/plain",
  "metadata" : {
    "cdmi_size" : "33",
    "cdmi_versioning" : "user",
    "cdmi_version_object" : "/cdmi_objectid/00007ED900100DA32EC94351F8970400",
    "cdmi_version_current" : "/cdmi_objectid/00007ED90010F077F4EB1C99C87524CC",
    "cdmi_version_oldest" : [
      "/cdmi_objectid/00007ED90010512EB55A9304EAC5D4AA"
    ],
    ...
  },
  "valuerange" : "0-32",
  "valuetransferencoding" : "utf-8",
  "value" : "Third version of this Data Object"
}
```

1477 EXAMPLE 11: GET to the URI of a data object version:

```
GET /cdmi_objectid/00007ED9001005192891EEBE599D94BB HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
```

1478 The following shows the response.

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

{
  "objectType" : "application/cdmi-object",
  "objectID" : "00007ED9001005192891EEBE599D94BB",
  "objectName" : "MyVersionedDataObject.txt",
  "parentURI" : "/MyContainer/",
  "parentID" : "00007E7F00102E230ED82694DAA975D2",
  "domainURI" : "/cdmi_domains/MyDomain/",
  "capabilitiesURI" : "/cdmi_capabilities/dataobject/dataobject_version/",
  "completionStatus" : "Complete",
  "mimetype" : "text/plain",
  "metadata" : {
    "cdmi_size" : "34",
    "cdmi_version_object" : "/cdmi_objectid/00007ED900100DA32EC94351F8970400",
    "cdmi_version_current" : "/cdmi_objectid/00007ED90010F077F4EB1C99C87524CC",
    "cdmi_version_oldest" : [
      "/cdmi_objectid/00007ED90010512EB55A9304EAC5D4AA"
    ],
    "cdmi_version_parent" : "/cdmi_objectid/00007ED90010512EB55A9304EAC5D4AA",
    "cdmi_version_children" : [
      "/cdmi_objectid/00007ED90010F077F4EB1C99C87524CC"
    ],
```

(continues on next page)

(continued from previous page)

```
    ...
  },
  "valuerange" : "0-33",
  "valuetransferencoding" : "utf-8",
  "value" : "Second version of this Data Object"
}
```

8.4 Update a Data Object using CDMI

8.4.1 Synopsis

To update an existing data object, the following requests shall be performed:

- `PUT <root URI>/<ContainerName>/<DataObjectName>`
- `PUT <root URI>/<ContainerName>/<DataObjectName>?value:<range>`
- `PUT <root URI>/<ContainerName>/<DataObjectName>?metadata:<metadataname>;....`
- `PUT <root URI>/cdmi_objectid/<DataObjectID>`
- `PUT <root URI>/cdmi_objectid/<DataObjectID>?value:<range>`
- `PUT <root URI>/cdmi_objectid/<DataObjectID>?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.
- `<DataObjectID>` is the ID of the data object to be updated.

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 31.

1507
1508 Table 31: 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 Section 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-Partial	Header String	Indicates that the newly created object is part of a series of writes and has not yet been fully created. When set to "`true`", the `completionStatus` field shall be set to "`Processing`". `X-CDMI-Partial` works across CDMI and non-CDMI operations. If the completionStatus field had previously been set to "Processing" by including this header in a create or update, the next update without this field shall change the completionStatus field back to "Complete".	Optional

1509 ### 8.4.4 Request Message Body

1510 The request message body fields for updating a data object using CDMI are shown in Table 32.

Table 32: Request Message Body - Update a CDMI 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. 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. If this field is set to "`application/cms`" or "`application/jose+json`", the CDMI server shall encrypt or reencrypt the value of the object in place, using the key specified by the "`cdmi_enc_key_id`" metadata item. If the "`cdmi_enc_key_id`" metadata item is not present, the object ID shall be used as the key identifier. The mimetype of the plaintext shall be stored in the CMS or JWE JSON representation. If a "`cdmi_enc_value_sign_id`" metadata item is present, the encrypted object shall also be signed. If this field is changed from "`application/cms`" or "`application/jose+json`" to any other mimetype, the CDMI server shall decrypt the value of the object in place, replacing the specified mimetype with the mimetype of the encrypted object, if stored as part of the encrypted object. For more details on encrypted objects, see clause 23.	Optional
metadata	JSON Object	Metadata for the data object. If present, the new metadata specified replaces the existing object metadata. If individual metadata items are specified in the URI, only those items are replaced; other items are preserved. See Clause 16 for a further description of metadata.	Optional
domainURI	JSON String	URI of the owning domain • If different from the parent domain, the user shall have the "`cross-domain`" privilege (see `cdmi_member_privileges` in Table 63). • If not specified, the existing domain shall be preserved.	Optional
deserialize	JSON String	URI of a serialized 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]

Continued on next page

Table 32 – 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 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]

Continued on next page

Table 32 – continued from previous page

Field Name	Type	Description	Requirement
valuetransferencoding	JSON String	The value transfer encoding used for the data object value. Three 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. If the contents of the data object value field are set or updated to any value other than a valid UTF-8 string, an HTTP status code of 400 Bad Request shall be returned to the client. • "`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. • "`json`" indicates that the data object contains a valid JSON object and shall be transported as a JSON object in the value field. If the contents of the data object value field are set or updated to any value other than a valid JSON object, an HTTP status code of 400 Bad Request shall be 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

Continued on next page

Table 32 – continued from previous page

Field Name	Type	Description	Requirement
value	JSON String	This field contains the new data for the object. If present, this value replaces the existing value. • If this field is not included and multi-part MIME is being used, the contents of the second and subsequent MIME parts shall be assigned to the corresponding byte ranges of the field value. • If the valuetransferencoding field indicates UTF-8 encoding, the value shall be a UTF-8 string escaped using the JSON escaping rules described in **RFC 4627**. • If the valuetransferencoding field indicates base 64 encoding, the value shall be first encoded using the base 64 encoding rules described in **RFC 4648**. • If the valuetransferencoding field indicates JSON encoding, the value field shall contain a valid JSON object. • If a value range was specified in the request, the new data shall be inserted at the location specified by the range. Any resulting gaps between ranges shall be treated as if zeros had been written and shall be included when calculating the size of the value. When storing a range, the value shall be encoded using base 64, and the valuetransferencoding field shall be set to "`base64`".	Optional[1]

8.4.5 Response Header

The HTTP response header for updating a data object using CDMI is shown in Table 33.

Table 33: 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 34.

[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 34: 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

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

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

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

The following shows the response.

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

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

```
PUT /MyContainer/MyDataObject.txt?mimetype HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-object

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

The following shows the response.

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

EXAMPLE 3: PUT to the data object URI to update a range of the value:

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

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

The following shows the response.

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

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

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

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

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

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

The following shows the response.

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

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

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

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

The following shows the response.

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

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

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

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

The following shows the response.

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

EXAMPLE 7: Delete a single metadata item:

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

{
    "metadata": {}
}
```

The following shows the response.

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

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

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

Host: cloud.example.com
Content-Type: application/cdmi-object

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

The following shows the response.

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

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

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

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

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

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

<37 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p--
```

The following shows the response.

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

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

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

--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--
```

1551 The following shows the response.

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

1552 EXAMPLE 11: PUT to the data object URI to encrypt an existing object:

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

{
 "mimetype" : "application/cms",
 "metadata" : {
 "cdmi_enc_key_id" : "testkey"
 }
}
```

1553 The following shows the response:

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

1554 EXAMPLE 12: PUT to the data object URI to decrypt an existing encrypted object:

1555 The following shows the response:

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

8.5 Delete a Data Object using CDMI

8.5.1 Synopsis

To delete an existing data object, the following requests shall be performed:

- `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.

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

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

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

1580 Table 35: HTTP Status Codes - Delete a CDMI Data Object using CDMI

1581

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.

1582 ### 8.5.8 Example

1583 EXAMPLE 1: DELETE by data object URI:

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

1584 The following shows the response.

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

1585 EXAMPLE 2: DELETE by data object ID:

```
DELETE /cdmi_objectid/00007ED90010D891022876A8DE0BC0FD HTTP/1.1
Host: cloud.example.com
```

1586 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 a set of well-defined fields that include:

- zero or more child objects,
- mandatory fields generated by the cloud storage system,
- mandatory metadata items generated by the cloud storage system,
- optional metadata generated by the cloud storage system; and
- optional metadata specified by the cloud user.

All cloud storage systems shall support containers, but the ability to create a containers is determiend by the presence or absence of the `cdmi_create_container` capability in the parent container.

Each CDMI container object is represented as a JSON object, containing one or more "fields". For example, the "`metadata`" field contains metadata items.

EXAMPLE 1: CDMI Container Object

```
{
    "objectType" : "application/cdmi-container",
    "objectID" : "00007ED900104E1D14771DC67C27BF8B",
    "objectName" : "MyContainer/",
    "parentURI" : "/",
    "parentID" : "00007E7F0010128E42D87EE34F5A6560",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/container/",
    "completionStatus" : "Complete",
    "metadata" : {
        "cdmi_ctime" : "2018-05-16T08:01:02.353Z"
    },
    "childrenrange" : "0-4",
    "children" : [
        "red",
        "green",
        "yellow",
        "orange/",
        "purple/"
    ]
}
```

The meaning, use, and permitted values of each field is described in each operation that creates, modifies or retreives CDMI container objects.

9.1.1 Container Object Addressing

Container objects are addressed in CDMI in two ways:

- by name (e.g. `http://cloud.example.com/container/`); and
- by 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.

When a container object is addressed via more than one unique URIs, 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 Section 9.2).

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

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

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

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

Containers may also be nested.

EXAMPLE 2: 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.

9.1.2 Container Object Fields

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

EXAMPLE 3: The following URI returns just the children field in the response body:

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

EXAMPLE 4: 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
```

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

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

EXAMPLE 5: The following URI would return the children and metadata fields in the response body:

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

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.

9.1.3 Container Object Metadata

The following optional container-specific data system metadata may be provided (see Table 36).

Table 36: 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.

9.1.4 Container Object Access Control

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.

9.1.5 Reserved Container Object Names

This international standard defines reserved container names that should not be used by clients 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.

Reserved container names defined in this specification include:

- "`cdmi_objectid`"
- "`cdmi_domains`"
- "`cdmi_capabilities`"
- "`cdmi_snapshots` "
- "`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_`".

9.1.6 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 37.

Table 37: 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

9.2.5 Request Message Body

The request message body fields for creating a container object using CDMI are shown in Table 38.

Table 38: Request Message Body - Create a Container Object using CDMI

Field Name	Type	Description	Requirement
metadata	JSON Object	Metadata for the container object • If this field is included when deserializing, serializing, copying, or moving a container object, the value provided in this field shall replace the metadata from the source URI. • If this field is not included when deserializing, serializing, copying, or moving a container object, the metadata from the source URI shall be used. • If this field is included when creating a new container object by specifying a value, the value provided in this field shall be used as the metadata. • If this field is not included when creating a new container object by specifying a value, an empty JSON object (i.e., “`{}`”) shall be assigned as the field value. • This field shall not be included when referencing a container object.	Optional
domainURI	JSON String	URI of the owning domain • If different from the parent domain, the user shall have the “`cross-domain`” privilege (see `cdmi_member_privileges` in Table 63 . • If not specified, the existing domain shall be preserved.	Optional
exports	JSON Object	A structure for each protocol enabled for this container object (see `ref_exported_protocols`). This field shall not be included when referencing a container object.	Optional

Continued on next page

Table 38 – continued from previous page

Field Name	Type	Description	Requirement
deserialize	JSON String	URI of a CDMI data object that shall be deserialized to create the new container object, including all child objects inside the source serialized data object (see `ref_serialization/deserialization`). When deserializing a container object, any exported protocols from the original serialized container object are not applied to the newly created container object(s).	Optional[1]
copy	JSON String	URI of a source CDMI container object that shall be copied into the new destination container object. • If the destination container object URI and the copy source object URI both do not specify individual fields, the destination container object shall be a complete copy of the source container object, including all child objects under the source container object. • If the destination container object URI or the copy source object URI specifies individual fields, only the fields specified shall be used to create the destination container object. If specified fields are not present in the source, default field values shall be used. • If the destination container object URI and the copy source object URI both specify fields, an HTTP status code of `400 Bad Request` shall be returned to the client. When copying a container object, exported protocols are not preserved across the copy. If there are insufficient permissions to read the container object at the source URI or create the container object at the destination URI, or if the read operation fails, the copy shall return an HTTP status code of `400 Bad Request`, and the destination container object shall not be created.	Optional[1]
move	JSON String	URI of an existing local or remote CDMI container object (source URI) that shall be relocated, along with all child objects, to the URI specified in the PUT. The contents of the container object and all children, including the object ID, shall be preserved by a move, and the container object and all children of the source URI shall be removed after the objects at the destination have been successfully created. If there are insufficient permissions to read the objects at the source URI, write the objects at the destination URI, or delete the objects at the source URI, or if any of these operations fail, the move shall return an HTTP status code of `400 Bad Request`, and the source and destination are left unchanged.	Optional[1]
reference	JSON String	URI of a CDMI container object that shall be redirected to by a reference. If other fields are supplied when creating a reference, the server shall respond with an HTTP status code of `400 Bad Request`.	Optional[1]
deserializevalue	JSON String	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.	Optional[1]

9.2.6 Response Headers

The HTTP response headers for creating a CDMI container object using CDMI are shown in Table 39.

[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 39: Response Headers - Create a Container Object using CDMI

Header	Type	Description	Requirement
Content-Type	Header String	“application/cdmi-container”	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 40.

Table 40: 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	A string indicating the percentage of completion if the object is still in the process of bewing created or updated by another operation. • When the value of completionStatus is “`Processing`”, this field, if provided, shall indicate the percentage of completion as a numeric integer value from “`0`” through “`100`”. • When the value of completionStatus is “`Complete`”, this field, if provided, shall contain the value “`100`”. • When the value of completionStatus is “`Error`”, this field, if provided, may contain any integer value from “`0`” through “`100`”.	Optional
metadata	JSON Object	Metadata for the container object. This field includes any user and data system metadata specified in the request body metadata field, along with storage system metadata generated by the cloud storage system. See Clause 16 for a further description of metadata.	Mandatory

Continued on next page

Table 40 – continued from previous page

Field Name	Type	Description	Requirement
exports	JSON Object	A structure for each protocol that is enabled for this container object. See `ref_exported_protocols`.	Optional[2]
snapshots	JSON Array of JSON Strings	URI(s) of the snapshot container objects. See `ref_cdmi_snapshots`.	Optional[2]
childrenrange	JSON String	The children of the container expressed as a range. If a range of children is requested, this field indicates the children returned as a range.	Optional
children	JSON Array of JSON Strings	Names of the children objects in the container object. Child container objects end with "/".	Optional

9.2.8 Response Status

Table 41 describes the HTTP status codes that occur when creating a container object using CDMI.

Table 41: 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

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

{
    "metadata" : {

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

(continues on next page)

[2] Returned only if present.

(continued from previous page)

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

1742 The following shows the response.

```
HTTP/1.1 201 Created
Content-Type: application/cdmi-container

{
    "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 an existing container object, the following requests shall be performed:

- `GET <root URI>/<ContainerName>/<TheContainerName>/`
- `GET <root URI>/<ContainerName>/<TheContainerName>/?<fieldname>;<fieldname>;...`
- `GET <root URI>/<ContainerName>/<TheContainerName>/?children:<range>;...`
- `GET <root URI>/<ContainerName>/<TheContainerName>/?metadata:<prefix>;...`
- `GET <root URI>/cdmi_objectid/<ContainerObjectID>/`
- `GET <root URI>/cdmi_objectid/<ContainerObjectID>/?<fieldname>;<fieldname>;...`
- `GET <root URI>/cdmi_objectid/<ContainerObjectID>/?children:<range>;...`
- `GET <root URI>/cdmi_objectid/<ContainerObjectID>/?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.
- `<ContainerObjectID>` is the ID of the data object to be read from.

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 42.

Table 42: 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

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 43: Response Headers - Read a Container Object using CDMI

Header	Type	Description	Requirement
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 44

Table 44: Response Message Body - Read 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 • 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

Continued on next page

Table 44 – continued from previous page

Field Name	Type	Description	Requirement
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	A string indicating the percentage of completion if the object is still in the process of bewing created or updated by another operation. • When the value of completionStatus is "`Processing`", this field, if provided, shall indicate the percentage of completion as a numeric integer value from 0 through 100. • When the value of completionStatus is "`Complete`", this field, if provided, shall contain the value "`100`". • When the value of completionStatus is "`Error`", this field, if provided, may contain any integer value from "`0`" through "`100`".	Optional
metadata	JSON Object	Metadata for the container object. This field includes any user and data system metadata specified in the request body metadata field, along with storage system metadata generated by the cloud storage system. See Clause 16 for a further description of metadata.	Mandatory
exports	JSON Object	A structure for each protocol that is enabled for this container object (see `ref_exported_protocols`)	Optional[1]
snapshots	JSON Array of JSON Strings	URIs of the snapshot container objects	Optional[1]
childrenrange	JSON String	The children of the container expressed as a range. If a range of children is requested, this field indicates the children returned as a range.	Mandatory
children	JSON Array of JSON Strings	Names of the children objects in the container object. When a client uses a child name in a request URI or a header URI, the client shall escape reserved characters according to **RFC 3986**, e.g., a "`%`" character in a child name shall be replaced with "`%25`". • Children that are container objects shall have "`/`" appended to the child name. • Children that are references shall have "`?`" appended to the child name.	Mandatory

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

9.3.7 Response Status

Table 45 describes the HTTP status codes that occur when reading a container object using CDMI.

[1] Returned only if present.

Table 45: HTTP Status Codes - Read a Container Object using CDMI

HTTP Status	Description
200 OK	The metadata for the container object is provided in the message body.
302 Found	The resource is a reference to another resource.
400 Bad Request	The request contains invalid parameters or field names.
401 Unauthorized	The authentication credentials are missing or invalid.
403 Forbidden	The client lacks the proper authorization to perform this request.
404 Not Found	The resource was not found at the specified URI.
406 Not Acceptable	The server is unable to provide the object in the content type specified in the Accept header.

9.3.8 Examples

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

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

The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-container

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

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

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

1794 The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-container

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

1795 EXAMPLE 3: GET to the container object URI to read children 0..2 and childrenrange of the container object:

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

1796 The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-container

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

1797 EXAMPLE 4: GET to the container object by ID to read children 0..2 and childrenrange of the container object:

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

1798 The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-container

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

9.4 Update a Container Object using CDMI

9.4.1 Synopsis

To update an existing container object, the following requests shall be performed:

- `PUT <root URI>/<ContainerName>/<TheContainerName>`
- `PUT <root URI>/<ContainerName>/<TheContainerName>?metadata:<metadataname>;....`
- `PUT <root URI>/cdmi_objectid/<ContainerObjectID>`
- `PUT <root URI>/cdmi_objectid/<ContainerObjectID>?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.
- `<ContainerObjectID>` is the ID of the data object to be updated.

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 46.

Table 46: Request Headers - Update a Container Object using CDMI

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

9.4.5 Request Message Body

The request message body fields for updating a container object using CDMI are shown in Table 47.

Table 47: Request Message Body - Update a Container Object using CDMI

Field Name	Type	Description	Requirement
metadata	JSON Object	Metadata for the container object. If present, the new metadata specified replaces the existing object metadata. If individual metadata items are specified in the URI, only those items are replaced; other items are preserved. See Clause 16 for a further description of metadata.	Optional
domainURI	JSON String	URI of the owning domain • If different from the parent domain, the user shall have the "`cross-domain`" privilege (see `cdmi_member_privileges` in Table 63). • If not specified, the parent domain shall be used.	Optional
snapshot	JSON String	Name of the snapshot to be taken. This is not a URL, but rather, the final component of the absolute URL where the snapshot will exist when the snapshot operation successfully completes. • If a snapshot is added or changed, the PUT operation only returns after the snapshot is added to the snapshot list. • After they are created, snapshots may be accessed as children container objects under the `cdmi_snapshots` child container object of the container object receiving a snapshot. • When creating a snapshot with the same name as an existing snapshot, the new snapshot will replace the existing snapshot.	Optional

Continued on next page

Table 47 – continued from previous page

Field Name	Type	Description	Requirement
deserialize	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.	Optional[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.	Optional[1]
deserializevalue	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.	Optional[1]

Continued on next page

Table 47 – continued from previous page

Field Name	Type	Description	Requirement
exports	JSON Object	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.	Optional

9.4.6 Response Header

The HTTP response header for updating a CDMI container object using CDMI is shown in Table 48.

Table 48: 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 49 describes the HTTP status codes that occur when updating a container object using CDMI.

Table 49: 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

EXAMPLE 1: PUT to the container object URI to replace all metadata with new metadata:

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

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

(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.

(continued from previous page)

```
    }
}
```

1856 The following shows the response.

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

1857 EXAMPLE 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

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

1858 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 requests shall be performed:

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

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.
- `<ContainerObjectID>` is the ID of the container object to be deleted.

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

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

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

Table 50: HTTP Status Codes - Delete a Container Object Using CDMI

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

9.5.8 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
```

EXAMPLE 2: DELETE by container object ID:

```
DELETE /cdmi_objectid/00007ED900104E1D14771DC67C27BF8B/ HTTP/1.1
Host: cloud.example.com
```

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, 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 Section 5.3.1), 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.
- `<DataObjectName>` is the name specified for the data object to be created.

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>`.

If created in "`/cdmi_objectid/`", the data object shall only 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 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.

9.6.3 Capabilities

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

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

9.6.4 Request Headers

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

Table 51: Request Headers - Create a New Data Object Using CDMI

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

Continued on next page

Table 51 – continued from previous page

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 Section 6.2. • 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-Partial	Header String	Indicates that the newly created object is part of a series of writes and has not yet been fully created. When set to "`true`", the `completionStatus` field shall be set to "`Processing`". `X-CDMI-Partial` works across CDMI and non-CDMI operations.	Optional

9.6.5 Request Message Body

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

Table 52: Request Message Body - Create a New Data Object Using CDMI

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 52 – 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 63 . • 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 52 – 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]
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**. • If multi-part MIME is being used and this field contains the value of the MIME boundary parameter, the contents of the second MIME part shall be assigned as the field value. • If the serialized data object in the second MIME part does not include a value field, the contents of the third MIME part shall be assigned as the field value of the value field.	Optional[1]

Continued on next page

Table 52 – continued from previous page

Field Name	Type	Description	Requirement
valuetransferencoding	JSON String	The value transfer encoding used for the data object value. Three 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. • "`json`" indicates that the data object contains a valid JSON object, and the `value` field shall be a JSON object containing valid JSON data. If the contents of the `value` field are set to any value other than a valid JSON object, an HTTP status code of `400 Bad Request` shall be 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]
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**. • If the `valuetransferencoding` field indicates JSON encoding, the value shall contain a valid JSON object.	Optional[1]

9.6.6 Response Headers

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

Table 53: Response Headers - Create a New Data Object Using CDMI

Header	Type	Description	Requirement
Content-Type	Header String	"`application/cdmi-object`"	Mandatory
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 54.

Table 54: 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

Continued 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`.

Table 54 – continued from previous page

Field Name	Type	Description	Requirement
domainURI	JSON String	URI of the owning domain	Mandatory
capabilitiesURI	JSON String	URI to the capabilities for the object	Mandatory
completionStatus	JSON String	A string indicating if the object is still in the process of being created or updated by another operation, and after that operation is complete, indicates if it was successfully created or updated or if an error occurred. The value shall be the string "`Processing`", the string "`Complete`", or an error string starting with the value "`Error`".	Mandatory
percentComplete	JSON String	A string indicating the percentage of completion if the object is still in the process of bewing created or updated by another operation. • When the value of completionStatus is "`Processing`", this field, if provided, shall indicate the percentage of completion as a numeric integer value from "`0`" through "`100`". • When the value of completionStatus is "`Complete`", this field, if provided, shall contain the value "`100`". • When the value of completionStatus is "`Error`", this field, if provided, may contain any integer value from "`0`" through "`100`".	Optional
mimetype	JSON String	MIME type of the value of the data object	Mandatory
metadata	JSON Object	Metadata for the data object. This field includes any user and data system metadata specified in the request body metadata field, along with storage system metadata generated by the cloud storage system. See Clause 16 for a further description of metadata.	Mandatory

9.6.8 Response Status

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

Table 55: HTTP Status Codes - Create a New Data Object Using CDMI

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

9.6.9 Examples

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

```
POST /MyContainer/ HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
Content-Type: application/cdmi-object

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

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

1979 The following shows the response.

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

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

1980 EXAMPLE 2: POST to the object ID URI the data object contents:

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

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

1981 The following shows the response.

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

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

(continues on next page)

(continued from previous page)

```
                "identifier": "OWNER@",
                "aceflags": "NO_FLAGS",
                "acemask": "ALL_PERMS"
            }
        ],
                ...
    }
}
```

1982 EXAMPLE 3: POST to the object ID URI the data object fields and binary contents using multi-part MIME:

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

--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--
```

1983 The following shows the response.

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

{
    "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 Section 11) in a specified container where the name of the queue object is a server-assigned object identifier, the following request shall be performed:

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

To create a new queue object where the queue object does not belong to a container and is only accessible by ID (see Section 5.3.1), 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 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>`.

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

9.7.2 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` 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 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 of the operation that has completed (0 to 100).

GET shall not return any value for the queue 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.

9.7.3 Capabilities

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.

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.

9.7.4 Request Headers

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

Table 56: Request Headers - Create a New Queue Object Using CDMI

Header	Type	Description	Requirement
Accept	Header String	"`application/cdmi-object`" or a consistent value as per clause Section 5.5.2	Optional
Content-Type	Header String	"`application/cdmi-queue`"	Mandatory
Content-Range	Header String	A valid ranges-specifier (see **RFC 2616** Section 14.35.1)	Optional

9.7.5 Request Message Body

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

Table 57: Request Message Body - Create a New Queue Object Using CDMI

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 • If different from the parent domain, the user shall have the "`cross-domain`" privilege (see `cdmi_member_privileges` in Table 63 . • If not specified, the domain of the parent container 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]

[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`.

9.7.6 Response Headers

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

Table 58: Response Headers - Create a New Queue Object Using CDMI

Header	Type	Description	Requirement
Content-Type	Header String	"`application/cdmi-queue`"	Mandatory
Location	Header String	The unique absolute URI for the new queue object as assigned by the system.	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 59.

Table 59: Response Message Body - Create a New 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 • 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

Continued on next page

Table 59 – continued from previous page

Field Name	Type	Description	Requirement
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	A string indicating the percentage of completion if the object is still in the process of bewing created or updated by another operation. • When the value of completionStatus is "`Processing`", this field, if provided, shall indicate the percentage of completion as a numeric integer value from "`0`" through "`100`". • When the value of completionStatus is "`Complete`", this field, if provided, shall contain the value "`100`". • When the value of completionStatus is "`Error`", this field, if provided, may contain any integer value from "`0`" through "`100`".	Optional
metadata	JSON Object	Metadata for the queue object. This field includes any user and data system metadata specified in the request body metadata field, along with storage system metadata generated by the cloud storage system. See Clause 16 for a further description of metadata.	Mandatory
queueValues	JSON String	The range of designators for enqueued values. Every enqueued value shall be assigned a unique, monotonically-incrementing positive integer designator, starting from 0. If no values are enqueued, an empty string shall be returned. If values are enqueued, the lowest designator, followed by a hyphen ("-"), followed by the highest designator shall be returned.	Mandatory

9.7.8 Response Status

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

Table 60: HTTP Status Codes - Create a New Queue Object Using CDMI

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

9.7.9 Example

EXAMPLE 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

{
}
```

The following shows the response.

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

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

Section IV

CDMI Advanced

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. Each object may be owned and managed by a different administrative entity, which is expressed as a domain.

If a cloud storage system supports domains, the `cdmi_domains` system-wide capability shall be present, and the `cdmi_domains` container shall be present in the CDMI root container.

A cloud storage system 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.

FIXME - Add diagram (Issue #96)

Each CDMI domain object is represented as a JSON object, containing one or more “fields”. For example, the “`metadata`” field contains metadata items.

EXAMPLE 1: CDMI Domain Object

```
{
    "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/"
    ]
}
```

The meaning, use, and permitted values of each field is described in each operation that creates, modifies or retreives CDMI domain objects.

10.1.1 Domain Object Addressing

Domain objects are created in a special `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.

Domain objects are addressed in CDMI in two ways:

- by name (e.g., `http://cloud.example.com/cdmi_domains/myDomain/`); and
- by 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 at least one URI address that allows the domain object to be accessed. Following the URI conventions for hierarchical paths, domain URIs shall start with "`<root URI>/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`.

Domain objects may also be nested.

EXAMPLE 2: The following URI represents a nested domains:

```
http://cloud.example.com/cdmi_domains/myDomain/subDomain/
```

A nested domain has a parent domain object, shall be included in the children field of the parent domain object, and shall inherit <FIXME> from its parent domain.

10.1.2 Domain Object Fields

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

EXAMPLE 3: The following URI returns just the children field in the response message body:

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

EXAMPLE 4: By specifying a range after the children field name, specific ranges of the children field may be accessed.

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

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

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

EXAMPLE 5: The following URI would return the children and metadata fields in the response body:

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

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.

10.1.3 Domain Object Metadata

The following domain-specific field shall be present for each domain (see Table 61).

Table 61: Required metadata for a domain object

Metadata Name	Type	Description	Requirement
cdmi_domain_enabled	JSON String	Indicates if the domain is enabled and specified at the time of creation. Values shall be "`true`" or "`false`". • If a domain is disabled, the cloud storage system shall not permit any operations to be performed against any URI managed by that domain. • If this metadata item is not present at the time of domain creation, the value is set to "`false`".	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 16.2 and Section 16.3 Domain data system metadata shall be inherited to child domain objects.

10.1.4 Domain Object Access Control

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.

10.1.5 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.

10.1.6 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 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 17.1) 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.

EXAMPLE 1: If a domain were created on July 4, 2009, at noon, the daily summary "2009-07-04" would contain information from noon until midnight, the monthly summary "2009-07" would contain information from noon on July 4 until midnight on July 31, and the yearly summary "2009" would contain information from noon on July 4 until midnight on December 31.

If a time period starts after the time when the domain was created and ends earlier than the time of access, the summary data object contains complete information for that time period.

EXAMPLE 2: If a domain were created on July 4, 2009, and on July 10, the "2009-07-06" daily summary data object was accessed, it would contain information for the complete day.

If a time period ends after the current access time, the domain summary data object contains partial information from the start of the time period (or the time the domain was created) until the time of access.

EXAMPLE 3: If a domain were created on July 4, 2009, and at noon on July 10, the "2009-07-10" daily summary data object was accessed, it would contain information from the beginning of the day until noon.

The information in Table 62 shall be present within the contents of each domain summary object, which are in JSON representation.

Table 62: Contents of domain summary objects

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

Continued on next page

Table 62 – continued from previous page

Metadata Name	Type	Description	Requirement
cdmi_summary_end	JSON String	An ISO-8601 time indicating the end of the time range that the summary information is presenting	Mandatory
cdmi_summary_objecthours	JSON String	The sum of the time each object belonging to the domain existed during the summary time period	Optional
cdmi_summary_objectsmin	JSON String	The minimum number of objects belonging to the domain during the summary time period	Optional
cdmi_summary_objectsmax	JSON String	The maximum number of objects belonging to the domain during the summary time period	Optional
cdmi_summary_objectsaverage	JSON String	The average number of objects belonging to the domain during the summary time period	Optional
cdmi_summary_puts	JSON String	The number of objects written to the domain	Optional
cdmi_summary_gets	JSON String	The number of objects read from the domain	Optional
cdmi_summary_bytehours	JSON String	The sum of the time each byte belonging to the domain existed during the summary time period	Optional
cdmi_summary_bytesmin	JSON String	The minimum number of bytes belonging to the domain during the summary time period	Optional
cdmi_summary_bytesmax	JSON String	The maximum number of bytes belonging to the domain during the summary time period	Optional
cdmi_summary_bytesaverage	JSON String	The average number of bytes belonging to the domain during the summary time period	Optional
cdmi_summary_writes	JSON String	The number of bytes written to the domain	Optional
cdmi_summary_reads	JSON String	The number of bytes read from the domain	Optional
cdmi_summary_charge	JSON String	An ISO 4217 currency code (see `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

2197 EXAMPLE 4: An example of a daily domain summary object is as follows:

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

(continues on next page)

(continued from previous page)

```
    "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_".

10.3 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 17.1), 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 10.1.5). 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 17.1) that restricts access to this information.

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

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

The domain membership container may also contain subcontainers with data objects. Data objects in these subcontainers are treated the same as data objects in the domain membership container, and no meaning is inferred from the subcontainer name. This organization is used to create different access security relationships for groups of user objects and to allow delegation to a common set of members.

Table 63 lists the domain settings that shall be present within each domain member user object.

Table 63: Required settings for domain member user objects

Metadata Name	Type	Description	Requirement
cdmi_member_enabled	JSON String	If true, this field indicates that requests associated with this domain member are allowed. If false, all requests performed by this domain member shall result in an HTTP status code of `403 Forbidden`.	Mandatory
cdmi_member_type	JSON String	This field indicates the type of member record. Values include "`user`", "`group`", and "`delegation`".	Mandatory
cdmi_member_name	JSON String	This field contains the user or group name as presented by the client. This will normally be the standard full name of the principal.	Mandatory
cdmi_member_credentials	JSON String	This field contains credentials to be matched against the credentials as presented by the client. If this field is not present, one or more delegations shall be present and shall be used to resolve user credentials. As one cannot log in as a group but only as a member of a group, the "`group`" type member records shall not have credentials.	Optional
cdmi_member_principal	JSON String	This field indicates to which principal name (used in ACLs) the user or group is mapped. If this field is not present, one or more delegations shall be present and shall be used to resolve the principal.	Optional

Continued on next page

Table 63 – continued from previous page

Metadata Name	Type	Description	Requirement
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

2228 Table 64 lists the domain settings that shall be present within each domain member delegation object.

Table 64: Required settings for domain member delegation objects

Metadata Name	Type	Description	Requirement
cdmi_member_enabled	JSON String	If true, this field indicates that requests associated with this domain member are allowed. If false, all requests performed by this domain member shall result in an HTTP status code of 403 Forbidden.	Mandatory
cdmi_member_type	JSON String	This field indicates the type of member record. Values include "user" and "delegation".	Mandatory
cdmi_delegation_URI	JSON String	This field contains the URI of an external identity resolution provider (such as LDAP or Active Directory) or the URI of a domain membership container object. External delegations are expressed in the form of ldap://<uri> or ad://<uri>.	Mandatory

2229 EXAMPLE 1: An example of a domain membership object for a user is as follows:

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

2230 EXAMPLE 2: An example of a domain membership object for a delegation is as follows:

```
{
    "cdmi_member_enabled" : "true",
    "cdmi_member_type" : "delegation",
    "cdmi_delegation_URI" : "/cdmi_domains/MyDomain/"
}
```

10.4 Create a Domain Object using CDMI

10.4.1 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, with one slash (i.e., "`/`") between each pair of domain names.
- `<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>/`.

10.4.2 Delayed Completion of Create

Delayed completion shall not be supported for creating domain objects.

10.4.3 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.

10.4.4 Request Headers

The HTTP request headers for creating a CDMI domain object using CDMI are shown in Table 65

Table 65: Request headers - Create a domain object using CDMI

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

10.4.5 Request Message Body

The request message body fields for creating a domain object using CDMI are shown in Table 66.

Table 66: Request Message Body Create a Domain Object using CDMI

Field Name	Type	Description	Requirement
metadata	JSON Object	Metadata for the domain object • If this field is included when deserializing, serializing, copying, or moving a domain object, the value provided in this field shall replace the metadata from the source URI. • If this field is not included when deserializing, serializing, copying, or moving a domain object, the metadata from the source URI shall be used. • If this field is included when creating a new domain object by specifying a value, the value provided in this field shall be used as the metadata. • If this field is not included when creating a new domain object by specifying a value, an empty JSON object (i.e., "`{}`") shall be assigned as the field value.	Optional
copy	JSON String	URI of a CDMI domain that shall be copied into the new domain, including all child domains and membership from the source domain	Optional[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]
deserializevalue	JSON String	A domain object serialized as specified in Clause 15 and encoded using base 64 encoding rules described in **RFC 4648**.	Optional[1]

10.4.6 Response Headers

The HTTP response headers for creating a domain object using CDMI are shown in Table 67

Table 67: Response headers - Create a domain object using CDMI

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

[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`.

10.4.7 Response Message Body

The response message body fields for creating a domain object using CDMI are shown in Table 68

Table 68: Response message body - Create a domain object using CDMI

Field Name	Type	Description	Requirement
objectType	JSON String	"application/cdmi-domain"	Mandatory
objectID	JSON String	Object ID of the domain	Mandatory
objectName	JSON String	Name of the object	Mandatory
parentURI	JSON String	URI for the parent 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. 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

10.4.8 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 69: HTTP status codes - Create a domain object using CDMI

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

10.4.9 Example

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

"metadata":
{
    "cdmi_domain_enabled": "true"
}
```

2265 The following shows the response.

```
HTTP/1.1 201 Created
Content-Type: application/cdmi-domain

{
    "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/"
    ]
}
```

10.5 Read a Domain Object using CDMI

10.5.1 Synopsis

To read an existing domain object, the following requests shall be performed:

- `GET <root URI>/cdmi_domains/<DomainName>/<TheDomainName>/`
- `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>;...`
- `GET <root URI>/cdmi_objectid/<DomainObjectID>/`
- `GET <root URI>/cdmi_objectid/<DomainObjectID>/?<fieldname>;<fieldname>;...`
- `GET <root URI>/cdmi_objectid/<DomainObjectID>/?children:<range>;...`
- `GET <root URI>/cdmi_objectid/<DomainObjectID>/?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.
- `<DomainObjectID>` is the ID of the domain object to be read from.

10.5.2 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.

10.5.3 Request Headers

The HTTP request headers for reading a CDMI domain object using CDMI are shown in Table 70.

Table 70: Request Headers - Read a Domain Object using CDMI

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

10.5.4 Request Message Body

A request body shall not be provided.

10.5.5 Response Headers

The HTTP response headers for reading a CDMI domain object using CDMI are shown in Table 71.

Table 71: Response Headers - Read a Domain Object using CDMI

Header	Type	Description	Requirement
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

10.5.6 Response Message Body

The response message body fields for reading a CDMI domain object using CDMI are shown in Table 72

Table 72: Response Message Body - Read a Domain Object using CDMI

Field Name	Type	Description	Requirement
objectType	JSON String	"application/cdmi-domain"	Mandatory
objectID	JSON String	Object ID of the domain	Mandatory
objectName	JSON String	Name of the object	Mandatory
parentURI	JSON String	URI for the parent object Appending the "objectName" to the "parentURI" shall always produce a valid URI for the object.	Mandatory
parentID	JSON String	Object ID of the parent domain object • For domain objects directly under "cdmi_domains", the object ID of "cdmi_domains" container shall be returned. • For domain objects under another domain, the object ID of the parent domain shall be returned.	Mandatory
domainURI	JSON String	URI of the owning domain. A domain object shall always be 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.

10.5.7 Response Status

Table 73 describes the HTTP status codes that occur when reading a domain object using CDMI.

Table 73: HTTP Status Codes Read a Domain Object using CDMI

HTTP Status	Description
200 OK	The domain object content was returned in the response.
302 Found	The resource is a reference to another resource.
400 Bad Request	The request contains invalid parameters or field names.
401 Unauthorized	The authentication credentials are missing or invalid.
403 Forbidden	The client lacks the proper authorization to perform this request.
404 Not Found	The resource was not found at the specified URI.
406 Not Acceptable	The server is unable to provide the object in the content type specified in the Accept header.

10.5.8 Examples

EXAMPLE 1: GET to the domain URI to read all the fields of the domain:

```
GET /cdmi_domains/MyDomain/ HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-domain
```

The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-domain

{
    "objectType": "application/cdmi-domain",
    "objectID": "00007E7F00104BE66AB53A9572F9F51E",
    "objectName": "MyDomain/",
    "parentURI": "/cdmi_domains/",
    "parentID": "00007E7F0010C058374D08B0AC7B3550",
    "domainURI": "/cdmi_domains/MyDomain/",
    "capabilitiesURI": "/cdmi_capabilities/domain/",
    "metadata": {
        "cdmi_domain_enabled": "true",
        "cdmi_authentication_methods": "anonymous, basic",
            ...
    },
    "childrenrange": "0-1",
    "children": [
        "cdmi_domain_summary/",
        "cdmi_domain_members/"
    ]
}
```

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

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

The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-domain

{
```

(continues on next page)

(continued from previous page)

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

2311 EXAMPLE 3: GET to the domain URI to read the first two children of the domain:

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

2312 The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-domain

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

10.6 Update a Domain Object using CDMI

10.6.1 Synopsis

To update an existing domain object, the following requests shall be performed:

- `PUT <root URI>/cdmi_domains/<DomainName>/<TheDomainName>/`
- `PUT <root URI>/cdmi_domains/<DomainName>/<TheDomainName>/?metadata:<metadataname>; ...`
- `PUT <root URI>/cdmi_objectid/<DomainObjectID>`
- `PUT <root URI>/cdmi_objectid/<DomainObjectID>?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 read from.
- `<DomainObjectID>` is the ID of the data object to be read from.

10.6.2 Delayed Completion of Update

Delayed completion shall not be supported for creating domain objects.

10.6.3 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.

10.6.4 Request Headers

The HTTP request headers for updating a CDMI domain object using CDMI are shown in Table 74.

Table 74: Request Headers - Update a Domain Object using CDMI

Header	Type	Description	Requirement
Content-Type	Header String	“`application/cdmi-domain`”	Mandatory

10.6.5 Request Message Body

The request message body fields for updating a domain object using CDMI are shown in Table 75.

Table 75: Request Message Body - Update a domain object using CDMI

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 object itself shall be copied, not any sub-domains of the domain object. • If the destination domain object URI and the copy source domain object URI both do not specify individual fields, the destination domain object metadata and membership shall be replaced with the source domain object metadata and membership. • If the destination domain object URI or the copy source domain object URI specifies individual fields, only the fields specified shall be used to update the destination domain object. If specified fields are not present in the source, these fields shall be ignored. • If the destination domain object URI and the copy source domain 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 domain object at the source URI or create the domain 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 domain object shall not be updated.	Optional[2]
deserialize	JSON String	The URL of 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. Otherwise, the server shall return an HTTP status code of `400 Bad Request`. • If the serialized domain object does not contain sub-domains, the update is applied only to the domain object, and any existing sub-domains are left as is. • If the serialized domain object does contain sub-domains, then creates, updates, and deletes are recursively applied for each sub-domain, depending on the differences between the provided serialized state and the current state of the sub-domains.	Optional[2]

Continued on next page

Table 75 – continued from previous page

Field Name	Type	Description	Requirement
deserializevalue	JSON String	The value of 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. Otherwise, the server shall return an HTTP status code of `400 Bad Request`. • If the serialized domain object does not contain sub-domains, the update is applied only to the domain object, and any existing sub-domains are left as is. • If the serialized domain object does contain sub-domains, then creates, updates, and deletes are recursively applied for each sub-domain, depending on the differences between the provided serialized state and the current state of the sub-domains.	Optional[2]

10.6.6 Response Header

The HTTP response header for updating a CDMI domain object using CDMI is shown in Table 76

Table 76: Response Header - Update a domain object using CDMI

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

10.6.7 Response Message Body

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

10.6.8 Response Status

Table 77 describes the HTTP status codes that occur when updating a domain object using CDMI.

Table 77: HTTP Status Codes - Update a Domain Object using CDMI

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

10.6.9 Example

EXAMPLE 1: PUT to the domain URI to set new field values:

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

```
PUT /cdmi_domains/MyDomain/ HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-domain

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

The following shows the response.

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

10.7 Delete a Domain Object using CDMI

10.7.1 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>/`
- `DELETE <root URI>/cdmi_objectid/<DomainObjectID>`

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.
- `<DomainObjectID>` is the ID of the domain object to be deleted.

10.7.2 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.

10.7.3 Request Headers

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

10.7.4 Request Message Body

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

10.7.5 Response Headers

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

10.7.6 Response Message Body

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

10.7.7 Response Status

Table 78 describes the HTTP status codes that occur when deleting a domain object using CDMI.

Table 78: HTTP status codes - Delete a Domain Object using CDMI

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

10.7.8 Example

EXAMPLE 1: DELETE to the domain object URI:

```
DELETE /cdmi_domains/MyDomain/ HTTP/1.1
Host: cloud.example.com
```

The following shows the response.

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

Clause 11

Queue Object Resource Operations using CDMI

11.1 Overview

Queue objects are similar to data objects, only with first-in, first-out access “queue”-style accesse semantics when storing and retrieving value data.

If a cloud storage system supports queues, the `cdmi_queues` system-wide capability shall be present. The ability to create a queue object is determiend by the presence or absence of the `cdmi_create_queue` and `cdmi_post_queue` capabilities in the parent container, and by the `cdmi_post_queue_by_ID` system-wide capability for creation by ID.

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. Queues 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 Section 9.7 and this clause.

FIXME - Add diagram (Issue #120)

Each CDMI queue object is represented as a JSON object, containing one or more “fields”. For example, the “`metadata`” field contains metadata items.

EXAMPLE 1: CDMI Queue Object

```
{
    "objectType": "application/cdmi-queue",
    "objectID": "00007E7F00104BE66AB53A9572F9F51E",
    "objectName": "MyQueue",
    "parentURI": "/MyContainer/",
    "parentID" : "00007ED900104F67307652BAC9A37C93",
    "domainURI": "/cdmi_domains/MyDomain/",
    "capabilitiesURI": "/cdmi_capabilities/queue/",
    "completionStatus": "Complete",
    "metadata": {},
    "queueValues": "1-1",
    "mimetype": [
        "text/plain"
    ],
    "valuerange": [
        "0-19"
    ],
    "valuetransferencoding": [
        "utf-8"
    ],
    "value": [
        "First Enqueued Value"
    ]
}
```

The meaning, use, and permitted values of each field is described in each operation that creates, modifies or retreives CDMI queue objects.

11.1.1 Queue Object Addressing

Queue objects are addressed in CDMI in two ways:

- by name (e.g., `http://cloud.example.com/queueobject`); and
- by ID (e.g., `http://cloud.example.com/cdmi_objectid/00007ED900104F67307652BAC9A37C93/`).

Every queue object has a single, globally-unique object ID that remains constant for the life of the object. Each queue object may also have one or more URI addresses that allow the queue object to be accessed.

11.1.2 Queue Object Fields

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

EXAMPLE 2: The following URI returns just the number of values stored in the queue object in the response body:

```
http://cloud.example.com/queueobject?queueValues
```

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

EXAMPLE 3: The following URI returns the number of values stored and metadata fields in the response body:

```
http://cloud.example.com/queueobject?queueValues;metadata
```

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.

11.1.3 Queue Object Value

The encoding of the data stored 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.
- If the value transfer encoding of the object is set to "`json`", the data stored in the value of the queue object shall be a valid JSON object, and the value field shall contain a valid JSON object.

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

EXAMPLE 4: 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.

Multiple non-overlapping ranges of the value of a queue 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.

11.1.4 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. 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.

Every queue object has a parent object from which the queue object inherits data system metadata that is not explicitly specified in the data object itself.

EXAMPLE 5: The “pages” queue object stored at the following URI would inherit data system metadata from its parent container, “OCR”:

```
http://cloud.example.com/OCR/pages
```

11.1.5 Queue Object Access Control

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.

11.1.6 Queue Object Consistency

Writing to a queue object is an atomic operation.

For non-value-related fields:

- If a client reads a queue object simultaneously with a write to that same queue 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 queue object shall be as if the write never occurred (i.e., writes are atomic in the face of errors).

For value-related fields:

- If a client dequeues or deletes one or more queue values simultaneously with one or more queue values being enqueued to that same queue object, the order of operations shall be as if the dequeue/delete operation happens before the enqueue operation.
- If a dequeue, delete or enqueue is terminated due to errors, the contents of the queue object shall be as if the dequeue/delete/enqueue 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.

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

11.1.7 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 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.
- `<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` 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 queue 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 queue 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.

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 79

Table 79: Request Headers - Create A Queue Object Using CDMI

Header	Type	Description	Requirement
Accept	Header String	“`application/cdmi-queue`”	Mandatory
Content-Type	Header String	“`application/cdmi-queue`”	Mandatory

11.2.5 Request Message Body

The request message body fields for creating a queue object using CDMI are shown in Table 80.

Table 80: Request Message Body - Create A 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., "`{}`") shall be assigned as the field value. • This field shall not be included when referencing a queue object.	Optional
domainURI	JSON String	URI of the owning domain • If different from the parent domain, the user shall have the "`cross_domain`" privilege (see `cdmi_member_privileges` in Table 63). • 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 queue object	Optional[1]
copy	JSON String	URI of a source CDMI queue object that shall be copied into the new destination queue object. • If the destination queue object URI and the copy source queue object URI both do not specify individual fields, the destination queue object shall be a complete copy of the source queue object, including all enqueued values. • If the destination queue object URI or the copy source queue object URI specifies individual fields, only the fields specified shall be used to create the destination queue object. If specified fields are not present in the source, default field values shall be used. • If the destination queue object URI and the copy source queue object URI both specify fields, an HTTP status code of `400 Bad Request` shall be returned to the client. If there are insufficient permissions to read the queue object at the source URI or create the queue object at the destination URI, or if the read operation fails, the copy shall return an HTTP status code of `400 Bad Request`, and the destination queue object shall not be created.	Optional[1]

Continued on next page

Table 80 – continued from previous page

Field Name	Type	Description	Requirement
move	JSON String	URI of an existing local or remote CDMI queue object (source URI) that shall be relocated to the URI specified in the PUT. The contents of the queue object, including the object ID, shall be preserved by a move, and the queue object at the source URI shall be removed after the queue object at the destination has been successfully created. If there are insufficient permissions to read the queue object at the source URI, write the queue object at the destination URI, or delete the queue object at the source URI, or if any of these operations fail, the move shall return an HTTP status code of `400 Bad Request`, and the source and destination are left unchanged.	Optional[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]

11.2.6 Response Status

The HTTP response headers for creating a CDMI queue object using CDMI are shown in Table 81

Table 81: Response Headers - Create A Queue Object Using CDMI

Header	Type	Description	Requirement
Content-Type	Header String	"`application/cdmi-queue`"	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 82

Table 82: 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

Continued 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`.

Table 82 – continued from previous page

Field Name	Type	Description	Requirement
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	A string indicating the percentage of completion if the object is still in the process of bewing created or updated by another operation. • When the value of completionStatus is "`Processing`", this field, if provided, shall indicate the percentage of completion as a numeric integer value from "`0`" through "`100`". • When the value of completionStatus is "`Complete`", this field, if provided, shall contain the value "`100`". • When the value of completionStatus is "`Error`", this field, if provided, may contain any integer value from "`0`" through "`100`".	Optional
metadata	JSON Object	Metadata for the queue object. This field includes any user and data system metadata specified in the request body metadata field, along with storage system metadata generated by the cloud storage system. See Clause 16 for a further description of metadata.	Mandatory
queueValues	JSON String	The range of designators for enqueued values. Every enqueued value shall be assigned a unique, monotonically-incrementing positive integer designator, starting from 0. If no values are enqueued, an empty string shall be returned. If values are enqueued, the lowest designator, followed by a hyphen ("-"), followed by the highest designator shall be returned.	Mandatory

11.2.8 Response Status

The HTTP status codes that occur when creating a queue object using CDMI are described in Table 83.

Table 83: 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

Example 1: PUT to the queue URI the queue object name and contents:

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

{
    "metadata" : {

    }
}
```

The following shows the response.

```
HTTP/1.1 201 Created
Content-Type: application/cdmi-queue

{
    "objectType" : "application/cdmi-queue",
    "objectID" : "00007E7F00104BE66AB53A9572F9F51E",
    "objectName" : "MyQueue",
    "parentURI " : "/MyContainer/",
    "parentID" : "00007ED900104F67307652BAC9A37C93",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/queue/",
    "completionStatus" : "Complete",
    "metadata" : {
        ...
    },
    "queueValues" : ""
}
```

EXAMPLE 2: PUT to the queue object URI to create a new queue, copying from another queue:

```
PUT /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-queue

{
    "copy": "/MyContainer/SourceQueue?value:0-9"
}
```

The following shows the response.

```
HTTP/1.1 201 Created
Content-Type: application/cdmi-queue

{
    "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>`
- `GET <root URI>/<ContainerName>/<QueueName>?<fieldname>;<fieldname>;...`
- `GET <root URI>/<ContainerName>/<QueueName>?value:<range>;...`
- `GET <root URI>/<ContainerName>/<QueueName>?metadata:<prefix>;...`
- `GET <root URI>/<ContainerName>/<QueueName>?values:<count>`
- `GET <root URI>/cdmi_objectid/<QueueObjectID>`
- `GET <root URI>/cdmi_objectid/<QueueObjectID>?<fieldname>;<fieldname>;...`
- `GET <root URI>/cdmi_objectid/<QueueObjectID>?value:<range>;...`
- `GET <root URI>/cdmi_objectid/<QueueObjectID>?metadata:<prefix>;...`
- `GET <root URI>/cdmi_objectid/<QueueObjectID>?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.
- `<QueueObjectID>` is the ID of the queue object to be read from.

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 84

Table 84: 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 the corresponding queue value.	Optional

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 85.

Table 85: Response Headers - Read a Queue Object Using CDMI

Header	Type	Description	Requirement
Content-Type	Header String	"`application/cdmi-queue`" or "`multipart/mixed`"	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 86

Table 86: Response Message Body - Read 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 • 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

Continued on next page

Table 86 – continued from previous page

Field Name	Type	Description	Requirement
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	A string indicating the percentage of completion if the object is still in the process of bewing created or updated by another operation. • When the value of completionStatus is "Processing", this field, if provided, shall indicate the percentage of completion as a numeric integer value from 0 through 100. • When the value of completionStatus is "Complete", this field, if provided, shall contain the value "100". • When the value of completionStatus is "Error", this field, if provided, may contain any integer value from "0" through "100".	Optional
metadata	JSON Object	Metadata for the queue object. This field includes any user and data system metadata specified in the request body metadata field, along with storage system metadata generated by the cloud storage system. See Clause 16 for a further description of metadata.	Mandatory

Continued on next page

Table 86 – continued from previous page

Field Name	Type	Description	Requirement
queueValues	JSON String	The range of designators for enqueued values. Every enqueued value shall be assigned a unique, monotonically-incrementing positive integer designator, starting from 0. If no values are enqueued, an empty string shall be returned. If values are enqueued, the lowest designator, followed by a hyphen ("-"), followed by the highest designator shall be returned. • This field shall only be provided when completionStatus is "Complete" and when one or more values are enqueued.	Mandatory
mimetype	JSON Array of JSON Strings	MIME types for each queue object value * The MIME types of the values are returned, each corresponding to the value in the same position in the JSON array. * This field shall only be provided when completionStatus is "Complete" and when one or more values are enqueued.	Optional
valuerange	JSON Array of JSON Strings	The range of bytes of the queue object values to be returned in the value field • The value ranges of the values are returned, each corresponding to the value in the same position in the JSON array. • If a specific value range has been requested, the entry in the valuerange field shall correspond to the bytes requested. If the request extends beyond the end of the value, the valuerange field shall indicate the smaller byte range returned. • This field shall only be provided when completionStatus is "Complete" and when one or more values are enqueued.	Optional
valuetransferencoding	JSON Array of JSON Strings	The value transfer encoding used for each queue object value. Two value transfer encodings are defined: • "utf-8" indicates that the queue object value contains a valid UTF-8 string, and it shall be transported as a UTF-8 string in the value field. • "base64" indicates that the queue object value may contain arbitrary binary sequences, and it shall be transported as a base 64-encoded string in the value field. • "json" indicates that the queue object value contains a valid JSON object, and the value field shall contain a JSON object. The value transfer encodings are returned, each corresponding to the value in the same position in the JSON array. • This field shall only be provided when completionStatus is "Complete" and when one or more values are enqueued.	Optional

Continued on next page

Table 86 – continued from previous page

Field Name	Type	Description	Requirement
value	JSON Array of JSON Strings	The oldest enqueued queue object values • The values in the JSON array are returned in order from oldest to newest. • If the `valuetransferencoding` field indicates UTF-8 encoding, the corresponding value field shall contain a UTF-8 string using JSON escaping rules described in **RFC 4627**. • If the `valuetransferencoding` field indicates base 64 encoding, the corresponding value field shall contain a base 64-encoded string as described in RFC **RFC 4648**. • If the `valuetransferencoding` field indicates JSON encoding, the corresponding value field shall contain a JSON object. • The value field shall not be provided when using multi-part MIME. • The value field shall only be provided when the completionStatus field contains "`Complete`".	Conditional

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

11.3.7 Response Status

The HTTP status codes that occur when reading a queue object using CDMI are described in Table 87.

Table 87: HTTP Status Codes - Read A Queue Object Using CDMI

HTTP Status	Description
`200 OK`	The queue object content was returned in the response.
`302 Found`	The resource is a reference to another resource.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.
`406 Not Acceptable`	The server is unable to provide the object in the content type specified in the Accept header.

11.3.8 Examples

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

```
GET /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-queue
```

The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-queue

```

(continues on next page)

(continued from previous page)

```
{
    "objectType": "application/cdmi-queue",
    "objectID": "00007E7F00104BE66AB53A9572F9F51E",
    "objectName": "MyQueue",
    "parentURI": "/MyContainer/",
    "parentID" : "00007ED900104F67307652BAC9A37C93",
    "domainURI": "/cdmi_domains/MyDomain/",
    "capabilitiesURI": "/cdmi_capabilities/queue/",
    "completionStatus": "Complete",
    "metadata": {},
    "queueValues": "1-1",
    "mimetype": [
        "text/plain"
    ],
    "valuerange": [
        "0-19"
    ],
    "valuetransferencoding": [
        "utf-8"
    ],
    "value": [
        "First Enqueued Value"
    ]
}
```

2612 EXAMPLE 2: GET to the queue object URI to read the value and queue items of the queue object:

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

2613 The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-queue

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

2614 EXAMPLE 3: GET to the queue object URI to read the first five bytes of the value of the queue object:

```
GET /MyContainer/MyQueue?value:0-4 HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-queue
```

2615 The following shows the response:

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-queue

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

2616 EXAMPLE 4: GET to the queue object URI to read two values of the queue object:

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

2617 The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-queue

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

2618 EXAMPLE 5: GET to the queue object URI to read the queue object using multi-part MIME:

```
GET /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Accept: multipart/mixed
```

2619 The following shows the response.

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

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

{
    "objectType": "application/cdmi-queue",
    "objectID": "00007ED9001035E14BD1BA70C2EE98FC",
    "objectName": "MyQueue",
    "parentURI": "/MyContainer/",
    "parentID" : " 00007ED90010C2414303B5C6D4F83170",
    "domainURI": "/cdmi_domains/MyDomain/",
    "capabilitiesURI": "/cdmi_capabilities/queue/",
    "completionStatus": "Complete",
    "metadata": {
        ...
},
    "queueValues": "1-2",
    "mimetype": [
        "application/octet-stream",
        "application/octet-stream"
    ],
    "valuerange": [
        "0-19",
        "0-36"
    ],
    "valuetransferencoding": [
        "base64",
        "base64"
    ]
}

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

(continues on next page)

(continued from previous page)

```
<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>`
- `PUT <root URI>/<ContainerName>/<QueueName>?metadata:<metadataname>;...`
- `PUT <root URI>/cdmi_objectid/<QueueObjectID>`
- `PUT <root URI>/cdmi_objectid/<QueueObjectID>?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.
- `<QueueObjectID>` is the ID of the queue object to be updated.

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 88

Table 88: Request Headers - Update A Queue Object Using CDMI

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

11.4.4 Request Message Body

The request message body fields for updating a queue object using CDMI are shown in Table 89.

Table 89: 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

Continued on next page

Table 89 – continued from previous page

Field Name	Type	Description	Requirement
domainURI	JSON String	URI of the owning domain • If different from the parent domain, the user shall have the "`cross-domain`" privilege (see `cdmi_member_privileges` in Table 63). • If not specified, the existing domain shall be preserved.	Optional
deserialize	JSON String	URI of a source serialized queue object that shall be copied into the existing destination queue object. • If the destination queue object URI and the source serialized queue object URI both do not specify individual fields, the destination queue object shall be replaced with the contents of the serialized source queue object, with the exception that the destination queue values shall be preserved. See Section 11.6 to deserailize enqueued items. • If the destination queue object URI or the source serialized 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 source serialized 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 serialized queue object at the source URI or update the queue object at the destination URI, or if the read operation fails, the update shall return an HTTP status code of `400 Bad Request`, and the destination queue object shall not be updated.	Optional[2]

Continued on next page

Table 89 – continued from previous page

Field Name	Type	Description	Requirement
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 Section 11.6 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 update 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 data object serialized as specified in **RFC 4648** that shall be copied into the existing destination queue object. • If the destination queue object URI does not specify individual fields, the destination queue object shall be replaced with the contents of the serialized source queue object, with the exception that the destination queue values shall be preserved. See Section 11.6 to deserailize enqueued items. • If the destination 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 there are insufficient permissions update the queue object at the destination URI, the update shall return an HTTP status code of `400 Bad Request`, and the destination queue object shall not be updated.	Optional[2]

11.4.5 Response Header

The HTTP response header for updating a CDMI queue object using CDMI is shown in Table 90

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

Table 90: Response Header - Update A Queue Object Using CDMI

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

11.4.6 Response Message Body

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

11.4.7 Response Status

Table 91 describes the HTTP status codes that occur when updating a queue object using CDMI.

Table 91: 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

EXAMPLE 1: PUT to the queue object URI to set new metadata:

```
PUT /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-queue

{
    "metadata" : {

    }
}
```

The following shows the response.

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

EXAMPLE 2: PUT to the queue object URI to move six queue values from another queue:

```
PUT /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-queue

{
    "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>`
- `DELETE <root URI>/cdmi_objectid/<QueueObjectID>`

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.
- `<QueueObjectID>` is the ID of the queue object to be deleted.

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

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

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

Table 92: 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

EXAMPLE 1: DELETE to the queue object URI:

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

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

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.
- `<QueueObjectID>` is the ID of the queue object to be enqueued into.

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 93

Table 93: 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

11.6.4 Request Message Body

The request message body fields for enqueuing a new queue object value using CDMI are shown in Table 94.

Table 94: 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 multi-part MIME is being used, the value of the "`Content-Type`" header of the corresponding MIME part shall be assigned as the field value. • The same number of array elements shall be present as is present in the value field, and the mimetype field shall be associated with the value in the corresponding position. • This mimetype field value shall be converted to lower case before being stored.	Optional
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 enqueued. • If a move source object URI to a data object is provided, the value, mimetype, and valuetransferencoding field values from the source data object are used to enqueue the new item into the destination queue object, and the source data object is atomically deleted. • If a move source object URI to a queue object is provided, the corresponding value, mimetype, and valuetransferencoding field values of the specified number of enqueued items in the source queue object are transferred to the destination queue object and atomically removed from the source queue object.	Optional[3]

Continued on next page

Table 94 – continued from previous page

Field Name	Type	Description	Requirement
valuetransferencoding	JSON Array of JSON Strings	The value transfer encoding used for the queue object value. Two value transfer encodings are defined: • "`utf-8`" indicates that the queue object value contains a valid UTF-8 string, and shall be transported as a UTF-8 string in the value field. • "`base64`" indicates that the queue object value may contain arbitrary binary sequences, and shall be transported as a base 64 encoded string in the value field. Setting the contents of the queue object value field to any value other than a valid base 64 string shall result in an HTTP status code of `400 Bad Request` being returned to the client. • "`json`" indicates that the queue object value contains a valid JSON object, and the value field shall contain a JSON object. Setting the contents of the queue object value field to any value other than a valid JSOM object shall result in an HTTP status code of `400 Bad Request` being returned to the client. • If this field is not included and multi-part MIME is not being used, the value of "`utf-8`" shall be assigned as the field value. • If this field is not included and multi-part MIME is being used, the value of "`utf-8`" shall be assigned as the field value if the "`Content-Type`" header of the corresponding MIME part includes the charset parameter as defined in RFC 2046 of "`utf-8`" (e.g., "`;charset=utf-8`"). Otherwise, the value of "`base64`" shall be assigned as the field value. This field applies only to the encoding of the value when represented in JSON; the "`Content-Transfer-Encoding`" header of the part specifies the encoding of the value within a multi-part MIME request, as defined in **RFC 2045**. • This field shall be stored as part of the object.	Optional

Continued on next page

Table 94 – continued from previous page

Field Name	Type	Description	Requirement
value	JSON Array of JSON Strings	Data to be enqueued into the queue object. • If this field is not included and multi-part MIME is being used, the contents of the MIME parts shall be assigned as the field value. • If the corresponding valuetransferencoding field indicates UTF-8 encoding, the value shall be a UTF-8 string escaped using the JSON escaping rules described in **RFC 4627**. • If the corresponding valuetransferencoding field indicates base 64 encoding, the value shall be first encoded using the base 64 encoding rules as described in **RFC 4648**. • If the corresponding valuetransferencoding field indicates JSON encoding, the value shall contain a JSON object.	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

Table 95 describes the HTTP status codes that occur when enqueuing a new queue object using CDMI.

Table 95: 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

EXAMPLE 1: POST to the queue object URI a new value:

[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`.

```
POST /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-queue

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

2723 The following shows the response.

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

2724 EXAMPLE 2: POST to the queue object URI to copy an existing value:

```
POST /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-object

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

2725 The following shows the response.

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

2726 EXAMPLE 3: POST to the queue object URI to transfer 20 values from another queue object:

```
POST /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-object

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

2727 The following shows the response.

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

2728 EXAMPLE 4: POST to the queue object URI two new values:

```
POST /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-object

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

2729 The following shows the response.

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

EXAMPLE 5: POST to the queue object URI two new values, one with base 64 transfer encoding and one with utf-8 transfer encoding:

```
POST /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-object

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

The following shows the response.

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

EXAMPLE 6: POST to the queue object URI the binary contents of two new values using multi-part MIME:

```
POST /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Content-Type: multipart/mixed; boundary=gc0p4Jq0M2Yt08j34c0p

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

{}

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

<20 bytes of binary data>

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

<37 bytes of binary data>

--gc0p4Jq0M2Yt08j34c0p--
```

The following shows the response.

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

EXAMPLE 7: POST to the queue object URI the mime types and binary contents of two new values using multi-part MIME:

```
POST /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Content-Type: multipart/mixed; boundary=gc0p4Jq0M2Yt08j34c0p

--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--
```

2737 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>`
- `DELETE <root URI>/cdmi_objectid/<QueueObjectID>?value`
- `DELETE <root URI>/cdmi_objectid/<QueueObjectID>?values:<count>`
- `DELETE <root URI>/cdmi_objectid/<QueueObjectID>?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 from.
- `<QueueObjectID>` is the ID of the queue object to be deleted from.
- `<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 "?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

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

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

Table 96: 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

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
```

The following shows the response.

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

EXAMPLE 2: DELETE to the queue object URI value to remove the ten oldest values:

```
DELETE /MyContainer/MyQueue?values:10 HTTP/1.1
Host: cloud.example.com
```

The following shows the response.

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

EXAMPLE 3: DELETE to the queue object URI value to remove queue values 10 through 19:

```
DELETE /MyContainer/MyQueue?values:10-19 HTTP/1.1
Host: cloud.example.com
```

The following shows the response.

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

Clause 12

Capability Object Resource Operations using CDMI

12.1 Overview

Capability objects indicate what specific functionality and operations are supported by a given CDMI server, and allow CDMI clients to discover what subset of this international standard is implemented.

All CDMI servers shall support capabilities and the ability for CDMI clients to read capabilities.

Each CDMI capability object is represented as a JSON object, containing one or more "fields". For example, the "`capabilities`" field contains specific capability items.

EXAMPLE 1: CDMI Capability Object

```
{
  "objectType": "application/cdmi-capability",
  "objectID": "00007E7F00104BE66AB53A9572F9F51E",
  "objectName": "cdmi_capabilities/",
  "parentURI": "/",
  "parentID": "00007E7F0010128E42D87EE34F5A6560",
  "capabilities": {
      "cdmi_domains": "true",
      "cdmi_export_nfs": "true",
      "cdmi_export_iscsi": "true",
      "cdmi_queues": "true",
      "cdmi_notification": "true",
      "cdmi_query": "true",
      "cdmi_metadata_maxsize": "4096",
      "cdmi_metadata_maxitems": "1024"
  },
  "childrenrange": "0-3",
  "children": [
    "domain/",
    "container/",
    "dataobject/",
    "queue/"
  ]
}
```

The meaning, use, and permitted values of each field is described in Section 12.2.

12.1.1 Capability Object Addressing

Capability objects are addressed in CDMI in two ways:

- by name (e.g. `http://cloud.example.com/cdmi_capabilities/`); and
- by ID (e.g. `http://cloud.example.com/cdmi_objectid/00007E7F00104BE66AB53A9572F9F51E/`).

Every capability object has a single, globally-unique object ID that remains constant for the life of the object. Each capability object may also have one or more URI addresses that allow the capability object to be accessed.

When a capability object is addressed via more than one unique URIs, all operations may be performed through any of these URIs. For example, a capability object may be accessible via multiple virtual hosting paths, where `http://cloud.example.com/users/snia/cdmi/cdmi_capabilities/` is also accessible through `http://snia.example.com/cdmi/cdmi_cabilities/`.

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

If a request is performed against an existing capability 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.

Capabiltiies may also be nested.

EXAMPLE 2: The following URI represents a nested capability:

`http://cloud.example.com/cdmi_capabilities/container/`

A nested capability has a parent capability object, and shall be included in the children field of the parent capability object.

12.1.2 Capability Object Fields

Every CDMI object (excluding capability objects) includes a server-generated "`capabilitiesURI`" field that contains the URI of the capabilities object that describes which operations are permitted for that CDMI object.

Fig. 6 (shown on the next page) shows the hierarchy of capabilities and shows how the capabilitiesURI links data objects and container objects into the capabilities tree.

System-wide capabilities are described by the root capabilities object, which is accessible at "`<root URI>/cdmi_capabilities/`".

Capabilities cannot be altered by clients, but may be changed by the CDMI server to reflect configuration changes or operational changes. For example, if a CDMI server is upgraded or reconfigured, additional capabilities may become present, or existing capabilties may no longer be present. In practice, capabilities rarely change, and a client can assume that they shall remain constant for the duration of a client-server HTTP/HTTPS session.

Cloud clients should 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.

The capabilities defined as part of this international standard are described starting in Section 12.1.7. Vendor-defined capabilities not specified in this international standard shall not start with "`cdmi_`".

12.1.3 Capability Object Metadata

Capability objects do not have metadata.

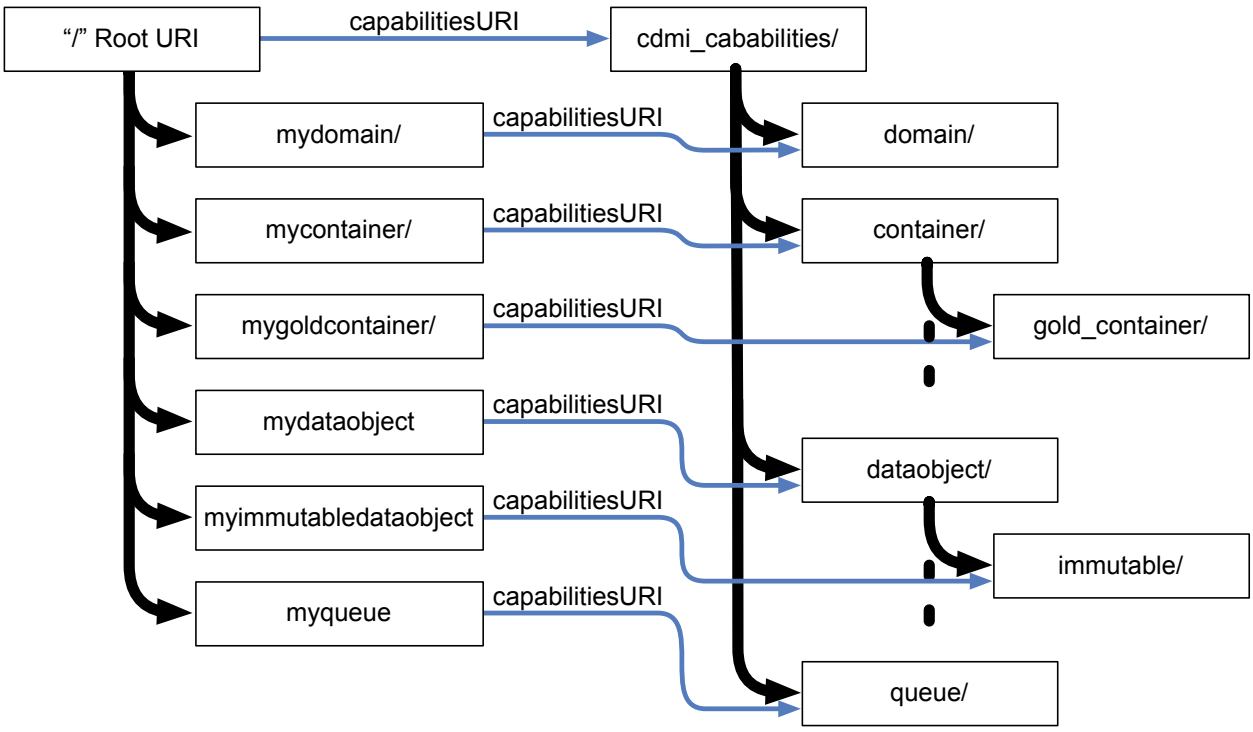


Fig. 6: Hierarchy of Capabilities

12.1.4 Capability Object Access Control

Capability objects are not subject to CDMI ACLs. Any authenticated CDMI client shall be capable of reading all Capability objects[1] .

Capabilities may differ from the operations permitted by an Access Control List (ACL) (see Section 17.1) associated with a given object. For example, a read-only cloud may not permit write access to a container or object, despite the presence of an ACL allowing write access.

12.1.5 Queue Object Consistency

Capabilitiy objects are read-only.

12.1.6 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.

[1] A CDMI Server may filter the visibility of capability objects and/or capability items for security purposes, for example, to prevent the client discovery of the names and characteristics of classification levels above the client's maximum classification level. Such filtering is out of scope of this international standard.

12.1.7 Cloud Storage System-Wide Capabilities

Table 97 defines the system-wide capabilities in a cloud storage system. These capabilities, which are found in the capabilities object, are referred to by the root URI (root capabilities).

Table 97: System-Wide Capabilities

Capability Name	Type	Definition
cdmi_domains	JSON String	If present and "`true`", the CDMI server supports domains. If not present, the domainURI field shall not be present in response bodies and the "`cdmi_domains`" URI shall not be present.
cdmi_export_cifs	JSON String	If present and "`true`", the CDMI server supports CIFS exports.
cdmi_dataobjects	JSON String	If present and "`true`", the CDMI server supports data objects.
cdmi_export_iscsi	JSON String	If present and "`true`", the CDMI server supports iSCSI exports.
cdmi_export_nfs	JSON String	If present and "`true`", the CDMI server supports NFS protocol exports.
cdmi_export_occi_iscsi	JSON String	If present and "`true`", the CDMI server supports OCCI/iSCSI exports.
cdmi_export_webdav	JSON String	If present and "`true`", the CDMI server supports WebDAV exports.
cdmi_metadata_maxitems	JSON String	If present, this capability indicates the maximum number of user-defined metadata items supported per object. If not present, there is no limit placed on the number of user-defined metadata items.
cdmi_metadata_maxsize	JSON String	If present, this capability indicates the maximum size, in bytes, of each user-defined metadata item supported per object. If not present, there is no limit placed on the size of user-defined metadata items.
cdmi_metadata_maxtotalsize	JSON String	If present, this capability indicates the maximum size, in bytes, of user-defined metadata supported by the CDMI server. If not present, there is no limit placed on the size of user-defined metadata.
cdmi_notification	JSON String	If present and "`true`", the CDMI server supports notification queues.
cdmi_logging	JSON String	If present and "`true`", the CDMI server supports logging queues.
cdmi_query	JSON String	If present and "`true`", the CDMI server supports query queues.
cdmi_query_regex	JSON String	If present and "`true`", the CDMI server supports query with regular expressions.
cdmi_query_contains	JSON String	If present and "`true`", the CDMI server supports query with "contains" expressions.
cdmi_query_tags	JSON String	If present and "`true`", the CDMI server supports query with tag-matching expressions.
cdmi_query_value	JSON String	If present and "`true`", the CDMI server supports query of value fields.
cdmi_queues	JSON String	If present and "`true`", the CDMI server supports queue objects.
cdmi_security_access_control	JSON String	If present and "`true`", the CDMI server supports ACLs. See Section 12.1.9 for additional information.

Continued on next page

Table 97 – continued from previous page

Capability Name	Type	Definition
cdmi_security_audit	JSON String	If present and "`true`", the CDMI server supports audit logging. See `ref_security_logging` for additional information.
cdmi_security_data_integrity	JSON String	If present and "`true`", the CDMI server supports data integrity/authenticity. See Section 12.1.9 for additional information.
cdmi_security_encryption	JSON String	If present and "`true`", the CDMI server supports data at-rest encryption. See Section 12.1.9 for additional information.
cdmi_security_immutability	JSON String	If present and "`true`", the CDMI server supports data immutability/retentions. See Section 12.1.9 for additional information.
cdmi_security_sanitization	JSON String	If present and "`true`", the CDMI server supports data/media sanitization. See Section 12.1.9 for additional information.
cdmi_serialization_json	JSON String	If present and "`true`", the CDMI server supports JSON as a serialization format.
cdmi_snapshots	JSON String	If present and "`true`", the CDMI server supports snapshots.
cdmi_references	JSON String	If present and "`true`", the CDMI server supports references.
cdmi_object_move_from_local	JSON String	If present and "`true`", the CDMI server supports moving CDMI objects from URIs within the same storage system.
cdmi_object_move_from_remote	JSON String	If present and "`true`", the CDMI server supports moving CDMI objects from URIs within other CDMI storage systems.
cdmi_object_move_from_ID	JSON String	If present and "`true`", the CDMI server supports moving CDMI objects without a path from a `/cdmi_objectid/` URI within the same storage system. This effectively adds a path, allowing the object to be accessed by ID and by path.
cdmi_object_move_to_ID	JSON String	If present and "`true`", the CDMI server supports moving CDMI objects with a path to a `/cdmi_objectid/` URI within the same storage system. This effectively removes the path, leaving the object only accessible by ID.
cdmi_object_copy_from_local	JSON String	If present and "`true`", the CDMI server supports copying CDMI objects from URIs within the same storage system.
cdmi_object_copy_from_remote	JSON String	If present and "`true`", the CDMI server supports copying CDMI objects from URIs within other CDMI storage systems.
cdmi_object_access_by_ID	JSON String	If present and "`true`", the CDMI server supports accessing, updating, and deleting objects through `/cdmi_objectid/`.
cdmi_post_dataobject_by_ID	JSON String	If present and "`true`", the CDMI server supports adding a new data object by ID via POST to "`/cdmi_objectid/`".
cdmi_post_queue_by_ID	JSON String	If present and "`true`", the CDMI server supports adding a new queue object by ID via POST to "`/cdmi_objectid/`".
cdmi_deserialize_dataobject_by_ID	JSON String	If present and "`true`", the CDMI server supports deserializating serialized data objects when creating a new data object by ID via POST to "`/cdmi_objectid/`".

Continued on next page

Table 97 – continued from previous page

Capability Name	Type	Definition
cdmi_deserialize_queue_by_ID	JSON String	If present and "true", the CDMI server supports deserializating serialized queue objects when creating a new queue object by ID via POST to "/cdmi_objectid/".
cdmi_serialize_dataobject_to_ID	JSON String	If present and "true", the CDMI server supports serializing data objects when creating a new data object by ID via POST to "/cdmi_objectid/".
cdmi_serialize_domain_to_ID	JSON String	If present and "true", the CDMI server supports serializing domain objects when creating a new data object by ID via POST to "/cdmi_objectid/".
cdmi_serialize_container_to_ID	JSON String	If present and "true", the CDMI server supports serializing container objects when creating a new data object by ID via POST to "/cdmi_objectid/".
cdmi_serialize_queue_to_ID	JSON String	If present and "true", the CDMI server supports serializing queue objects when creating a new data object by ID via POST to "/cdmi_objectid/".
cdmi_copy_dataobject_by_ID	JSON String	If present and "true", the CDMI server supports copying an existing data object when creating a new data object by ID via POST to "/cdmi_objectid/".
cdmi_copy_queue_by_ID	JSON String	If present and "true", the CDMI server supports copying an existing queue object when creating a new queue object by ID via POST to "/cdmi_objectid/".
cdmi_create_reference_by_ID	JSON String	If present and "true", the CDMI server supports creating a new reference via POST to "/cdmi_objectid/".
cdmi_copy_dataobject_from_queue	JSON String	If present and "true", the CDMI server supports the ability to copy to a data object from a queue object.
cdmi_multipart_mime	JSON String	If present and "true", the CDMI server supports storing and retrieving the value of data and queue objects using multi-part MIME.
cdmi_create_value_range_by_ID	JSON String	If present and "true", the CDMI server supports a new data object's value to be created with byte ranges through "/cdmi_objectid/".
cdmi_dac	JSON String	If present and "true", the CDMI server supports delegated access control.
cdmi_dac_methods	JSON Array of JSON Strings	If present, this capability contains a list of URI schemes supported for DAC URIs, as specified in the IANA URI Schemes registry. The following schemes shall be supported: • "https" – mandatory for all DAC implementations The following schemes may be supported: • "http" – optional for DAC implementations • "mailto" – optional for DAC implementations
cdmi_enc_cms	JSON String	If present and "true", the CDMI server supports operations against the contents of CMS encrypted objects.
cdmi_enc_jwe	JSON String	If present and "true", the CDMI server supports operations against the contents of JWE encrypted objects.
cdmi_enc_inplace	JSON String	If present and "true", the CDMI server supports operations to encrypt and decrypt objects in place, including updates.

Continued on next page

Table 97 – continued from previous page

Capability Name	Type	Definition
cdmi_enc_access	JSON String	If present and "`true`", the CDMI server supports operations to decrypt objects on access.
cdmi_cms_encryption	JSON Array of JSON Strings	If present, this capability lists which CMS `ContentEncryptionAlgorithmIdentifier` encryption algorithms are supported for operations against the contents of CMS encrypted objects.
cdmi_cms_digest	JSON Array of JSON Strings	If present, this capability lists which CMS `MessageAuthenticationCodeAlgorithm` digest algorithms are supported for operations against the contents of CMS encrypted objects.
cdmi_cms_signature	JSON Array of JSON Strings	If present, this capability lists which CMS `SignatureAlgorithmIdentifier` signature algorithms are supported for operations against the contents of CMS encrypted objects.
cdmi_jwe_enc	JSON Array of JSON Strings	If present, this capability lists which JOSE "`enc`" encryption algorithms are supported for operations against the contents of JWE encrypted objects, as defined in **RFC 7518**.
cdmi_jwe_alg	JSON Array of JSON Strings	If present, this capability lists which JOSE "`alg`" encryption algorithms are supported for operations against the contents ofJWE encrypted objects, as defined in **RFC 7518**.
cdmi_jws_alg	JSON Array of JSON Strings	If present, this capability lists which JOSE "`alg`" encryption algorithms are supported for operations against the contents of JWS signatures, as defined in **RFC 7518**.
cdmi_valuetransferencoding_json	JSON String	If present and "`true`", the CDMI server supports JSON value transfer encodings.

12.1.8 Storage System Metadata Capabilities

Table 98 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 16.1 for a description of these storage system metadata items.

Table 98: Capabilities for Storage System Metadata

Capability Name	Type	Definition
cdmi_acl	JSON String	If present and "`true`", the CDMI server 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 Section 12.1.7 of Section 12.1.7 shall also be set to "`true`". If not present, there is no support for ACL-based access control.
cdmi_size	JSON String	If present and "`true`", the CDMI server shall generate a `cdmi_size` storage system metadata for each stored object.
cdmi_ctime	JSON String	If present and "`true`", the CDMI server shall generate a `cdmi_ctime` storage system metadata for each stored object.
cdmi_atime	JSON String	If present and "`true`", the CDMI server shall generate a `cdmi_atime` storage system metadata for each stored object.
cdmi_mtime	JSON String	If present and "`true`", the CDMI server shall generate a `cdmi_mtime` storage system metadata for each stored object.
cdmi_acount	JSON String	If present and "`true`", the CDMI server shall generate a `cdmi_acount` storage system metadata for each stored object.
cdmi_mcount	JSON String	If present and "`true`", the CDMI server shall generate a `cdmi_mcount` storage system metadata for each stored object.
cdmi_dac_uri	JSON String	If present and "`true`", the CDMI server supports delegated access control metadata.
cdmi_dac_certificate	JSON String	If present and "`true`", the CDMI server supports delegated access control metadata.
cdmi_enc_signature	JSON String	If present and "`true`", the CDMI server shall generate a `cdmi_signature` storage system metadata for each stored object when a corresponding `sign_id` data system metadata item is present.
cdmi_version_object	JSON String	If present and "`true`", the CDMI server shall generate a `cdmi_version_object` storage system metadata for each version-enabled data object and data object version.
cdmi_version_current	JSON String	If present and "`true`", the CDMI server shall generate a `cdmi_version_current` storage system metadata for each version-enabled data object and data object version.
cdmi_version_oldest	JSON Array of JSON Strings	If present and "`true`", the CDMI server shall generate a `cdmi_version_oldest` storage system metadata for each version-enabled data object and data object version.
cdmi_version_parent	JSON String	If present and "`true`", the CDMI server shall generate a `cdmi_version_parent` storage system metadata for each data object version that has a previous version.

Continued on next page

Table 98 – continued from previous page

Capability Name	Type	Definition
cdmi_version_children	JSON Array of JSON Strings	If present and "`true`", the CDMI server shall generate a `cdmi_version_children` storage system metadata for each data object version.

12.1.9 Data System Metadata Capabilities

Table 99 defines the capabilities that indicate which data system metadata items are interpreted 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 16.2 for a description of the meaning of the corresponding data system metadata items.

Table 99: Capabilities for Data System Metadata

Capability Name	Type	Definition
cdmi_assignedsize	JSON String	If present and "`true`", the CDMI server supports the `cdmi_assignedsize` data system metadata as defined in Section 16.2.
cdmi_data_redundancy	JSON String	If present, the CDMI server supports the `cdmi_data_redundancy` data system metadata as defined in Section 16.2. The value of the capability shall be set to a positive numeric string representing the maximum value that the server supports.
cdmi_data_dispersion	JSON String	If present and "`true`", the CDMI server supports the `cdmi_data_dispersion` data system metadata as defined in Section 16.2.
cdmi_data_retention	JSON String	If present and "`true`", the CDMI server supports both the `cdmi_retention_id` and `cdmi_retention_period` data system metadata as defined in Section 16.2.
cdmi_data_autodelete	JSON String	If present and "`true`", the CDMI server supports the `cdmi_data_autodelete` data system metadata as defined in Section 16.2.
cdmi_data_holds	JSON String	If present and "`true`", the CDMI server supports the `cdmi_hold_id` data system metadata as defined in Section 16.2. When a cloud storage system supports holds for the purpose of making data immutable, the system-wide capability of `cdmi_security_immutability` specified in Table 97 of Section 12.1.7 shall be present and set to "`true`".
cdmi_encryption	JSON Array of JSON Strings	If present, the CDMI server supports the `cdmi_encryption` data system metadata as defined in Section 16.2. When present, this capability shall contain one or more JSON strings, each string corresponding to an algorithm/mode/length value as described in the `cdmi_encryption` data system metadata section in Section 16.2. When a cloud storage system supports at-rest encryption, the system-wide capability of `cdmi_security_encryption` specified in Table 97 of Section 12.1.7 shall be present and set to "`true`".
cdmi_geographic_placement	JSON String	If present and "`true`", the CDMI server supports the `cdmi_geographic_placement` data system metadata as defined in Section 16.2.
cdmi_immediate_redundancy	JSON String	If present, the CDMI server supports the `cdmi_immediate_redundancy` data system metadata as defined in Section 16.2. When present, this capability shall contain a string set to a positive numeric string representing the maximum value that the server supports.

Continued on next page

Table 99 – continued from previous page

Capability Name	Type	Definition
cdmi_infrastructure_redundancy	JSON String	If present, the CDMI server supports the `cdmi_infrastructure_redundancy` data system metadata as defined in Section 16.2. When present, this capability shall contain a string set to a positive numeric string representing the maximum value that the server supports.
cdmi_latency	JSON String	If present and "`true`", the CDMI server supports the `cdmi_latency` data system metadata as defined in Section 16.2.
cdmi_RPO	JSON String	If present and "`true`", the CDMI server supports the cdmi_RPO data system metadata as defined in Section 16.2.
cdmi_RTO	JSON String	If present and "`true`", the CDMI server supports the `cdmi_RTO` data system metadata as defined in Section 16.2
cdmi_sanitization_method	JSON Array of JSON Strings	If present, the CDMI server supports the `cdmi_sanitization_method` data system metadata as defined in Section 16.2. When present, this capability shall contain one or more JSON strings, each string corresponding to a sanitization method as described in the `cdmi_sanitization_method` data system metadata section in Section 16.2. When a cloud storage system supports sanitization, the system-wide capability of `cdmi_security_sanitization` specified in Table 97 of Section 12.1.7 shall be present and set to "`true`".
cdmi_throughput	JSON String	If present and "`true`", the CDMI server supports the cdmi_throughput data system metadata as defined in Section 16.2.
cdmi_value_hash	JSON Array of JSON Strings	If present, the CDMI server supports the cdmi_value_hash data system metadata as defined in Section 16.2. When present, this capability shall contain one or more JSON strings, each string corresponding to an algorithm/length value as described in the `cdmi_value_hash` data system metadata section in Section 16.2. When a cloud storage system supports value hashing, the system-wide capability of `cdmi_security_data_integrity` specified in Table 97 of Section 12.1.7 shall be present and set to "`true`".
cdmi_enc_key_id	JSON String	When the cloud storage system supports the `cdmi_enc_key_id` data system metadata as defined in clause 16.2, the `cdmi_enc_key_id` 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_enc_key_id` data system metadata shall not be used.
cdmi_enc_value_sign_id	JSON String	When the cloud storage system supports the `cdmi_enc_value_sign_id` data system metadata as defined in clause 16.2, the `cdmi_enc_value_sign_id` 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_enc_value_sign_id` data system metadata shall not be used.

Continued on next page

Table 99 – continued from previous page

Capability Name	Type	Definition
cdmi_enc_value_verify_id	JSON String	When the cloud storage system supports the `cdmi_enc_value_verify_id` data system metadata as defined in clause 16.2, the `cdmi_enc_value_verify_id` 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_enc_value_verify_id` data system metadata shall not be used.
cdmi_enc_object_sign_id	JSON String	When the cloud storage system supports the `cdmi_enc_object_sign_id` data system metadata as defined in clause 16.2, the `cdmi_enc_object_sign_id` 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_enc_object_sign_id` data system metadata shall not be used.
cdmi_enc_object_verify_id	JSON String	When the cloud storage system supports the `cdmi_enc_object_verify_id` data system metadata as defined in clause 16.2, the `cdmi_enc_object_verify_id` 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_enc_object_verify_id` data system metadata shall not be used.
cdmi_versioning	JSON array of JSON strings	If present, this capability indicates that the cloud storage system shall support versioning of data objects and contains a list of which versioning behaviors are supported. The following values are defined: • "value" indicates that the system shall support the versioning of the object value. • "user" indicates that the system shall support the versioning of the object value and user metadata. • "all" indicates that the system shall support the versioning of all updates made to a data object. When present, the system shall support the following storage system metadata: `cdmi_version_object`, `cdmi_version_current`, `cdmi_version_oldest`, `cdmi_version_parent`, and `cdmi_version_children` as indicated by the corresponding storage system metadata capabilities.
cdmi_versions_count	JSON String	If present, this capability specifies the maximum number of historical versions that may be specified. If absent, restrictions on the number of historical versions specified shall be ignored.
cdmi_version_age	JSON String	If present, this capability specifies the maximum age of historical versions that may be specified. If absent, restrictions on the age of historical versions specified shall be ignored.
cdmi_versions_size	JSON String	If present, this capability specifies the maximum total size of historical versions that may be specified. If absent, restrictions on the size of historical versions specified shall be ignored.

12.1.10 Data Object Capabilities

Table 100 defines the capabilities for data objects in a cloud storage system.

Table 100: Capabilities for Data Objects

Capability Name	Type	Definition
cdmi_read_value	JSON String	If present and "`true`", this capability indicates that the CDMI server 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 CDMI server 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 CDMI server shall support the ability to read the object's metadata.
cdmi_modify_value	JSON String	If present and "`true`", this capability indicates that the CDMI server 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 CDMI server 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 CDMI server 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 CDMI server 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 CDMI server shall support the ability to delete the object.

12.1.11 Container Capabilities

Table 101 defines the capabilities for containers in a cloud storage system.

Table 101: Capabilities for Containers

Capability Name	Type	Definition
cdmi_list_children	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability to list the container's children.
cdmi_list_children_range	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability to list the container's children with ranges.
cdmi_read_metadata	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability to read the container's metadata.
cdmi_modify_metadata	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability to modify the container's metadata.
cdmi_modify_deserialize_container	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability of the container object to deserialize a serialized container object into the container object as an update.
cdmi_snapshot	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability of the container object to create a new snapshot.
cdmi_serialize_dataobject	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability to serialize a data object.
cdmi_serialize_container	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability to serialize the container and all children's contents.
cdmi_serialize_queue	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability to serialize a queue object.
cdmi_serialize_domain	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability to serialize the domain and all child domains.
cdmi_deserialize_container	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability of the container to deserialize the serialized containers and associated serialized children into the container.
cdmi_deserialize_queue	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability of the container to deserialize the serialized queue objects into the container.
cdmi_deserialize_dataobject	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability of the container to deserialize the serialized data objects into the container.
cdmi_create_dataobject	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability of the container to add a new data object.
cdmi_post_dataobject	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability of the container to add a new data object via POST.
cdmi_post_queue	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability of the container to add a new queue object via POST.

Continued on next page

Table 101 – continued from previous page

Capability Name	Type	Definition
cdmi_create_container	JSON String	If present and “`true`”, this capability indicates that the CDMI server shall support the ability to create a new container object via PUT.
cdmi_create_queue	JSON String	If present and “`true`”, this capability indicates that the CDMI server shall support the ability to create new queue objects..
cdmi_create_reference	JSON String	If present and “`true`”, this capability indicates that the CDMI server shall support the ability to create a new child reference via PUT.
cdmi_export_container_cifs	JSON String	If present and “`true`”, this capability indicates that the CDMI server shall support the ability to export a container as a file system via CIFS.
cdmi_export_container_nfs	JSON String	If present and “`true`”, this capability indicates that the CDMI server shall support the ability to export a container as a file system via NFS.
cdmi_export_container_iscsi	JSON String	If present and “`true`”, this capability indicates that the CDMI server shall support the ability to export a container as a file system via iSCSI.
cdmi_export_container_occi	JSON String	If present and “`true`”, this capability indicates that the CDMI server shall support the ability to export a container as a file system via OCCI.
cdmi_export_container_webdav	JSON String	If present and “`true`”, this capability indicates that the CDMI server shall support the ability to export a container as a file system via WebDAV.
cdmi_delete_container	JSON String	If present and “`true`”, this capability indicates that the CDMI server shall support the ability to delete a container.
cdmi_move_container	JSON String	If present and “`true`”, this capability indicates that the CDMI server shall support the ability to move a container object into a container.
cdmi_copy_container	JSON String	If present and “`true`”, this capability indicates that the CDMI server shall support the ability to copy a container object into a container.
cdmi_move_dataobject	JSON String	If present and “`true`”, this capability indicates that the CDMI server shall support the ability to move a data object into a container.
cdmi_copy_dataobject	JSON String	If present and “`true`”, this capability indicates that the CDMI server shall support the ability to copy a data object into a container.
cdmi_create_value_range	JSON String	If present and “`true`”, this capability indicates that the container allows a new data object’s value to be created with byte ranges.

12.1.12 Domain Object Capabilities

Table 102 defines the capabilities for domains in a cloud storage system. (All capabilities refer to what may be done via CDMI content-type operations.

Table 102: Capabilities for Domain Objects

Capability Name	Type	Definition
cdmi_create_domain	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability to add a new subdomain.
cdmi_delete_domain	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability to delete a domain.
cdmi_move_domain	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability to move a domain.
cdmi_domain_summary	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability to support domain summaries.
cdmi_domain_members	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability to support domain user management.
cdmi_list_children	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability to list the domain's children.
cdmi_read_metadata	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability to read the domain's metadata.
cdmi_modify_metadata	JSON String	If present and "`true`", this capability indicates that the CDMI server 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 CDMI server 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 CDMI server 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 CDMI server shall support the ability to deserialize serialized domains and associated serialized children into the domain.

Continued on next page

Table 102 – continued from previous page

Capability Name	Type	Definition
cdmi_authentication_methods	JSON Array of JSON Strings	If present, the CDMI server supports authentication methods that are supported by a domain. When present, this capability shall contain one or more of the following JSON strings: • “anonymous” - Absence of authentication supported • “basic” - HTTP basic authentication supported (**RFC 2617**) • “digest” - HTTP digest authentication supported (**RFC 2617**) • “krb5” - Kerberos authentication supported, using the Kerberos domain specified in the CDMI domain (**RFC 4559**) • “x509” - certificate-based authentication via TLS (**RFC 5246**) • “s3” - S3 API signed header authentication supported • “openstack” - OpenStack Identity API header authentication supported Interoperability with these authentication methods are not defined by this international standard. Servers may include other authentication methods not included in the above list. In these cases, it is up to the CDMI client and CDMI server to ensure interoperability.

12.1.13 Queue Object Capabilities

Table 103 defines the capabilities for queue objects in a cloud storage system.

Table 103: Capabilities for Queue Objects

Capability Name	Type	Definition
cdmi_read_value	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability to read a queue's value.
cdmi_read_metadata	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability to read the queue's metadata.
cdmi_modify_value	JSON String	If present and "`true`", this capability indicates that the CDMI server shall support the ability to modify the queue's value.
cdmi_modify_metadata	JSON String	If present and "`true`", this capability indicates that the CDMI server 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 CDMI server 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 CDMI server shall support the ability to delete a queue.
cdmi_move_queue	JSON String	If present and "`true`", this capability indicates that the CDMI server 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 CDMI server 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 CDMI server shall support the ability to reference a queue from another queue.

12.2 Read a Capabilities Object using CDMI

12.2.1 Synopsis

To read an existing capability object, the following requests shall be performed:

- `GET <root URI>/cdmi_capabilities/<Capability>/<TheCapability>/`
- `GET <root URI>/cdmi_capabilities/<Capability>/<TheCapability>/?<fieldname>;<fieldname>;...`
- `GET <root URI>/cdmi_capabilities/<Capability>/<TheCapability>/?children:<range>;...`
- `GET <root URI>/cdmi_objectid/<CapabilityObjectID>/`
- `GET <root URI>/cdmi_objectid/<CapabilityObjectID>/?<fieldname>;<fieldname>;...`
- `GET <root URI>/cdmi_objectid/<CapabilityObjectID>/?children:<range>;...`

Where:

- `<root URI>` is the path to the CDMI cloud.
- `<Capability>` is zero or more parent capabilities.
- `<TheCapability>` is the name specified for the capability 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.
- `<CapabilityObjectID>` is the ID of the capability object to be read from.

12.2.2 Capabilities

No capabilities are associated with reading a capability object.

12.2.3 Request Headers

The HTTP request headers for reading a CDMI capabilities object using CDMI are shown in Table 104.

Table 104: 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

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 105.

Table 105: Response Headers - Read a Capabilities Object Using CDMI

Header	Type	Description	Requirement
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 106.

Table 106: 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 Appending the "`objectName`" to the "`parentURI`" shall always produce a valid URI for the object.	Mandatory
parentID	JSON String	Object ID of the parent capability 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.	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 CDMI server.	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 107 describes the HTTP status codes that occur when reading a capabilities object using CDMI.

Table 107: HTTP Status Codes Read a Capabilities Object using CDMI

HTTP Status	Description
`200 OK`	The capabilities object content was returned in the response.
`400 Bad Request`	The request contains invalid parameters or field names.
`401 Unauthorized`	The authentication credentials are missing or invalid.
`403 Forbidden`	The client lacks the proper authorization to perform this request.
`404 Not Found`	The resource was not found at the specified URI.
`406 Not Acceptable`	The server is unable to provide the object in the content type specified in the Accept header.

12.2.8 Examples

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

```
GET /cdmi_capabilities/ HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-capability
```

The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-capability

{
    "objectType": "application/cdmi-capability",
    "objectID": "00007E7F00104BE66AB53A9572F9F51E",
    "objectName": "cdmi_capabilities/",
    "parentURI": "/",
    "parentID": "00007E7F0010128E42D87EE34F5A6560",
    "capabilities": {
        "cdmi_domains": "true",
        "cdmi_export_nfs": "true",
        "cdmi_export_iscsi": "true",
        "cdmi_queues": "true",
        "cdmi_notification": "true",
        "cdmi_query": "true",
        "cdmi_metadata_maxsize": "4096",
        "cdmi_metadata_maxitems": "1024"
    },
    "childrenrange": "0-3",
    "children": [
        "domain/",
        "container/",
        "dataobject/",
        "queue/"
    ]
}
```

EXAMPLE 2: GET to the root container capabilities URI to read the capabilities and children of the container:

```
GET /cdmi_capabilities/?capabilities;children HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-capability
```

The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-capability

{
    "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/"
    ]
}
```

2909 EXAMPLE 3: GET to the root container capabilities URI to read the first two children contained within a domain:

```
GET /cdmi_capabilities/?childrenrange;children:0-1 HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-capability
```

2910 The following shows the response.

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-capability

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

Clause 13

Exported Protocols

13.1 Overview

Container objects can be exported via multiple storage protocols. This is specified by adding an export field to the container object. Each container object export field has a set of well-defined items that include:

- zero or more export types, which specifies the export protocol;
- for each export type, the identity of the container as standardized by the export protocol;
- for each export type, the domain of the protocol name server for the clients being served;
- for each export type, 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;
- for each export type, required protocol-specific export parameters;
- for each export type, optional protocol-specific export parameters; and
- for each export type, export control parameters.

All cloud storage systems shall support the storage of the export field. The ability to export containers via a specified protocol is determined by the presence or absence of a `cdmi_export_<protocol>` system wide capabilities, which are listed in `ref_cloud_storage_system`. The ability to export a specific container via a specified protocol is indicated by the `cdmi_export_<protocol>` capability in the specified container.

Container object exports are represented as a JSON object, containing zero or more named protocol exports.

EXAMPLE 1: CDMI Container Exports

```
{
    "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" : {
```

(continues on next page)

(continued from previous page)

```
            "hosts" : "*",
            "domain" : "lab.mycollege.edu",
            "usermap" : {
                { "jimsmith", "<-->", "james.smith" }
                { [ordered list of CDMIname/operator/NFSname triples] },
                { "*", "<-->", "*" }
            }
            "groupmap" : {
                { "admins", "<-", "Administrators" },
                { "everyone", "<-", "*" }
            }
        }
    }
}
```

The meaning, use, and permitted values for the fields associated with each export type are described later in this section.

13.1.1 Container Object Exports Addressing

Container object exports are addressed in CDMI in two ways:

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

See Section 9.1 for more detials on container object addressing.

13.1.2 Container Object Exports Fields

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.

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 an external domain is used, or 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.

13.1.3 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 Section 13.1.8 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.

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.

13.1.4 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.

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.

EXAMPLE 2: NFS Export User Mapping

```
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

{
    "exports": {
        "nfs": {
            "usermap": [
                [
                    "nobody",
                    "<-",
                    "root"
                ],
                [
                    "*",
                    "<-->",
```

(continues on next page)

(continued from previous page)

```
                    "*"
                ]
            ]
        }
    }
}
```

13.1.5 Mapping Domains from CDMI to Another Protocol

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.

13.1.6 User and Domain Mapping 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.

13.1.7 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 Section 13.1.8.

13.1.8 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@`".

13.2 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 108.

Table 108: Required members of the NFS protocol structure

Member	Description
protocol	The protocol being requested. This value shall be "`NFSv3`", "`NFSv4`", "`NFSv4.1`", or any subsequent NFS version enshrined in a major IETF RFC. Version 2 of NFS is not supported by CDMI.
exportpath	The pathname to which the export should be surfaced. This value shall be a UTF8 string of the form `[<server>]:/<path>`, where the <server> component is optional, (e.g., "`eeserver:/lessons/number1`"). The <server> component of the path must be obtained from an administrator of the service running the CDMI implementation.
exportdomain	The internet domain of the protocol name server for the clients being served. This value is normally the name of the LDAP domain for the organization, e.g., "`iti.edu`". A value of "`.`" shall be interpreted to be the DNS name of the domain occupied by the CDMI server.
mode	This value shall be "`ro`", "`rw`", "`root`" or "`rpc_gsssec`" and becomes the default export mode. Hosts requiring different access shall be specified in the optional "`rw_mode`", "`ro_mode`", and "`root_mode`" structure members. However, the "`rpc_gsssec`" mode overrides all other modes, and all other mode members and their contents shall be ignored if it is specified.
control	Export control for the container. This value shall be "`immediate`", "`off`", "`on`", or <n> (a number). Servers may set the value to on, but clients shall not. A numeric value (<n>) indicates that the export should be shut down in <n> seconds, possibly after a message has been sent to clients mounting the export. If a client specifies a value for <n> but the server does not support delayed shutdown of exports, then <n> shall be interpreted to mean off.

Optional export parameters for NFS are described in Table 109.

Table 109: Optional members of the NFS protocol structure

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.

EXAMPLE 3: NFS Export Options

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

13.2.1 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.

13.3 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 110

Table 110: Required Members of the CIFS protocol structure

Member	Description
share_name	The name that CIFS shall use to discover the share.
exportdomain	The domain of the protocol name server for the clients being served. This value is normally the name of the Active Directory LDAP domain for the organization, e.g. "`iti.edu`". A value of "`.`" shall be interpreted to be the domain occupied by the CDMI server.
mode	This value shall be either "`ro`" or "`rw`".
control	Export control for the container. This value shall be "`immediate`", "`off`", or <n> (a number). Servers may set the value to on, but clients shall not. The semantics and normative requirements are exactly the same as for NFS, as documented in the paragraph "`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.

EXAMPLE 4: CIFS Export

```
"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.

13.4 OCCI Exported Protocol

Container objects can be exported via multiple protocols. This is especially useful when CDMI is being used as a storage interface in a cloud computing environemnt, as illustrated in Fig. 7 below.

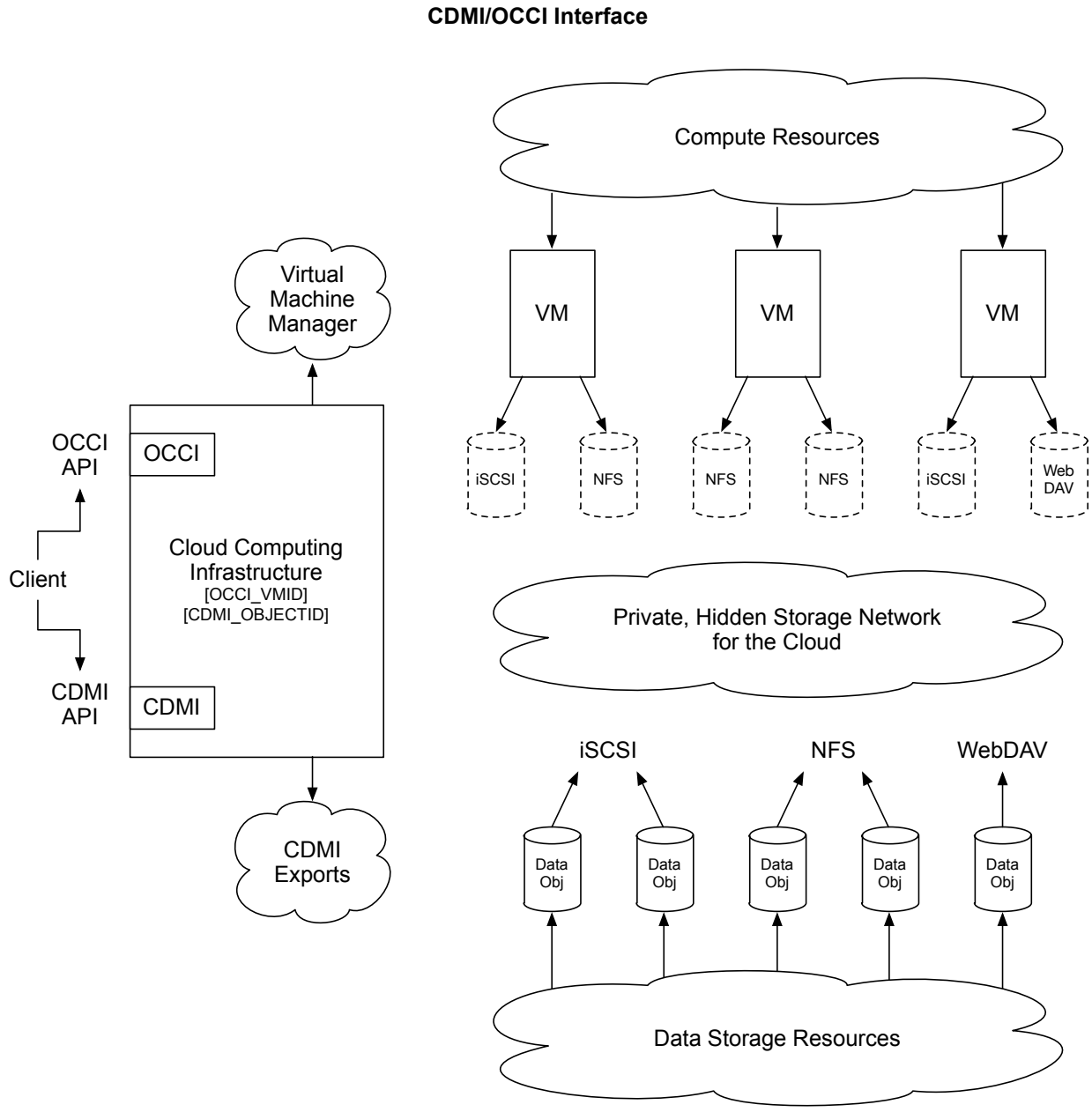


Fig. 7: CDMI and OCCI in an Integrated Cloud Computing Environment

In this example, CDMI containers may also be used as virtual disks by virtual machines in the cloud computing environemnt. 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.

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.

EXAMPLE 5: OCCI Export

```
"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.

13.5 iSCSI Exported Protocol

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.

13.5.1 Read Container

All of the information in the export structure is returned:

EXAMPLE 6: iSCSI Export

```
"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"
        ]
    }
}
```

13.5.2 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.

EXAMPLE 7: iSCSI Export Creation

```
"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.

13.5.3 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.

EXAMPLE 8: iSCSI Export Modification

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

13.6 WebDAV Exported Protocol

CDMI defines an export protocol structure for the WebDAV standard as follows (see **RFC 4918**):

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

EXAMPLE 9: iSCSI Export

```
"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

14.1 Overview

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. 8).

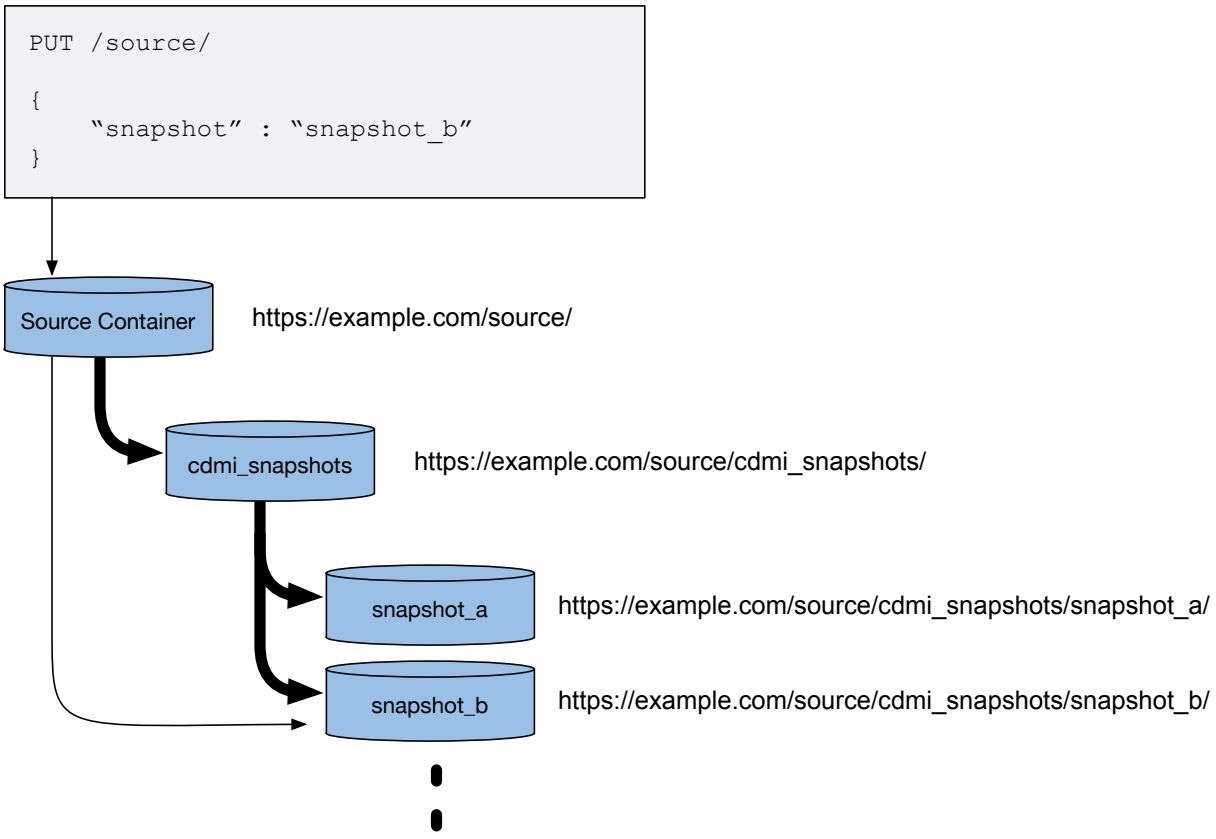


Fig. 8: Snapshot Container Structure

14.2 Creating a Snapshot

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

Bulk data movement is often 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 data and storage system metadata is inherited from enclosing container objects, all parent metadata shall be represented in the top-level of the canonical format. 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

EXAMPLE 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": "/",
```

(continues on next page)

(continued from previous page)

```
    "parentID": "00007E7F0010128E42D87EE34F5A6560",
    "domainURI": "/cdmi_domains/MyDomain/",
    "capabilitiesURI": "/cdmi_capabilities/container/",
    "completionStatus": "Complete",
    "metadata": {
                ...
        },
    "exports": {
        "OCCI/iSCSI": {
            "identifier": "00007E7F00104BE66AB53A9572F9F51E",
            "permissions": [
                "http://example.com/compute/0/",
                "http://example.com/compute/1/"
            ]
        },
        "Network/NFSv4": {
            "identifier": "/users",
            "permissions": "domain"
        }
    },
    "childrenrange": "0-1",
    "children": [
        {
            "objectType": "application/cdmi-object",
            "objectID": "00007ED900104F67307652BAC9A37C93",
            "objectName": "MyDataObject.txt",
            "parentURI": "/MyContainer/",
            "parentID": "00007E7F00102E230ED82694DAA975D2",
            "domainURI": "/cdmi_domains/MyDomain/",
            "capabilitiesURI": "/cdmi_capabilities/dataobject/",
            "completionStatus": "Complete",
            "mimetype": "text/plain",
            "metadata": {
                                ...
                            },
            "valuerange": "0-36",
            "valuetransferencoding": "utf-8",
            "value": "This is the Value of this Data Object"
        },
        {
            "objectType": "application/cdmi-queue",
            "objectID": "00007E7F00104BE66AB53A9572F9F51E",
            "objectName": "MyQueue",
            "parentURI": "/MyContainer/",
            "parentID": "00007E7F00102E230ED82694DAA975D2",
            "domainURI": "/cdmi_domains/MyDomain/",
            "capabilitiesURI": "/cdmi_capabilities/queue/",
            "completionStatus": "Complete",
            "metadata": {
                                ...
                            },
            "queueValues": "0-1",
            "mimetype": [
                "text/plain",
                "text/plain"
            ],
            "valuetransferencoding": [
                "utf-8",
                "utf-8"
            ],
            "valuerange": [
                "0-2",
                "0-3"
            ],
            "value": [
                "red",
                "blue"
            ]
```

(continues on next page)

(continued from previous page)

```
            }
        ]
}
```

To allow efficient deserialization in stream mode when serializing container objects to JSON, data object `value` fields and container `children` arrays should be the last items in the canonical serialized JSON format.

Clause 16

Metadata

16.1 Support for Storage System Metadata

After an object has been created or updated, the storage system metadata, as described in Table 111, shall be generated or updated 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.

Which storage system metadata is supported by the CDMI server defined in Section 12.1.8. Storage system metadata that is not supported by the CDMI server shall be preserved.

Table 111: Storage System Metadata

Metadata Name	Type	Description	Requirement
cdmi_size	JSON String	The number of bytes consumed by the object. This storage system metadata item is computed by the storage system, and any attempts to set or modify it will be ignored.	Optional
cdmi_ctime	JSON String	The time when the object was created, in ISO-8601 point-in-time format, as described in Section 5.6. 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_atime	JSON String	The time when the object was last accessed in ISO-8601 point-in-time format, as described in Section 5.6. 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

Continued on next page

Table 111 – continued from previous page

Metadata Name	Type	Description	Requirement
cdmi_mtime	JSON String	The time when the object was last modified, in ISO-8601 point-in-time format, as described in Section 5.6. 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. This storage system metadata item is computed by the storage system, and any attempts to set or modify it will be ignored.	Optional
cdmi_owner	JSON String	The name of the principal that has owner privileges for the object. If not specified when the object is created, this principal associated with the user creating the object shall be used. This metadata value can be updated by users with appropriate permissions.	Optional
cdmi_acl	JSON Array of JSON Objects	Standard ACL metadata as described in Section 17.1. If not specified when the object is created, the ACL metadata shall be generated in by the system. This metadata value can be updated by users with appropriate permissions.	Optional

Continued on next page

Table 111 – continued from previous page

Metadata Name	Type	Description	Requirement
cdmi_dac_uri	JSON String	Contains the URI used to submit a DAC request for the data object. URI schemes supported is defined in the `cdmi_dac_methods capability`. Both `cdmi_dac_certificate` and `cdmi_dac_uri` shall be included for delegated access control to be enabled for a given object.	Optional
cdmi_dac_certificate	JSON Object	A JSON object, containing a JWE JWK which shall include a public key that is used to submit a DAC request for the data object, and should contains a X.509 certificate or certificate chain used to verify the identity of the DAC provider. Both `cdmi_dac_certificate` and `cdmi_dac_uri` shall be included for delegated access control to be enabled for a given object.	Optional
cdmi_enc_signature	JSON Object	Contains JWS compact serialization of a signature for the entire object (value and metadata). See clause 23.7 for more details.	Optional

16.2 Support for Data System Metadata

When specified, data system metadata, as described in Table 112, provides guidelines to the cloud storage system on how to provide storage data services for data managed through the CDMI interface. Data system metadata is inherited from parent objects to any children. If a child explicitly contains data system metadata, the metadata value of the child data system metadata shall override the metadata value of the parent data system metadata.

Which data system metadata is supported by the CDMI server defined in Section 12.1.9. Data system metadata that is not supported by the CDMI server shall be preserved.

Table 112: 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 Section 9.1.3). The system is not required to reserve this space and may thin-provision the requested space. Thus, the requested value may be greater than the actual storage space consumed. 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. This data system metadata item is only applied against container objects and is not inherited by child objects.	Optional
cdmi_infrastructure_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 independent storage infrastructures supporting the multiple copies of data. This metadata is used to convey that, of the copies specified in `cdmi_data_redundancy`, these copies shall be stored on this many separate infrastructures. 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

Continued on next page

Table 112 – continued from previous page

Metadata Name	Type	Description	Requirement
cdmi_data_dispersion	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 minimum desired distance (in km) between the infrastructures supporting the multiple copies of data. This metadata is used to separate the (`cdmi_infrastructure_redundancy` number of) infrastructures by a minimum geographic distance to prevent data loss due to site disasters. 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_geographic_placement	JSON Array of JSON Strings	If this data system metadata item is present and set to zero or more geopolitical identifiers, it indicates that the client is requesting restrictions on the geographic regions where the object is permitted to be stored. Each geopolitical identifier shall be in the form of either a string containing a valid ISO 3166 country/country-subdivision code, which indicates that storage is permitted within that geopolitical region, or in the form of a string starting with the "`!`" character in front of a valid ISO 3166 country/country-subdivision code, which excludes that country/country-subdivision from the previous list of geopolitical regions. The list is evaluated, in order, from left to right, with evaluation of each candidate storage location stopping when the candidate location is a permitted or prohibited region or is contained within a permitted or prohibited region. In addition to the ISO 3166 codes, "`*`" shall indicate all regions. If a candidate location does not match any of the entries in the list, the candidate location shall be considered to be prohibited. • When this data system metadata item is absent, this data system metadata item shall not be used. • When this data system metadata item is present and does not contain valid geopolitical identifiers, the create, update, or deserialize operation shall fail with an HTTP status code of `400 Bad Request`. • When this data system metadata item is present and valid, but no available storage locations are permitted, the create, update, or deserialize operation shall fail with an HTTP status code of `403 Forbidden`.	Optional

Continued on next page

Table 112 – continued from previous page

Metadata Name	Type	Description	Requirement
cdmi_retention_id	JSON String	If this data system metadata item is present and not an empty string, it indicates that the client is requesting that the string be used to tag a given object as being managed by a specific retention policy. This data system metadata item is not required to place an object under retention, but is useful when needing to be able to perform a query to find all objects under a specific retention policy. When this data system metadata item is absent, or is present and an empty string, this data system metadata item shall not be used.	Optional
cdmi_retention_period	JSON String	If this data system metadata item is present and contains a valid ISO 8601:2004 time interval (as described in), it indicates that the client is requesting that an object be placed under retention (see Section 18.3). When this data system metadata item is absent, this data system metadata item shall not be used. When this data system metadata item is present but does not contain a valid ISO 8601:2004 time interval, the create, update, or deserialize operation shall fail with an HTTP status code of `400 Bad Request`. If this data system metadata item is updated and the new end date is before the current end date, the update operation shall fail with an HTTP status code of `403 Forbidden`.	Optional
cdmi_retention_autodelete	JSON String	If this data system metadata item is present and set to "`true`", it indicates that the client is requesting that an object under retention be automatically deleted when retention expires. 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.	Optional
cdmi_hold_id	JSON Array of JSON Strings	If this data system metadata item is present and not an empty array, it indicates that the client is requesting that an object be placed under hold (see Section 18.4). Each string in the array shall contain a unique user-specified hold identifier. When this data system metadata item is absent, or is present and is an empty JSON array, this data system metadata item shall not be used. If this data system metadata item is updated, and a previously existing hold string has been removed or changed in the update, the update operation shall fail with an HTTP status code of `403 Forbidden`. (See Section 18.4 concerning releasing holds.)	Optional

Continued on next page

Table 112 – continued from previous page

Metadata Name	Type	Description	Requirement
cdmi_encryption	JSON String	If this data system metadata item is present and not an empty string, it indicates that the client is requesting that the object be encrypted while at rest. If encrypted, all data and metadata related to the object shall be encrypted. Supported algorithm/mode/length values are provided by the `cdmi_encryption` capability. When this data system metadata item is absent, this data system metadata item shall not be used. If this data system metadata item is present but does not contain a valid encryption algorithm/mode/length string, the system is free to choose to ignore the data system metadata, to fail with an HTTP status code of `400 Bad Request`, or to select an encryption algorithm/mode/length of the system's choice. Supported encryption algorithms are expressed as a string in the form of ALGORITHM_MODE_KEYLENGTH, where: • "`ALGORITHM`" is the encryption algorithm (e.g., "`AES`" or "`3DES`"). • "`MODE`" is the mode of operation (e.g., "`XTS`", "`CBC`", or "`CTR`"). • "`KEYLENGTH`" is the key size in bytes (e.g., "`128`", "`192`", "`256`"). To improve interoperability between CDMI implementations, the following designators should be used for the more common encryption combinations: • "`3DES_ECB_168`" for the three-key TripleDES algorithm, the Electronic Code Book (ECB) mode of operation, and a key size of 168 bits; • "`3DES_CBC_168`" for the three-key TripleDES algorithm, the Cipher Block Chaining (CBC) mode of operation, and a key size of 168 bits; • "`AES_CBC_128`" for the AES algorithm, the CBC mode of operation, and a key size of 128 bits; • "`AES_CBC_256`" for the AES algorithm, the CBC mode of operation, and a key size of 256 bits; • "`AES_XTS_128`" for the AES algorithm, the XTS mode of operation, and a key size of 128 bits; and • "`AES_XTS_256`" for the AES algorithm, the XTS mode of operation, and a key size of 256 bits.	Optional

Continued on next page

Table 112 – continued from previous page

Metadata Name	Type	Description	Requirement
cdmi_value_hash	JSON String	If this data system metadata item is present and not an empty string, it indicates that the client is requesting that the system hash the object value using the hashing algorithm and length requested. The result of the hash shall be provided in the `cdmi_hash` storage system metadata item. Supported algorithm/length values are provided by the `cdmi_value_hash` storage system capability. When this data system metadata item is absent, this data system metadata item shall not be used. If this data system metadata item is present but does not contain a valid hash algorithm/length string, the system is free to choose to ignore the data system metadata, to fail with an HTTP status code of `400 Bad Request`, or to select a hash algorithm/length of the system's choice. Supported hash algorithms are expressed as a string in the form of ALGORITHM LENGTH, where: • "`ALGORITHM`" is the hash algorithm (e.g., "`SHA`"). • "`LENGTH`" is the hash size in bytes (e.g., "`160`", "`256`"). To improve interoperability between CDMI implementations, the following designators should be used for the more common encryption combinations: • "`SHA160`" for SHA-1, and • "`SHA256`" for SHA-2.	Optional
cdmi_latency	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 maximum time to first byte, in milliseconds. This metadata is the desired latency (in milliseconds) to the first byte of data, as measured from the edge of the cloud and factoring out any propagation latency between the client and the cloud. For example, this metadata may be used to determine, in an interoperable way, from what type of storage medium the data may be served. 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_throughput	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 maximum data rate on retrieve, in bytes per second. This metadata is the desired bandwidth to the data, as measured from the edge of the cloud and factoring out any bandwidth capability between the client and the cloud. This metadata is used to stage the data in locations where there is sufficient bandwidth to accommodate a maximum usage. 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

Continued on next page

Table 112 – continued from previous page

Metadata Name	Type	Description	Requirement
cdmi_sanitization_method	JSON String	If this data system metadata item is present and not an empty string, it indicates that the client is requesting that the system use a specific sanitization method to delete data such that the data is unrecoverable after an update or delete operation. Supported sanitization method values are provided by the cdmi_sanitization_method capability. When this data system metadata item is absent, this data system metadata item shall not be used. If this data system metadata item is present but does not contain a valid sanitization method string, the system is free to choose to ignore the data system metadata, to fail with an HTTP status code of `400 Bad Request`, or to select a sanitization method of the system's choice. Supported sanitization methods are defined as system-specific strings.	Optional
cdmi_RPO	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 largest acceptable duration in time between an update or create and when the object may be recovered, specified in seconds. This metadata is used to indicate the desired backup frequency from the primary copy or copies of the data to the secondary copy or copies. It is the maximum acceptable time period before a failure or disaster during which changes to data may be lost as a consequence of recovery. 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_RTO	JSON String	If this data system metadata item is present and set to a positive numeric string, it indicates that the client is requesting the largest acceptable duration in time to restore data, specified in seconds. This metadata is used to indicate the desired maximum acceptable duration to restore the primary copy or copies of the data from a secondary backup copy or copies. 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_authentication_methods	JSON Array of JSON Strings	The client shall set this metadata to a list of authentication methods requested to be enabled for the domain. Supported authentication method values are indicated by the cdmi_authentication_methods capability.	Optional
cdmi_enc_key_id	JSON String	Contains a unique key identifier (e.g. KMIP Identifier) for the symmetric key used to encrypt and decrypt the object.	Optional
cdmi_enc_value_sign_id	JSON String	Contains a unique key identifier (e.g. KMIP Identifier) for the private key used for signing the value of the object.	Optional
cdmi_enc_value_verify_id	JSON String	Contains a unique key identifier (e.g. KMIP Identifier) for the public key or certificate chain used for verifying the signature of the value of the object.	Optional

Continued on next page

Table 112 – continued from previous page

Metadata Name	Type	Description	Requirement
cdmi_enc_object_sign_id	JSON String	Contains a unique key identifier (e.g. KMIP Identifier) for the private key used for signing the entire object.	Optional
cdmi_enc_object_verify_id	JSON String	Contains a unique key identifier (.e.g. KMIP Identifier) for the public key or certificate chain used for verifying the signature of the entire object.	Optional
cdmi_versioning	JSON String	If present, this metadata item indicates that versioning is requested to be enabled for the data object. • If set to the value "`value`", versions shall be created when the value is updated. • If set to the value "`user`", versions shall be created when the value and/or user metadata is updated. • If set to the value "`all`", versions shall be created when any update is performed against the version-enabled data object. This data system metadata item shall not be present in data object versions.	Optional
cdmi_versions_count	JSON String	This metadata item contains the maximum number of historical versions requested to be retained. • If `cdmi_versions_count` is not present, no limit should be placed on the number of versions that are retained. • If `cdmi_versions_count` is present and has a value of zero, only the current version should be retained. • If `cdmi_versions_count` is present and has a value greater than zero, up to the specified number of historical versions should be retained. • If the number of historical versions exceeds the value specified, historical versions should be deleted from the oldest to the newest until the number of historical versions equals the value contained in `cdmi_versions_count`.	Optional
cdmi_versions_age	JSON String	This metadata item contains the maximum age of the oldest historical version requested to be retained, specified as the number of seconds before the current time. • If `cdmi_versions_age` is not present, no limit should be placed on the age of versions that are retained. • If `cdmi_versions_age` is present, historical versions should be retained until their age is greater than the value contained in `cdmi_versions_age`. • If the age of a historical version exceeds the value specified, that historical version should be deleted.	Optional

Continued on next page

Table 112 – continued from previous page

Metadata Name	Type	Description	Requirement
cdmi_versions_size	JSON String	This metadata item contains the maximum amount of space requested to be used to retain historical versions, specified in bytes. • If `cdmi_versions_size` is not present, no limit should be placed on the size of versions that are retained. • If `cdmi_versions_size` is present, historic versions should be retained until the total storage consumption of the historical versions exceeds the value contained in `cdmi_versions_size`. • If the total size consumed by historical versions exceeds the value specified, historical versions should be deleted from the oldest to the newest until the total storage consumption of historical versions is equal or less than the value contained in `cdmi_versions_count`.	Optional

16.3 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 113. Data system-provided metadata items are read only. Updates of these metadata items shall be ignored.

Table 113: Provided Values of Data System Metadata

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 ISO-8601 time interval (as described in Section 5.6) 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 `cdmi_encryption` in Section 16.2 for the format.)	Optional
cdmi_value_hash_provided	JSON String	Contains the algorithm and length being used to hash the object value. See `cdmi_value_hash` in Section 16.2 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 `cdmi_sanitization_method` in Section 16.2 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

Continued on next page

Table 113 – continued from previous page

Metadata Name	Type	Description	Requirement
cdmi_authentication_methods_provided	JSON Array of JSON Strings	Contains a list of authentication methods enabled for the domain. See `cdmi_authentication_methods` in Section 16.2 for the format.	Optional
cdmi_versioning_provided	JSON String	Contains the value "`value`", "`user`", or "`all`" if versioning is enabled for the data object.	Optional
cdmi_versions_count_provided	JSON String	Contains the maximum number of historical versions that will be retained.	Optional
cdmi_versions_age_provided	JSON String	Contains the oldest age of a historical version that will be retained, in seconds before the current time.	Optional
cdmi_versions_size_provided	JSON String	Contains the maximum amount of space that can be used to retain historical versions, in bytes.	Optional

16.4 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.

16.5 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

Access Control

17.1 Access Control Lists

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).

17.1.1 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*, *ACE Flags*, and *ACE Bit Masks*.

17.1.2 ACE Types

Table 114 defines the following ACE types, as specified in section 5.11.1 of **RFC 3530**.

Table 114: ACE Types

String Form	Description	Constant	Bit Mask
"`ALLOW`"	Allow access rights for a principal	"`CDMI_ACE_ACCESS_ALLOW`"	0x00000000
"`DENY`"	Deny access rights for a principal	"`CDMI_ACE_ACCESS_DENY`"	0x00000001
"`AUDIT`"	Generate an audit record when the principal attempts to exercise the specified access rights	"`CDMI_ACE_SYSTEM_AUDIT`"	0x00000002

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.

17.1.3 ACE Who

The special "who" identifiers need to be understood universally, rather than in the context of a particular external security domain (see *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 115: Who Identifiers

Who	Description
"OWNER@"	The owner of the file
"GROUP@"	The group associated with the file
"EVERYONE@"	The world
"ANONYMOUS@"	Access without authentication
"AUTHENTICATED@"	Any authenticated user (opposite of "ANONYMOUS@")
"ADMINISTRATOR@"	A user with administrative status, e.g., "root"
"ADMINUSERS@"	A group whose members are given administrative status

To avoid name conflicts, these special identifiers are distinguished by an appended "@" (with no domain name).

17.1.4 ACE Flags

CDMI allows for nested containers and mandates that objects and subcontainers be able to inherit access permissions from their parent containers. However, it is not enough to simply inherit all permissions from the parent; it might be desirable, for example, to have different default permissions on child objects and subcontainers of a given container. The flags in Table 116 govern this behavior.

Table 116: ACE Flags

String Form	Description	Constant	Bit Mask
"NO_FLAGS"	No flags are set	"CDMI_ACE_FLAGS_NONE"	0x00000000
"OBJECT_INHERIT"	An ACE on which "OBJECT_INHERIT" is set is inherited by objects as an effective ACE: "OBJECT_INHERIT" is cleared on the child object. When the ACE is inherited by a container, "OBJECT_INHERIT" is retained for the purpose of inheritance, and additionally, "INHERIT_ONLY" is set.	"CDMI_ACE_FLAGS_ OBJECT_INHERIT_ACE"	0x00000001
"CONTAINER_INHERIT"	An ACE on which "CONTAINER_INHERIT" is set is inherited by a subcontainer as an effective ACE. Both "INHERIT_ONLY" and "CONTAINER_INHERIT" are cleared on the child container.	"CDMI_ACE_FLAGS_ CONTAINER_INHERIT_ACE"	0x00000002
"NO_PROPAGATE"	An ACE on which "NO_PROPAGATE" is set is not inherited by any objects or subcontainers. It applies only to the container on which it is set.	"CDMI_ACE_FLAGS_ NO_PROPAGATE_ACE"	0x00000004

Continued on next page

Table 116 – continued from previous page

String Form	Description	Constant	Bit Mask
“INHERIT_ONLY”	An ACE on which “INHERIT_ONLY” is set is propagated to children during ACL inheritance as specified by ”OBJECT_INHERIT” and “CONTAINER_INHERIT”. The ACE is ignored when evaluating access to the container on which it is set and is always ignored when set on objects.	“CDMI_ACE_FLAGS_ INHERIT_ONLY_ACE”	0x00000008
“IDENTIFIER_GROUP”	An ACE on which “IDENTIFIER_GROUP” is set indicates that the “who” refers to a group identifier.	“CDMI_ACE_FLAGS_ IDENTIFIER_GROUP”	0x00000040
“INHERITED”	An ACE on which “INHERITED” is set indicates that this ACE is inherited from a parent directory. A server that supports automatic inheritance will place this flag on any ACEs inherited from the parent directory when creating a new object.	“CDMI_ACE_FLAGS_ INHERITED_ACE”	0x00000080

17.1.5 ACE Mask Bits

The mask field of an ACE contains a 32 bit mask, as specified in section 5.11.2 of **RFC 3530**.

Table 117: ACE Bit Masks

String Form	Description	Constant	Bit Mask
“READ_OBJECT”	If true, indicates permission to read the value of an object. If false: • A CDMI GET that requests all fields shall return all permitted fields with the value field excluded. • A CDMI GET that requests specific fields shall return requested permitted fields with the value field excluded. • 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

Continued on next page

Table 117 – continued from previous page

String Form	Description	Constant	Bit Mask
"LIST_CONTAINER"	If true, indicates permission to list the children of an object. If false: • A CDMI GET that requests all fields shall return all permitted fields with the children field and childrenrange field excluded. • A CDMI GET that requests specific fields shall return the requested permitted fields with the children field and childrenrange field excluded. • A CDMI GET for only the children field and/or childrenrange field shall return an HTTP status code of 403 Forbidden.	"CDMI_ACE_LIST_CONTAINER"	0x00000001
"WRITE_OBJECT"	If true, indicates permission to modify the value of an object If false, 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"	If true, indicates permission to add a new child data object or queue object. If false, 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"	If true, indicates permission to append data to the value of a data object. If "APPEND_DATA" is true and "WRITE_OBJECT" is false, 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"	If true, indicates permission to create a child container object or domain object. If false, a PUT that requests creation of a new child container object or new domain object shall return an HTTP status code of 403 Forbidden.	"CDMI_ACE_ADD_SUBCONTAINER"	0x00000004

Continued on next page

Table 117 – continued from previous page

String Form	Description	Constant	Bit Mask
"READ_METADATA"	If true, indicates permission to read the metadata of an object. If false: • A CDMI GET that requests all fields shall return all permitted fields with the metadata field excluded. • A CDMI GET that requests specific fields shall return the requested permitted fields with the metadata field excluded. • A CDMI GET for only the metadata field shall return an HTTP status code of 403 Forbidden.	"CDMI_ACE_READ_METADATA"	0x00000008
"WRITE_METADATA"	If true, indicates permission to modify the metadata of an object. If false, a CDMI PUT that requests modification of the metadata field of an object shall return an HTTP status code of 403 Forbidden.	"CDMI_ACE_WRITE_METADATA"	0x00000010
"EXECUTE"	If true, indicates permission to execute an object.	"CDMI_ACE_EXECUTE"	0x00000020
"TRAVERSE_CONTAINER"	If true, indicates permission to traverse a container object or domain object. If false, all operations against all children below the container shall return an HTTP status code of 403 Forbidden.	"CDMI_ACE_TRAVERSE_CONTAINER"	0x00000020
"DELETE_OBJECT"	If true, indicates permission to delete a child data object or child queue object from a container object. If false, all DELETE operations shall return an HTTP status code of 403 Forbidden.	"CDMI_ACE_DELETE_OBJECT"	0x00000040
"DELETE_SUBCONTAINER"	If true, indicates permission to delete a child container object from a container object or to delete a child domain object from a domain object. If false, all DELETE operations shall return an HTTP status code of 403 Forbidden.	"CDMI_ACE_DELETE_SUBCONTAINER"	0x00000040

Continued on next page

Table 117 – continued from previous page

String Form	Description	Constant	Bit Mask
"READ_ATTRIBUTES"	If true, indicates permission to read the attribute fields[1] of an object. If false: • A CDMI GET that requests all fields shall return all non-attribute fields and shall not return any attribute fields. • A CDMI GET that requests at least one non-attribute field shall only return the requested non-attribute fields. • A CDMI GET that requests only non-attribute fields shall return an HTTP status code of 403 Forbidden.	"CDMI_ACE_READ_ATTRIBUTES"	0x00000080
"WRITE_ATTRIBUTES"	If true, indicates permission to change attribute fields[#a]_ of an object. If false, a CDMI PUT that requests modification of any non-attribute field shall return an HTTP status code of 403 Forbidden.	"CDMI_ACE_WRITE_ATTRIBUTES"	0x00000100
"WRITE_RETENTION"	If true, indicates permission to change retention attributes of an object. If false, a CDMI PUT that requests modification of any non-hold retention metadata items shall return an HTTP status code of 403 Forbidden.	"CDMI_ACE_WRITE_RETENTION"	0x00000200
"WRITE_RETENTION_HOLD"	If true, indicates permission to change retention hold attributes of an object. If false, a CDMI PUT that requests modification of any retention hold metadata items shall return an HTTP status code of 403 Forbidden.	"CDMI_ACE_WRITE_RETENTION_HOLD"	0x00000400
"DELETE"	If true, indicates permission to delete an object. If false, all DELETE operations shall return an HTTP status code of 403 Forbidden.	"CDMI_ACE_DELETE"	0x00010000

Continued on next page

Table 117 – continued from previous page

String Form	Description	Constant	Bit Mask
“READ_ACL”	If true, indicates permission to read the ACL of an object. If false: • A CDMI GET that requests all metadata items shall return all permitted metadata items with the “cdmi_acl” metadata item excluded. • A CDMI GET that requests specific metadata items shall return the requested permitted metadata items with the “cdmi_acl” metadata item excluded. • A CDMI GET for only the cdmi_acl metadata item shall return an HTTP status code of 403 Forbidden. If “READ_ACL” is true and “READ_METADATA” is false, then to read the ACL, a client CDMI GET for only the “cdmi_acl” metadata item shall be permitted.	“CDMI_ACE_READ_ACL”	0x00020000
“WRITE_ACL”	If true, indicates permission to write the ACL of an object. If false: • If “WRITE_ACL” is false, a CDMI PUT that requests modification of the “cdmi_acl” metadata item shall return an HTTP status code of 403 Forbidden. • If “WRITE_ACL” is true and “WRITE_METADATA” is false, then to write the ACL, a client CDMI PUT for only the “cdmi_acl” metadata item shall be permitted.	“CDMI_ACE_WRITE_ACL”	0x00040000

Continued on next page

Table 117 – continued from previous page

String Form	Description	Constant	Bit Mask
"WRITE_OWNER"	If true, indicates permission to change the owner of an object. If false: • If "WRITE_OWNER" is false, a CDMI PUT that requests modification of the "cdmi_owner" metadata item shall return an HTTP status code of 403 Forbidden. • If "WRITE_OWNER" is true and "WRITE_METADATA" is false, then to write the owner, a client CDMI PUT for only the "cdmi_owner" metadata item shall be permitted.	"CDMI_ACE_WRITE_OWNER"	0x00080000
"SYNCHRONIZE"	If true, indicates permission to access an object locally at the server with synchronous reads and writes.	"CDMI_ACE_SYNCHRONIZE"	0x00100000

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.

17.1.6 ACL Evaluation

When evaluating whether access to a particular object O by a principal P is to be granted, the server shall traverse the object's logical ACL (its ACL after processing inheritance from parent containers) in list order, using a temporary permissions bitmask m, initially empty (all zeroes).

- If the object still does not contain an ACL, the algorithm terminates and access is denied for all users and groups. This condition is not expected, as CDMI implementations should require an inheritable default ACL on all root containers.
- ACEs that do not refer to the principal P requesting the operation are ignored.
- If an ACE is encountered that denies access to P for any of the requested mask bits, access is denied and the algorithm terminates.
- If an ACE is encountered that allows access to P, the permissions mask m for the operation is XORed with the permissions mask from the ACE. If m is sufficient for the operation, access is granted and the algorithm terminates.
- If the end of the ACL list is reached and permission has neither been granted nor explicitly denied, access is denied and the algorithm terminates, unless the object is a container root. In this case, the server shall:
  - allow access to the container owner, "ADMINISTRATOR@", and any member of "ADMINUSERS@"; and
  - log an event indicating what has happened.

When permission for the desired access is not explicitly given, even "ADMINISTRATOR@" and equivalents are denied for objects that aren't container roots. When an admin needs to access an object in such an instance, the root container shall be accessed and its inheritable ACEs changed in a way as to allow access to the original object. The resulting log entry then provides an audit trail for the access.

When a root container is created and no ACL is supplied, the server shall place an ACL containing the following ACEs on the container:

[1] The value fields, children fields, and metadata field are considered to be non-attribute fields. All other fields are considered to be attribute fields.

```
"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*, *ACE Flags*, and *ACE Bit Masks*:

```
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   = aceobinheritiitem | acecontinheritiitem | acenopropagateitem |
↪aceinheritonlyitem

aceobinheritiitem   = "CDMI_ACE_OBJECT_INHERIT_ACE" | 0x01
acecontinheritiitem = "CDMI_ACE_CONTAINER_INHERIT_ACE" | 0x02
acenopropagateitem = "CDMI_ACE_NO_PROPAGATE_INHERIT_ACE" | 0x04
aceinheritonlyitem = "CDMI_ACE_INHERIT_ONLY_ACE" | 0x08

acemaskstring  =   acemaskitem [| acemaskitem ...]
acemaskitem    =   acereaditem | acewriteitem | aceappenditem | acereadmetaitem |
↪acewritemetaitem | acedeleteitem | acedelselfitem | acereadaclitem | acewriteaclitem |
↪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":
[
```

(continues on next page)

(continued from previous page)

```
    {
        "acetype": "ALLOW",
        "identifier": "OWNER@",
        "aceflags": "OBJECT_INHERIT, CONTAINER_INHERIT",
        "acemask": "ALL_PERMS"
    }
]
```

17.1.7 Example ACE Mask Expressions

Example 1:

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

evaluates to 0x09 | 0x02 == 0x0

Example 2:

```
0x001F07FF
```

evaluates to 0x001F07FF == “ALL_PERMS”

Example 3:

```
"RW_ALL" | DELETE
```

evaluates to 0x000601DF | 0x00100000 == 0x000701DF

17.1.8 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 118: ACE Bit Mask/String

Hex Form	Object String Form	Container String Form
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.

17.1.9 JSON Format for ACLs

ACE flags and masks are members of a 32-bit quantity that is widely understood in its hexadecimal representations. The JSON data format does not support hexadecimal integers, however. For this reason, all hexadecimal integers in CDMI ACLs shall be represented as quoted strings containing a leading "`0x`".

ACLs containing one or more ACEs shall be represented in JSON as follows:

```
{
    "cdmi_acl" : [
        {
            "acetype" : "0xnn",
            "identifier" : "<user-or-group-name>",
            "aceflags" : "0xnn",
            "acemask" : "0xnn"
        },
        {
            "acetype" : "0xnn",
            "identifier" : "<user-or-group-name>",
            "aceflags" : "0xnn",
            "acemask" : "0xnn"
        }
    ]
}

ACEs in such an ACL shall be evaluated in order as they appear.
```

EXAMPLE 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!"
}
```

Clause 18

Retention and Hold Management

18.1 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.

18.2 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.

18.3 CDMI Retention

18.3.1 Overview

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

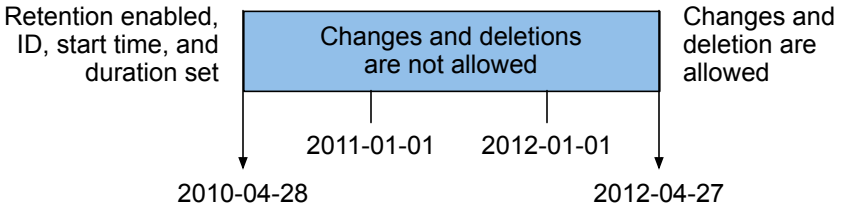
Retention management uses time criteria to determine the time period during which CDMI object deletion from the CDMI-based system shall be prohibited. CDMI retention criteria shall be specified by the following data system metadata:

- 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. 9 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.



Example: Retention start date of 2010-04-28 with a duration of 730 days. No holds.

Fig. 9: 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.

18.3.2 Examples

EXAMPLE 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

{
    "metadata" : {
        "cdmi_retention_id" : "1",
        "cdmi_retention_period" : "2010-04-28T00:00:00.000000Z/2012-04-27T00:00:00.
↪000000Z"
    }
}
```

The following shows the response.

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

EXAMPLE 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

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

The following shows the response.

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

EXAMPLE 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

{
    "metadata" : {
        "cdmi_retention_period" : "2011-04-28T00:00:00.000000Z/2012-01-27T00:00:00.
↪000000Z"
    }
}
```

The following shows the response.

```
HTTP/1.1 403 Forbidden
```

18.4 CDMI Hold

18.4.1 Overview

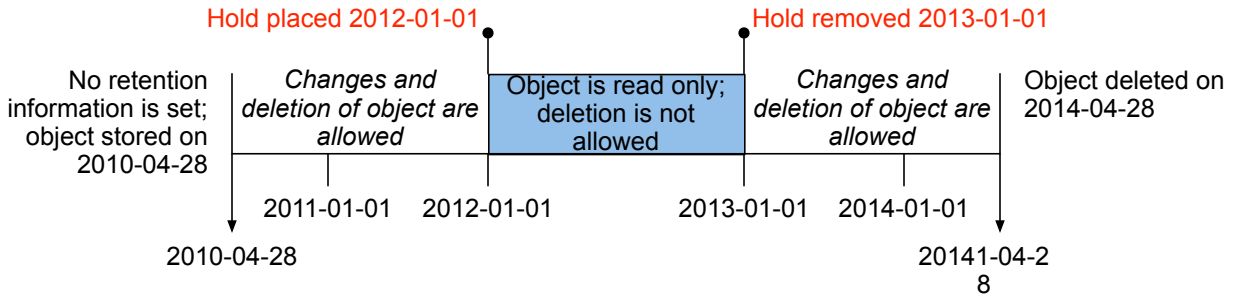
CDMI hold enforces read-only data object access and prohibition of object deletion. A cloud storage system shall allow multiple holds to be applied to a single object to satisfy multiple hold orders. While an object is on hold, a cloud storage system shall strictly enforce read-only access to the object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

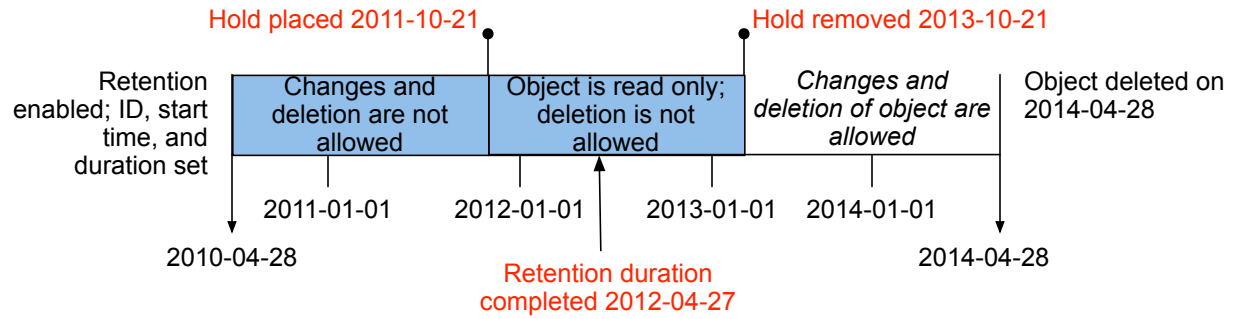
Fig. 10 shows how placing a hold on an object affects its read-only and deletion capability.



Example: Hold placed on the object on 2012-01-01 and removed on 2013-01-01

Fig. 10: Object Hold

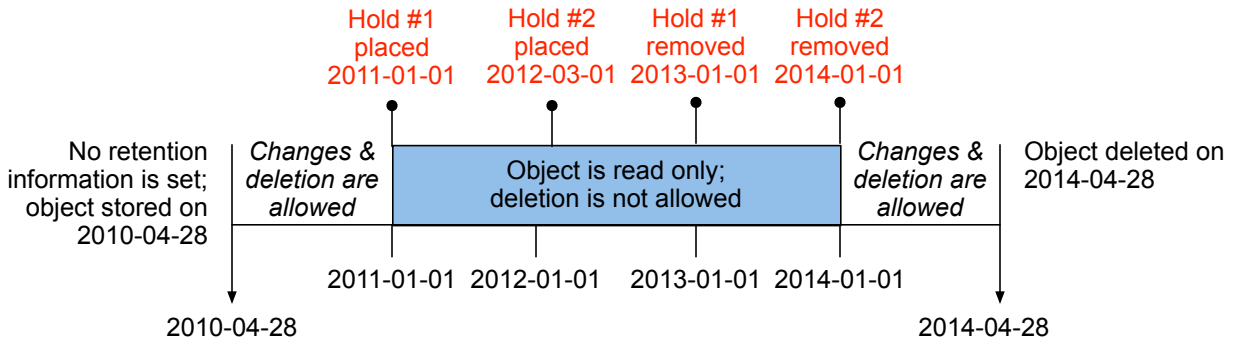
Fig. 11 shows how to establish time-based retention with a retention identifier that has a hold placed on the object. The value of the object data system metadata for the retention period shall not be reduced, and the value of the object data system metadata for hold identifiers shall not permit holds to be removed. Removing holds is outside the scope of the CDMI international standard.



Example: Start date of 2010-04-28 with a duration of 730 days; hold placed on the object

Fig. 11: Object Hold on Object with Retention

Fig. 12 shows how placing multiple holds on an object affects its read-only and deletion capability.



Example: Start date of 2010-04-28 with a duration of 730 days; holds placed on the object

Fig. 12: Object with Multiple Holds

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.

18.4.2 Examples

EXAMPLE 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

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

The following shows the response.

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

EXAMPLE 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

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

3469 The following shows the response.

```
HTTP/1.1 403 Forbidden
```

3470 EXAMPLE 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

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

3471 The following shows the response.

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

18.5 CDMI Auto-deletion

18.5.1 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 `cdmi_retention_autodelete` in *Data System Metadata*.)

CDMI objects shall be automatically deleted by the system at the retention period expiration by setting the `cdmi_retention_autodelete` data system metadata item. The `cdmi_retention_autodelete` data system metadata item 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.

EXAMPLE 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

{
    "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
```

18.6 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.

EXAMPLE 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 19

Scope Specification

19.1 Overview

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

{
    "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 scope specification JSON object represents a condition that shall be met for a field.

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

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

EXAMPLE 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/"
    }
]
```

EXAMPLE 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.

EXAMPLE 4: 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.

EXAMPLE 5: To query metadata items, the metadata object is included as an object within the query request. This query is shown as follows:

```
[
    {
        "metadata" : {
```

(continues on next page)

(continued from previous page)

```
            "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.

EXAMPLE 6: To query all objects with an ACE associated with the user "`jdoe`":

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

EXAMPLE 7: 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.

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

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

19.3 Query Matching Expressions

Query matching expressions are structured as “<operator>” or “<operator><sp><constant>”, and are defined in Table 119.

Table 119: Query Matching Expressions

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. 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.
“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. If the constant value does not match the value of the field, the condition shall be considered to be met.
“field”: “#!= constant”	The numeric equals matching expression tests for non-equality of the numeric equality of the value of the field and a specified constant 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. 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.
“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. 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.
“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. 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.
“field”: “<= constant”	The less than or equals to matching expression tests if the value of the field is lexicographically less than or equal to a specified constant value. The less than or equal test is case sensitive. If the constant value is less than or equal to the value of the field, the condition shall be considered to be met.
“field”: “#<= constant”	The numeric less than or equals to matching expression tests if the numeric value of the field is less than or equal to a specified constant value.
“field”: “starts constant”	The starts with matching expression tests if the field value starts with a specified constant value. If the constant value is equal to the start of the value of the field, the condition shall be considered to be met.
“field”: “!starts constant”	The not starts with matching expression tests if the field value does not start with a specified constant value. If the constant value is not equal to the start of the value of the field, the condition shall be considered to be met.
“field”: “ends constant”	The ends with matching expression tests if the field value ends with a specified constant value. If the constant value is equal to the end of the value of the field, the condition shall be considered to be met.

Continued on next page

Table 119 – continued from previous page

Matching Expression	Description
“field”:“!ends constant”	The not ends with matching expression tests if the field value does not end with a specified constant value. If the constant value is not equal to the end of the value of the field, the condition shall be considered to be met.
“field”:“contains constant”	The contains matching expression tests if the field value contains a specified constant value. If the constant value is found as a substring within the value of the field, the condition shall be considered to be met. The contains operator is only supported if the “cdmi_query_contains” capability is present.
“field”:“!contains constant”	The not contains matching expression tests if the field value does not contain a specified constant value. If the constant value is not found as a substring within the value of the field, the condition shall be considered to be met. The not contains operator is only supported if the “cdmi_query_contains” capability is present.
“field”:“tag constant”	The tag matching expression tests if the field value contains a specified constant tag value. The leading space character after the “tag” and before the constant value is not included in the comparison. The tag test is not case sensitive. If the constant value is found as a tag substring within the value of the field, the condition shall be considered to be met. Tag substrings start at the beginning of the value or a “,”, and end at the next “,” or the end of the string. Whitespace before and after “,” characters shall be stripped for the purpose of comparisons. Tag matching expressions are only supported if the “cdmi_query_tags” capability is present.
“field”:“!tag constant”	The not tag matching expression tests if the field value does not contain a specified constant tag value. The leading space character after the “!tag” and before the constant value is not included in the comparison. The not tag test is not case sensitive. If the constant value is not found as a tag substring within the value of the field, the condition shall be considered to be met. Tag substrings start at the beginning of the value or a “,”, and end at the next “,” or the end of the string. Whitespace before and after “,” characters shall be stripped for the purpose of comparisons. Tag matching expressions are only supported if the “cdmi_query_tags” capability is present.
“field”:“=~ constant”	The regular expression matching expression tests if the field value matches a specified constant regular expression value. If the regular expression evaluates to true against the value, the condition shall be considered to be met. Regular expression strings shall be processed according to the POSIX Extended Regular Expression (ERE) standard, as specified in *IEEE 1003.1*. Regex matching expressions are only supported if the “cdmi_query_regex” capability is present.
“field”:“!~ constant”	The not regular expression matching expression tests if the field value does not match a specified constant regular expression value. If the regular expression evaluates to false against the value, the condition shall be considered to be met. Regular expression strings shall be processed according to the POSIX Extended Regular Expression (ERE) standard, as specified in *IEEE 1003.1*. Regex matching expressions are only supported if the “cdmi_query_regex” capability is present.

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.

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.

EXAMPLE 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/"
    }
]
```

EXAMPLE 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 20

Results Specification

20.1 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

{
    "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" : "..."
}
```

20.2 Examples

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

EXAMPLE 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" : ""
        }
    }
}
```

EXAMPLE 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.

EXAMPLE 3: To request all metadata items be returned for each matching object, the following `cdmi_results_specification` shall be used:

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

EXAMPLE 4: 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 21

Notification Queues

21.1 Overview

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.

21.1.1 Required Metadata

When creating a notification queue, the metadata described in Table 120 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.

Table 120: Required metadata for a notification queue

Metadata Name	Type	Description	Requirement
cdmi_queue_type	JSON String	The queue type indicates how the cloud storage system shall manage the queue object. The type of `cdmi_notification_queue` is defined for notification queues.	Mandatory

Continued on next page

Table 120 – continued from previous page

Metadata Name	Type	Description	Requirement
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. • `cdmi_modify_complete` - Notifications are generated when an existing object is modified and is in the “Complete” completion status. This notification is also generated when an existing object being modified transitions from “Processing” to “Complete”. When an object transitions from “Processing” completion status, the “`cdmi_event_result`” is the HTTP result code that would have been returned if the modify operation was not delayed. • `cdmi_rename` - Notifications are generated when an object is renamed as part of a move operation. • `cdmi_copy` - Notifications are generated for the newly created copied object when the copy is completed. • `cdmi_reference` - Notifications are generated when a reference is created. • `cdmi_delete` - Notifications are generated when an object is deleted. • `cdmi_export` - Notifications are generated when a container is exported. • `cdmi_snapshot` - Notifications are generated when a container snapshot is created. • <implementor-specific events> Clients may include the desired notification event types in the `cdmi_notification_events` JSON array. If all notifications events are desired, an empty JSON array shall be used.	Mandatory

Continued on next page

Table 120 – continued from previous page

Metadata Name	Type	Description	Requirement
cdmi_scope_specification	JSON Array of JSON Objects	The scope specification determines the set of objects on which operations trigger the generation of notifications. If notifications are desired for all objects, include an empty JSON array. See Section 19 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 notification scope specification. See Section 20 for how to construct a results specification. In addition to the fields defined in Section 20, for notifications, four additional fields are defined: • cdmi_event - Indicates the event as specified in the "cdmi_notification_events" field that triggered the notification; • cdmi_event_result - Indicates the status result of the event that triggered the notification. The status is the same as the status that was returned over the HTTP request, i.e., 200 OK, 404 Not Found, etc.; • cdmi_event_time - Indicates the time of the event that triggered the notification. The time will be formatted in ISO-8601 time (see time_representations and ref_iso_8601:2004); and • cdmi_event_user - Indicates the principal (ACL name) of the user that caused the event that triggered the notification. If the system triggered the event, the name will be left as an empty string.	Mandatory

3602 EXAMPLE 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" : ""
```

(continues on next page)

(continued from previous page)

```
            }
        }
    }
}
```

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.

EXAMPLE 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.

21.1.2 System-created metadata

Table 121 describes the system-created metadata that provides details on the status of the notification queue.

Table 121: 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 122 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 122: Required Metadata for a Query Queue

Metadata Name	Type	Description	Requirement
cdmi_queue_type	JSON String	The queue type indicates how the cloud storage system shall manage the queue object. The type of "`cdmi_query_queue`" is defined for query queues.	Mandatory
cdmi_scope_specification	JSON Array of JSON Objects	The scope specification determines which objects are included in the query results. This scope specification is similar to a "`WHERE`" clause in SQL-like languages. To query all objects, specify an empty JSON array. See Section 19 for how to construct a scope specification.	Mandatory

Continued on next page

Table 122 – continued from previous page

Metadata Name	Type	Description	Requirement
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 Section 20 for how to construct a results specification.	Mandatory

EXAMPLE 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.

EXAMPLE 2: An example of a query result JSON object is as follows:

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

Table 123 describes the system-created metadata that provides details on the status of the query queue.

Table 123: 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.

NOTE: 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`".

Clause 23

Encrypted Objects

23.1 Overview

A cloud storage system may optionally implement additional operations against encrypted objects. Support for these operations are indicated by the presence of the cloud storage system-wide capabilities for encrypted objects.

Encrypted object operations include the ability to encrypt, re-encrypt, and decrypt objects that are already stored in the cloud (in-place), to sign and verify the signature of encrypted objects, and to access and update the plaintext associated with encrypted objects.

The CDMI International Standard does not specify the method by which keys are managed. Key management services are provided by an external key management system (KMS), and the use of the KMIP standard is given as an example of how a CDMI server interacts with an external KMS.

CDMI objects can contain values that are encrypted. Operations against an encrypted CDMI object are only supported if the encrypted object value is a valid CMS or JWE JSON format. The CMS or JWE JSON object shall include an embedded mimetype of the encrypted object. For JWE, the "`cty`" header shall be used for this purpose.

23.2 Encryption Operations

The state transition diagram for encrypted objects is shown in Figure Fig. 13:

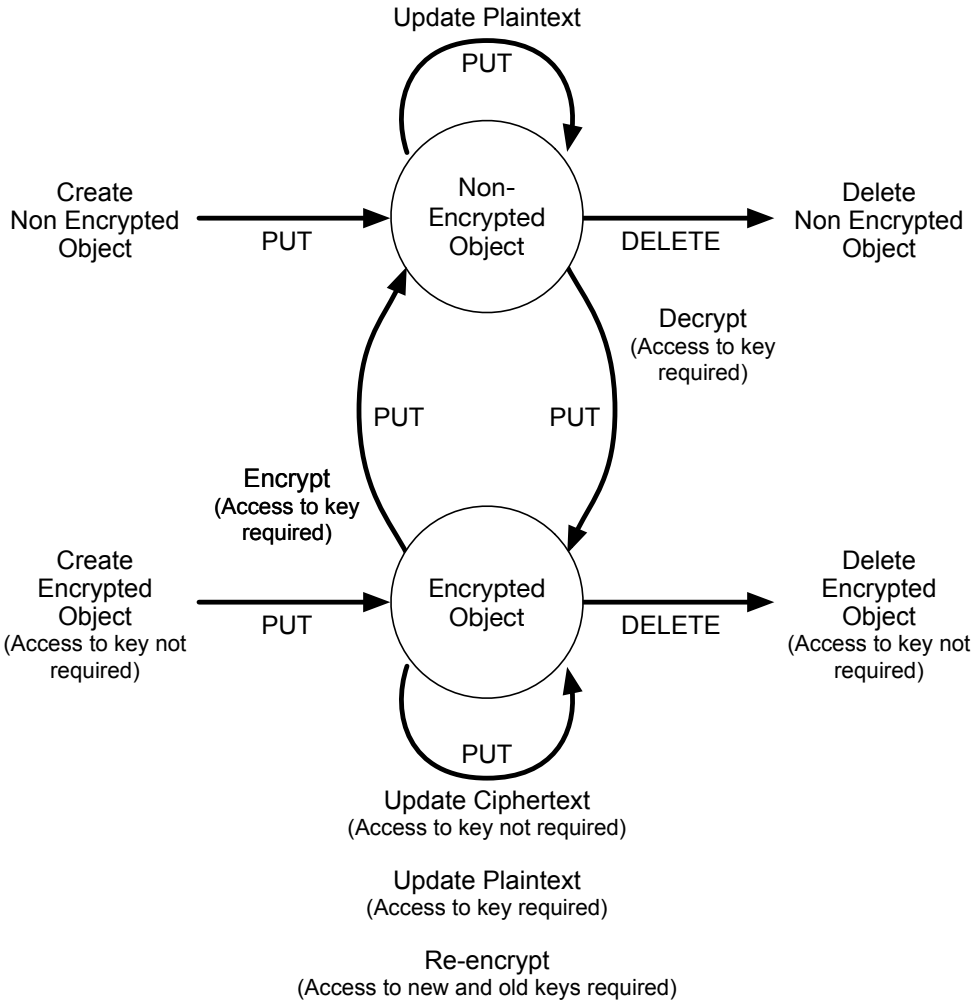


Fig. 13: Encrypted Object State Transistions

The following eight encryption operations are defined:

23.2.1 Create a new encrypted object

Client-encrypted objects shall be stored to a CDMI server using a standard HTTP or CDMI PUT operation, as described in clauses Section 7.2 and Section 8.2. The client shall indicate that an object is encrypted by specifying a mimetype of "`application/cms`" or "`application/jose+json`".

A client may register an encryption key, signing keys and/or verification keys with a Key Management System (KMS), and may indicate the Key IDs in `cdmi_enc_key_id`, `cdmi_enc_value_sign_id`, `cdmi_enc_object_sign_id`, `cdmi_enc_value_verify_id`, and/or `cdmi_enc_object_verify_id` metadata items. This allows the CDMI server to access the keys from the KMS on behalf of a client, when needed.

Creating an encrypted objects on a CDMI server does not require any encryption-specific capabilities to be supported, and is backwards compatible with earlier versions of the CDMI standard. This permits encrypted objects to be stored and transferred by CDMI servers that do not support encryption-specific functionality.

23.2.2 Delete an encrypted object

Encrypted objects shall be deleted using a standard HTTP or CDMI DELETE operation, as described in clause 7.5 and clause 8.5. Any client with sufficient permissions shall be permitted to delete an encrypted object, regardless of if they can access the decryption keys.

23.2.3 Encrypt an unencrypted object

Existing unencrypted objects shall be encrypted in-place by performing a CDMI PUT operation, as described in clause 8.4, that changes the object mimetype to "`application/cms`" or "`application/jose+json`" and specifies a `cdmi_enc_key_id` metadata item. The client may also specify a `cdmi_enc_value_sign_id` and/or `cdmi_enc_value_verify_id` metadata item to indicate that the object is to be signed, and to provide signature verification information.

The CDMI Server shall use the client's credentials (which are included in HTTP headers, and are out of scope of this International Standard) to retreive the encryption and signing keys, and encryption and signing algorithm information from the KMS, and shall use the keys to encrypt and sign the value of the object. The mimetype of the encrypted value is stored in the CMS wrapper, or in a "`cty`" field of the JWE JSON.

23.2.4 Decrypt an encrypted object

Existing encrypted objects shall be decrypted in-place by performing a CDMI PUT operation, as described in clause 8.4, that changes the object mimetype from "`application/cms`" or "`application/jose+json`" to the original mimetype as specified in the CMS wrapper, or in the "`cty`" field of the JWE JSON. Specifying any other fields or metadata shall return a ``400 Bad Request`` result code.

The CDMI Server shall use the client's credentials (which are included in HTTP headers, and are out of scope of this International Standard) to retreive the encryption, signing and verification keys, and encryption, signing and verification algorithm information from the KMS, and shall use the keys to decrypt and verify the encrypted value and user metadata included in the object.

23.2.5 Re-encrypt an encrypted object

Existing encrypted objects shall be re-encrypted in-place by performing a CDMI PUT operation, as described in clause 8.4, that retains the object mimetype of "`application/cms`" or "`application/jose+json`", or changes the object mimetype from "`application/cms`" to "`application/jose+json`", or vice-versa. The client shall also specify a new `cdmi_enc_key_id`, `cdmi_enc_value_sign_id` and/or `cdmi_enc_value_verify_id` metadata item to indicate the new key(s) to be used. Specifying any other fields or metadata shall return a `400 Bad Request` result code.

The CDMI Server shall use the client's credentials (which are included in HTTP headers, and are out of scope of this International Standard) to retreive both the original encryption and signing keys using the original metadata values, and the new encryption and signing keys using the new metadata values from the KMS, and shall use these keys to decrypt, verify, encrypt and sign the value of the object, as needed.

If an encrypted object does not have an existing `cdmi_enc_key_id` metadata item, does not have a "`kid`" header, and no keys are associated with the Object ID, the specified metadata shall be added to the object, and no re-encryption operation shall be performed.

23.2.6 Access ciphertext of an encrypted object

The ciphertext content of an encrypted object shall be read by performing an HTTP GET operation, as described in clause 6.3, with an Accept header value of "`application/cms`" or "`application/jose+json`", depending on the mimetype of the encrypted object.

The ciphertext content of an encrypted object shall also be read by performing a CDMI GET operation, as described in `clause %s`.

23.2.7 Access plaintext of an encrypted object

The plaintext value of an encrypted object shall be read by performing an HTTP GET operation, as described in clause 6.3, with an Accept header value other than "`application/cms`" or "`application/jose+json`", typically "`*/*`". Object plaintext cannot be transparently accessed using a CDMI GET.

The CDMI Server shall use the client's credentials (which are included in HTTP headers, and are out of scope of this International Standard) to retreive the encryption, signing and verification keys, and encryption, signing and verification algorithm information from the KMS, and shall use the keys to decrypt and verify the encrypted value included in the object.

When an encrypted object is decrypted for access, the plaintext shall not be retained or cached by the CDMI server.

23.2.8 Update plaintext of an encrypted object

The plaintext value of an encrypted object shall be modified by performing an HTTP PUT operation, as described in clause 6.4, with an Content-Type header value other than "`application/cms`" or "`application/jose+json`", typically "`*/*`"., depending on the mimetype of the encrypted object. Object plaintext cannot be transparently modified using a CDMI GET.

The CDMI Server shall use the client's credentials (which are included in HTTP headers, and are out of scope of this International Standard) to retreive the encryption, signing and verification keys, and encryption, signing and verification algorithm information from the KMS, and shall use the keys to decrypt and verify the encrypted value, update the value, and re-encrypt/re-sign the updated value.

When an encrypted object is decrypted for update, the plaintext shall not be retained or cached by the CDMI server.

23.2.9 Other CDMI Operations

Other operations specifed by this International Standard (such as copying, serializing, querying, etc) treat an encrypted value the same way as a non-encrypted value.

23.3 Example Uses of Encrypted Objects

Encrypted objects can be used with CDMI systems in the following ways:

- **Passthrough** – A client may store an encrypted object in any format in a CDMI server, with the ciphertext being accessible to the server and to other authorized clients. No access to the plaintext is provided. Passthrough use is compatible with all CDMI systems and is useful when the clients manage all security-related operations and want to protect against potentially untrustworthy clouds.
- **Server-side encryption and signing** – A client may instruct a CDMI server that supports encrypted object operations to take an existing CDMI object and encrypt or encrypt and sign it in place into CMS or JWE JSON representation, where the value of the object is persistently stored from that point on in an encrypted format. Server-side encryption and signing is useful when clients trust the CDMI server and want to increase object security without having to re-upload the data.
- **Server-side decryption** – A client may instruct a CDMI server that supports encrypted object operations to take an existing CDMI object and decrypt it in place from a CMS or JWE JSON representation, where the value of the object is persistently stored from that point on in a decrypted format. Server-side decryption is useful when a client trusts the CDMI server and wants to decrease object security without having to re-upload the data.
- **Client access decryption** – A CDMI server may automatically attempt to decrypt an encrypted object when accessed via HTTP. Client access decryption is useful to provide transparent access to authorized HTTP clients without requiring modifications to the HTTP clients.
- **Cloud access decryption** – A CDMI server may automatically decrypt encrypted objects when it has access to the decryption keys. Cloud access decryption is useful for cloud-resident data processing performed by the CDMI server, such as virus scanning, query, and analytics.
- **Signature verification** – A CDMI server can automatically verify signatures that are attached to encrypted objects that include a signature. Signature verification is useful for detecting corruption or alteration before delivering data to a client.

23.4 KMS Integration

The encryption key is obtained from the KMS using a unique identifier that is stored in the `cdmi_enc_key_id` metadata item associated with the encrypted object. If this metadata item is not present, the CDMI object ID shall be used to locate the key.

When a client requests that an operation be performed that requires accessing the key for the object, the CDMI server evaluates the credentials provided by the client to determine if the client is authorized to perform the requested operation. If the operation is permitted, the CDMI server retrieves the key from the KMS to complete the requested operation. To retrieve the key, the client may be required to provide additional information in the HTTP request that the CDMI server can then use to authenticate to the KMS.

The CDMI International Standard does not specify the mechanism by which the CDMI server communicates with the KMS. In this International Standard, the KMIP protocol is used as an example. CMS and JWE strings for algorithms, key lengths, etc., need to be mapped to the strings used by the KMS (see KMIP clause 9.1.3.2.7).

All keys are created and managed externally to the CDMI server, typically by the client or a system operating on behalf of the client. As a consequence, the CDMI server requires read-only access to the KMS. The CDMI server shall not cache keys.

23.5 CMS Format

Any valid CMS-formatted data can be stored to a CDMI server. However, encrypted object operations are only defined for the following subset of valid CMS-formatted data.

For encryption operations, the CDMI server shall support the following:

- `EnvelopedData`
- `EncryptedContentInfo`
- `contentEncryptionAlgorithm` value listed in the `cdmi_cms_encryption` capability of that CDMI server

For signature operations, the CDMI server shall support the following:

- `AuthenticatedData`
- `SignedData`
- `digestAlgorithms` value listed in the `cdmi_cms_digest` capability of that CDMI server
- `SignerInfo`
- `signatureAlgorithm` value listed in the `cdmi_cms_signature` capability of that CDMI server

The following CMS-formatted data may be ignored: `recipientInfos`

23.6 JOSE Format

Any valid JWE-formatted data can be stored to a CDMI server. However, encrypted object operations are only defined for the following subset of valid JWE-formatted data.

For encryption operations, the CDMI server shall support the following:

- JWE with Direct Encryption (Symmetric Key from KMS)
- JWE with Key Encryption (Public Key from KMS)

For signature operations, the CDMI server shall support the following:

- JWS RSA (Private Key from KMS)
- JWS ECDSA (Private Key from KMS)
- JWS HMAC-SHA2 (Symmetric Key from KMS)

The following JOSE-formatted data may be ignored: Multiple recipients and multiple signatures

23.7 Signature/Digest Verification

If a signature is present as part of the CMS or JWE JSON value, the CDMI server shall verify that the signature of the value is valid before allowing plaintext access or modification.

If a whole-object signature is present, the CDMI server shall verify that the signature contained in the `cdmi_enc_signature` metadata item is valid before allowing any read operations for the object. Write operations are permitted for an object with an invalid or unverifiable whole-object signature.

When present, a whole-object signature shall be attached as a "`cdmi_enc_signature`" metadata item in JWS compact format, with the second field (the JWS payload field) replaced with an empty string as described in Appendix F of RFC 7515.

For signature creation and verification, payload field shall be computed using the following process:

1. Create a serialized representation of the CDMI object, as described in clause 15
2. Remove the following metadata items, if present:
   - `cdmi_atime`
   - `cdmi_acount`
   - `cdmi_enc_signature`
   - Any `*_provided` metadata items
3. Sort all JSON objects in the serialized CDMI object according to the following rules:
   - Within each JSON object, name/value pair entries shall be sorted lexicographically by name
   - Within each JSON array, the initial order shall be preserved
4. Remove all JSON whitespace
5. Base64 URL encode, according to the JWS **RFC 7515**

23.8 Error Handling

If a decryption or signature validation operation is requested against a CDMI object containing an invalid CMS or JWE JSON representation, an HTTP status code of `500 Internal Error` shall be returned to the client.

If a decryption or signature validation operation is requested against a CDMI object containing a valid CMS or JWE JSON representation that uses an unsupported algorithm or feature, an HTTP status code of `501 Not Implemented` shall be returned to the client.

If a decryption or signature validation operation is requested against a CDMI object containing a valid CMS or JWE JSON representation, but the required keys are temporarily unavailable given the credentials presented, an HTTP status code of `408 Request Timeout` shall be returned to the client.

If a decryption or signature validation operation is requested against a CDMI object containing a valid CMS or JWE JSON representation, but the required keys are unavailable given the credentials presented, an HTTP status code of `401 Unauthorized` shall be returned to the client.

If a decryption or signature validation operation is requested against a CDMI object containing a valid CMS or JWE JSON representation, valid keys are available, and signature verification fails, an HTTP status code of `403 Forbidden` shall be returned to the client.

Clause 24

Delegated Access Control

24.1 Overview

CDMI access control is based around Access Control Lists (ACLs) that are stored as object metadata. When a client requests to perform an operation against a CDMI object, the CDMI server shall validate the client's identity and credentials against the object ACL to determine if the operation is allowed. This request assumes that the CDMI server is trusted and capable of making these access control decisions.

Fig. 14 illustrates an ACL-based access control request:

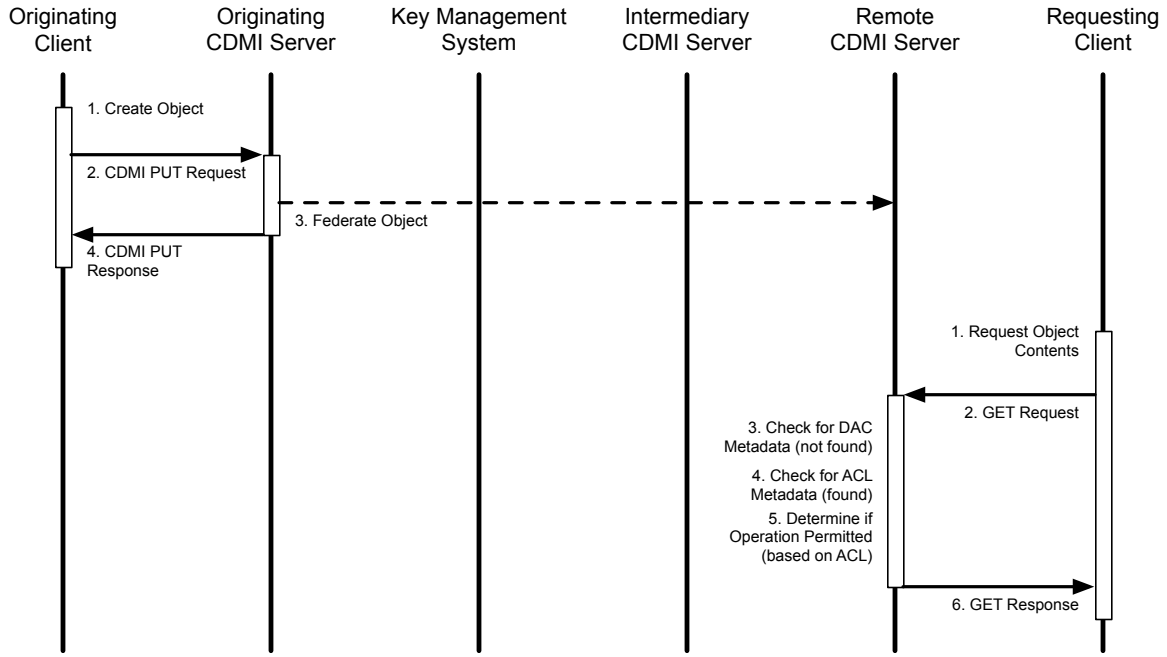


Fig. 14: Non-delegated (ACL-based) access control data flow

When an access control decision needs to be made by a third party (such as by the originating CDMI server in Fig. 14), access control is delegated. When `cdmi_dac_uri` and `cdmi_dac_certificate` object metadata is present, as specified in clause 16.1, Delegated Access Control (DAC) shall be used.

An example of an object with DAC metadata is shown below:

```
{
  "objectType": "application/cdmi-object",
```

(continues on next page)

(continued from previous page)


```
  "objectName": "MyObject.txt",
  "capabilitiesURI": "/cdmi_capabilities/dataobject/",
  "objectID": "0000000800182ADB37303732323136662D343564622D3462",
  "mimetype": "text/plain",
  "metadata": {
    "cdmi_size": "33",
    "cdmi_ctime": "2017-04-05T11:01:25",
    "cdmi_atime": "2017-04-05T11:44:28",
    "cdmi_dac_uri": "https://cloud.example.com/dac/",
    "cdmi_dac_certificate": {
      "kty": "EC",
      "x": "goqhRgM4hyEh1p-fD1oU15QAgdKXsBZTQ_0B-IgSz6M",
      "y": "cd8RTm8uLTGblIzioAzv8dzIkM85c08o23eksJrDt2Y",
      "crv": "P-256"
    }
  },
  "valueTransferEncoding": "utf-8",
  "valueRange": "33",
  "value": "This is an unencrypted text file."
}
```


The process by which objects are federated between systems is outside the scope of access control delegation and involves how objects are replicated, synchronized, mirrored, or migrated between CDMI servers. These processes are typically under the control of policies or external policy management systems. Federation is typically performed by third-party systems that use CDMI features including notification, serialization, and the preservation of globally unique object identifiers, which forms the basis for client-transparent interoperability.

24.2 Delegated Access Control (DAC)

A cloud storage system may implement support for DAC, which is indicated by the presence of the cdmi_dac system-wide capability.

DAC enables requests for operations against an object to be allowed or denied by a third-party DAC provider, in addition to ACL access control. When required by object metadata, DAC access control verification shall be performed after ACL evaluation, but before ACL enforcement, as the DAC provider may overrule local ACL evaluation results. When an encrypted object is accessed, the DAC provider may provide the decryption key. The decryption key enables access to encrypted objects, even if the CDMI server cannot access the keys directly.

Clients often have different degrees to which they trust the CDMI server with which they are interacting. Table Table 124 describes the four ways that DAC shall interact with stored objects.

Table 124: Access Modes for DAC

Mode of Access	Degree of Trust
Client-side decryption	CDMI server is not trusted with keys or to make delegated access control decisions. 1. Client requests encrypted object from CDMI Server 2. Client receives ciphertext from the CDMI Server 3. Client is responsible for getting decryption keys out of band 4. Client verifies signatures (if present) 5. Client verifies correct object 6. Client decrypts object This mode of access does not use any functionality indicated by the `cdmi_dac capability` and is supported by all CDMI servers.
Client-side decryption with DAC	CDMI server is not trusted with keys and is used to establish an opaque channel of communication between the client and the DAC provider for key delivery. 1. Client requests encrypted object from the CDMI Server, and includes custom DAC headers specifying information required for secure delivery of decryption key 2. Client receives ciphertext from the CDMI Server, along with custom DAC header from the DAC provider for the decryption key 3. Client is extracts decryption key from DAC provider headers 4. Client verifies signatures (if present) 5. Client verifies correct object 6. Client decrypts object This mode of access requires the cdmi_dac capability but does not require encrypted object support. In this mode, data is exchanged between the client and the DAC provider using one or more “CDMI-DAC-” headers, as described in clause 24.4.

Continued on next page

Table 124 – continued from previous page

Mode of Access	Degree of Trust
Direct Client DAC	CDMI server is not trusted with keys, and client establishes channel of communication between the client and the DAC provider for key delivery. 1. Client requests encrypted object from CDMI Server 2. Client receives ciphertext from CDMI Server 3. Client sends DAC request directly to DAC Provider 4. Client receive DAC response directly from DAC Provider 5. Client verifies signatures (if present) 6. Client verifies correct object 7. Client decrypts object This mode of access requires the cdmi_dac capability but does not require encrypted object support.
Server-side decryption with DAC	CDMI server is trusted with keys and to delegate access control decisions. DAC message exchange is used to get the decryption keys to decrypt the contents of the object, and keys are not revealed to the client. 1. Client requests encrypted object from CDMI Server 2. CDMI server contacts the DAC Provider to determine access control decision and gets decryption keys, where the keys are not revealed to the client. 3. CDMI server verifies signatures (if present) 4. CDMI server verifies correct object 5. CDMI server decrypts object 6. Client receives plaintext This mode of access requires DAC and encrypted object support.
Plaintext objects with DAC	CDMI server is trusted with plaintext and to not bypass delegated access control decisions. 1. Client requests non-encrypted object from CDMI Server 2. CDMI server contacts DAC provider to determine access control decision 3. CDMI server verifies signatures (if present) 4. CDMI server verifies correct object 5. Client receives plaintext This mode of access requires DAC support.

The `cdmi_dac_uri` metadata item indicates where delegated access control requests shall be submitted, and the `cdmi_dac_certificate` metadata item indicates how securely communication with the delegated access control provider shall be established. Both of these metadata items shall be present for DAC to be enabled for a given object.

DAC requests are submitted to a DAC provider using two typical methods:

- **Direct** - The DAC request shall be submitted directly to the absolute URI specified in the `cdmi_dac_uri metadata` item. This approach requires the host specified in the URI to be accessible from the CDMI server, and for the CDMI server making the request to have sufficient permissions to PUT the DAC request to that location.
- **Indirect** - The DAC request shall be sent to the DAC provider using an indirect route. Indirect routing is useful when the `cdmi_dac_uri` does not specify a host. An example of indirect routing is when the `cdmi_dac_uri` contains a mailto URI; the Internet mail system is then responsible for delivering the DAC request.

In other cases, the certificate included with the DAC request (taken from the `cdmi_dac_certificate` metadata) may be used by intermediary CDMI servers to determine the further routing of the DAC request. For example, DAC requests using a E.U.-issued certificate can be forwarded to a different intermediary CDMI server to those requests using a U.S.-issued certificate. How certificate fields are used to determine routing is not defined in this International Standard.

Both direct and indirect routing may be synchronous or asynchronous. If a DAC response is not received within the CDMI server or client timeout windows, the client request may time out; however a subsequent request may be processed locally if the DAC response allows response caching. When the CDMI server times out while waiting for a DAC response, it shall return an HTTP status code of `504 Gateway Timeout`.

24.3 Delegated Access Control Message Exchange

When a client requests to access or modify an object containing DAC metadata on a CDMI server that supports DAC, the CDMI server shall create and send a DAC request as specified in clause 24.5. Upon receiving a DAC response as specified in clause 24.7, the CDMI server shall allow or deny the operation based on the contents of the response.

Figure Fig. 15 provides an example of access control delegation for a non-encrypted object. The black solid lines show indirect routing, and gray dashed lines show direct routing.

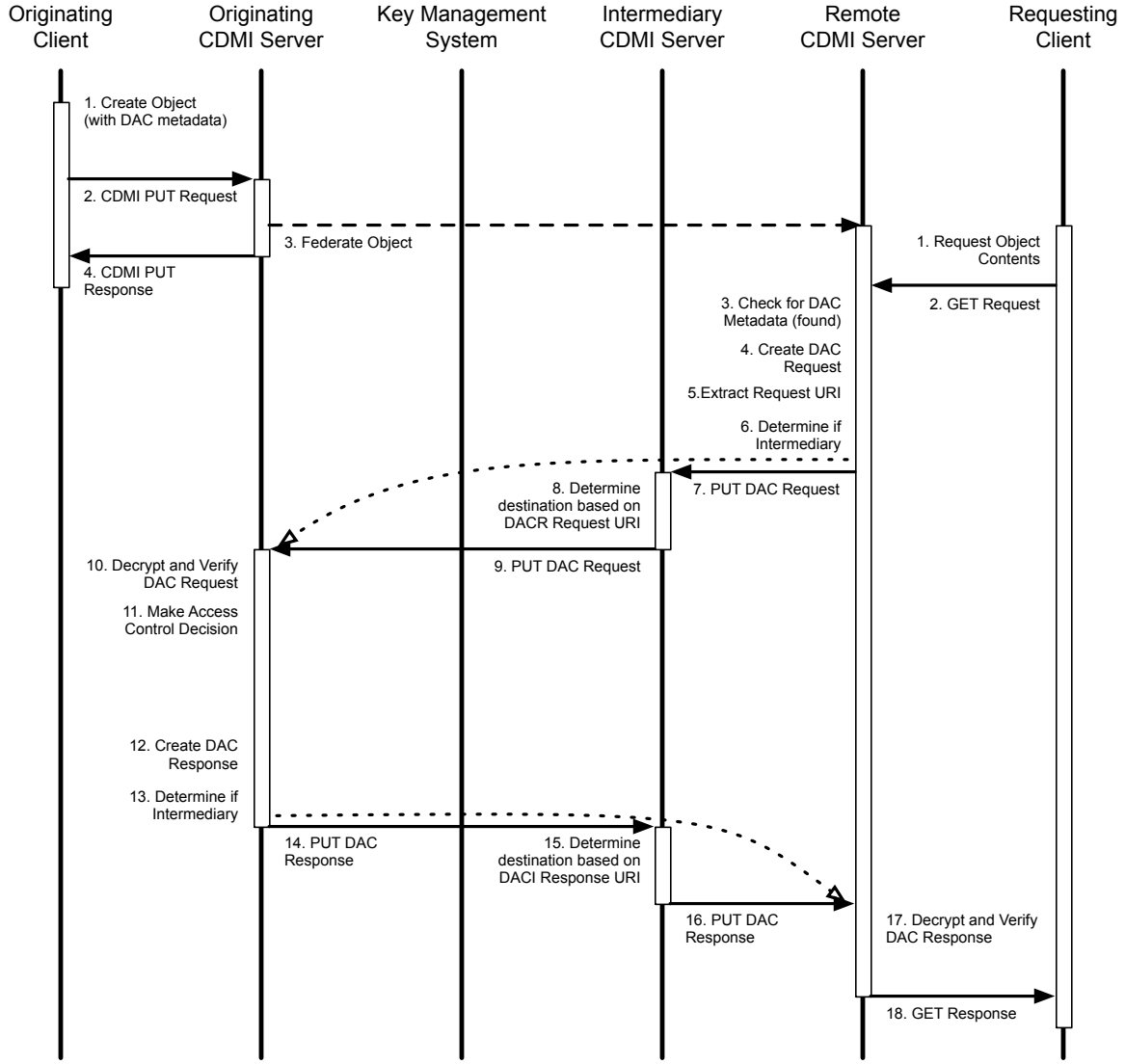


Fig. 15: Delegated access control data flow example for non-encrypted object

For non-encrypted objects, an originating client indicates that DAC is requested by including the DAC metadata items. It is important to emphasize that for non-encrypted objects, DAC cannot be guaranteed to be enforced, as when an object with DAC metadata is accessed from a CDMI server that does not support DAC; only ACL-based access control shall be evaluated.

Figure Fig. 16 provides a second example of access control delegation for an encrypted object. The black solid lines show indirect routing, and gray dashed lines show direct routing.

For encrypted objects, as access to the decryption keys are provided in the DAC response, the plaintext is inaccessible unless the CDMI server supports DAC.

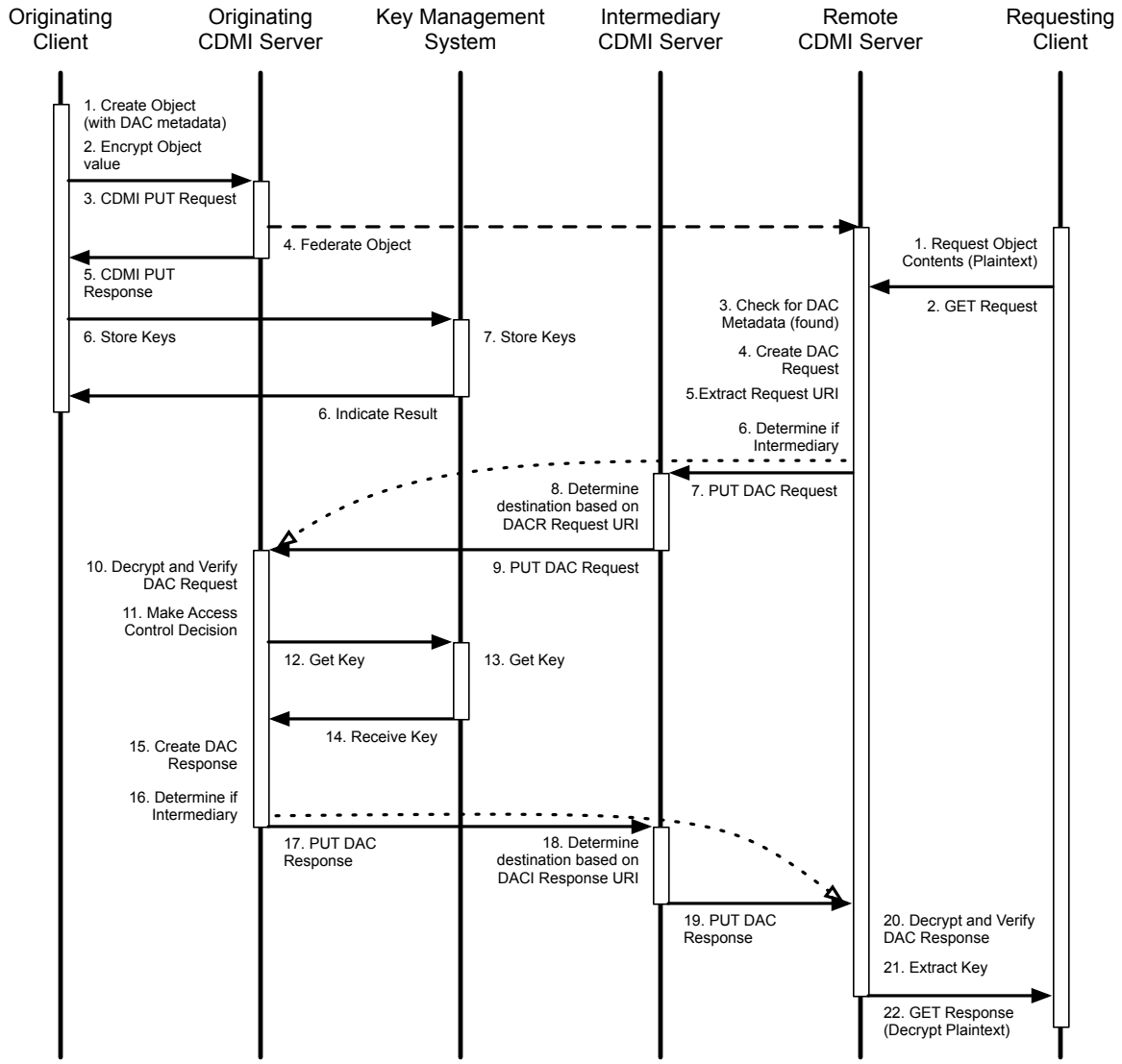


Fig. 16: Delegated access control data flow example for encrypted object

When the DAC provider processes the DAC request, if the operation is allowed and the key is requested by the CDMI server, the object key, if present, shall be obtained and sent back as part of the DAC response. Upon receiving the DAC response, the CDMI server shall extract the key to perform the client operation.

24.4 Client Header Passthrough

The Delegated Access Control extension provides facilities to allow client-provided HTTP request headers to be passed through to the DAC provider, and for the DAC provider to pass HTTP response headers back to the client. These headers are identified by the "`CDMI-DAC-`" prefix.

The contents and full names of these headers are not defined in this International Standard. However, it is anticipated that these headers shall be used to allow the client to provide additional information that may be required for the access control decision-making process, for audit purposes, or for secure key exchange.

For example, when an operation is allowed by a DAC provider, the object key may be encrypted using the public key from a client-provided certificate (verified by the DAC provider), which is included in a "`CDMI-DAC-`" request header, with the encrypted object key being sent back to the client in a "`CDMI-DAC-`" response header. In this scenario, the CDMI server cannot decrypt the ciphertext but can securely pass on the encrypted object key to the client. The client can then use its private key to decrypt the response header to get the object key, which can then be used to decrypt the object.

24.5 DAC Request

When a CDMI server that supports DAC needs to contact the DAC provider as specified in the DAC metadata, it shall construct a DAC request, as specified below:

3927
3928

Table 125: DAC Request

Field Name	Type	Description	Requirement
dac_request_version	JSON String	Indicates the version of the DAC request. This field shall be set to the value "1".	Mandatory
dac_request_id	JSON String	Contains a system-specified identifier that is used to match up the corresponding DAC response. This identifier shall be unique within the window that multiple DAC responses may be received.	Mandatory
server_identity	JSON Object	A JSON object, containing a JWE JWK which shall include a public key that is used to submit a DAC response, and should contains a X.509 certificate or certificate chain used to verify the identity of the CDMI server that is generating the DAC request. This ensures that only the CDMI Server that generated the DAC request can read the DAC response.	Mandatory
client_identity	JSON Object	A JSON object containing the following JSON entities: JSON String, "acl_name", containing the ACL name of the client requesting the operation. JSON Array, "acl_group", containing the ACL group(s) of the client requesting the operation.	Optional
acl_effective_mask	JSON String	A text or hexadecimal string representation of the ACE mask determined by ACL evaluation for the requested operation, as defined in clause %s.	Mandatory
client_headers	JSON Object	A JSON object containing a JSON string for each HTTP header in the operation request that starts with "CDMI-DAC-", where the JSON string name is the header name, and the JSON string value is the header value. These headers can be used for tunneling information from the client to the DAC provider.	Mandatory
cdmi_objectID	JSON String	Contains the object ID of the object the operation is performed against.	Mandatory
cdmi_encryption_key_id	JSON String	Contains the encryption key identifier (for example, a KMIP identifier) for the symmetric key that is used to encrypt and de-	Optional

3929 An example of a DAC request is shown below:


```
{
   "dac_request_version": "1",
   "dac_request_id": "037130fa-da72-44f0-8a31-62073263ac95",
   "server_identity": {
      "kty": "EC",
      "x": "joyfi05KEI3hcOhJeOfny_TWsZ9FFS1zUydFQhm3G78",
      "y": "Nsk3jX1ph0FH8APR2k0XSu6pDZYyF7f_Okplf7hZ_8k",
      "crv": "P-256"
   },
   "client_identity": {
      "acl_name": "anonymous",
      "acl_group": ["users"]
   },
   "acl_effective_mask": "READ_ALL",
   "client_headers": {
      "cdmi-dac-header1": "This is a test header"
   },
   "cdmi_objectID": "0000000800182ADB37303732323136662D343564622D3462",
   "cdmi_operation": "cdmi_read"
}
```

24.6 Packaged DAC Request

The above JSON (DAC request) shall be encrypted in JWE format, where the recipient is the public key of the DAC provider certificate (as specified in the DAC object `cdmi_dac_certificate` metadata), and is JWS signed using the private key of the CDMI server that corresponds to the server identity certificate included in the DAC request. The certificate of the DAC provider from the object is then attached:

Table 126: Packaged DAC Request

Field Name	Type	Description	Requirement
dac_request	JSON Object	JOSE encrypted and signed request	Mandatory
dac_request_dest_certificate	JSON Object	The cdmi_dac_certificate metadata value, which is used to indicate where the DAC request is being sent via indirect routing.	Mandatory
dac_request_dest_uri	JSON String	The `cdmi_dac_uri` metadata value, which is used to indicate where the DAC request is being sent via direct routing, or used to indicate the first location when being sent via indirect routing.	Mandatory

An example of a packaged DAC request is shown below[1]:


```
{
   "dac_request": {
      "protected":
         "eyJqd2siOiJ7XCJrdHlcIjpcIkVDXCIsXCJ4XCI6XCJqb3lmaTA1S0VJM2hjT2hK
          ZU9mbnlfVFdzWjlGRlMxelV5ZEZRaG0zRzc4XCIsXCJ5XCI6XCJOc2szalgxcGgw
          Rkg4QVBSMmswWFN1NnBEWll5RjdmX09rcGxmN2haXzhrXCIsXCJjcnZcIjpcIlAt
          MjU2XCJ9IiwiYWxnIjoiRVMyNTYifQ",
      "payload":
         "eyJwcm90ZWN0ZWQiOiJleUpoYkdjaU9pSkZRMFJJTFVWVElpd2laVzVqSWpvaVFU
          STFOa2REVFNJc0ltVndheUk2ZXlKcmRIa2lPaUpGUXlJc0luZ2lPaUpuUkZOek1F
          cFRXbU5VVVRsWGVGWXRiRXhSVTJ4elFsY3lXazFvTm1kb1JrcDJTVmt4TWt4d1dW
          TlJJaXdpZVNJNklsTkZNV1pXWkVkalZtdGtPVVZCVmpaVGMyeE9NVzQyUkdsdlpV
          dHVZV3BLWmpsdWVFOVljRlpoYmtFaUxDSmpjbllpT2lKUUxUSTFOaUo5ZlEiLCJl
          bmNyeXB0ZWRfa2V5IjoiIiwiaXYiOiJLRDlGRlBOcFh2cWNIYTdIIiwiY2lwaGVy
          dGV4dCI6Im42NlpmUzBXRmhjN3ZzT3Rnc1o5SXJtWU5paDI4RDVzT1psTk96dEdO
          TW5hakFRSGZTMGozcUhrMUxPME9IbFBYMnVfYXVWcVN2aDF2Z1IxSFlnOEl3TmFq
          TFZfS29ZMndGXzlkaDRtWFJlVXA4R1hpbm05MFE0ZWZmY1BLRm1IcEo0dE94TTVS
          VjlLN2VvdWNxSkxzczJKbHc1ZUJhOVQ5WjFyS1pvQmIxVURSLVVmRW91QlNZRFA3
          NUl1SEFRSWU4UW5qOW04QjFHb18tNTFPNndKb2d6cHh5Ulhpd3g2SWdoYlhSYmNX
          MWQ5bVRtZkR3UFBoSE4zTUp1UGUxbVBpelNLWnJ3NWNQM21NZmhKWmNoT3gyZkZt
          Q3NMME5zSkphQWo3WEs0elFiMGVBd0RSS1BzeTJ6MnZCZzFQT1lhUHppOVphNjRK
          RHgyZ3hWRTA2Y0xERGx3TXY4dW9CbFU1TVdyZF9YRGdScUZsSFl1T19aZEtxQkRp
          MVQ1SW5HeDc2YzdCcmVObzFIbnVqV200M0FsanpPRmIyTHBhdU5PQnlETl9oVXFi
          WGRISTZOWnZBUDU0MzVteHZDRi1SYUpMZGxFUENNeGhneXNFdy1oRGxoQmtFYUpf
          U0JtZUZtem5ITGFkZUNDYzI3cWNuOUlZVlZBMHZMZVY2N2xzbnZMY3VyOHI0OFlt
          SXRmZGNZbFVOLTh2c0xhSlZzbHhMSzc0VjdjdWNhbFNubWJvYktWTVV6TnZuU29K
          NHpldXBYZzItbl92WnIwbkZlSUFWelIxZmJvUVA0c1F4bXNSUWJNY2d4bmpSM21E
          eTJsQzY5dFN1TDJGYm1qUnZiYWM3XzFRa01CIiwidGFnIjoiNWlRcGVTdTlfb00y
          X2UtSTM3NjJpQSJ9",
      "signature":
         "rGz9Cku3csTIJ_p3qmHzUrPSLb1ZSD3ZlfaJDw0F-dNmJs6sgzizFC_jf5VgDVuo
          GT-wH2b2zVuP_O1HDcKPDQ"
   },
   "dac_request_dest_certificate": {
      "kty": "EC",
      "x": "goqhRgM4hyEh1p-fD1oU15QAgdKXsBZTQ_0B-IgSz6M",
      "y": "cd8RTm8uLTGblIzioAzv8dzIkM85c08o23eksJrDt2Y",
      "crv": "P-256"
   },
   "dac_request_dest_uri": "https://cloud.example.com/dac/"
}
```


[1] Decrypt with `"d": "NnU0IEyV4JSyLoKwIzKN1FAxDvL6qqawAHlPkpwBMSY"`.

Once created, the packaged DAC request shall be submitted using the DAC request URI specified in the DAC object metadata, for example, as an HTTP PUT operation of type "`application/json`", or via an SMTP email. The `dac_request_dest_certificate` and `dac_request_dest_uri` may be used to route the request through intermediary hops, as needed.

24.7 DAC Response

When a DAC provider receives a DAC request, it shall decrypt the request using its private key, verify the signature of the CDMI server, and shall evaluate the request. Based on the information provided, the DAC provider shall allow or deny operations by modifying or replacing the ACL mask that was initially determined by the CDMI server.

To indicate the result of the DAC request to the requesting CDMI server, the DAC provider shall construct a DAC response, as specified below:

Table 127: DAC Response

Field Name	Type	Description	Requirement
dac_response_version	JSON String	Indicates the version of the DAC response. This field shall be set to the value "1".	Mandatory
dac_response_id	JSON String	Contains the system-specified identifier specified in the corresponding dac_request_id.	Mandatory
dac_applied_mask	JSON String	A text or hexadecimal string representation of the ACE mask that shall be used, as defined in clause %s.	Mandatory
dac_object_key	JSON Object	The key for the object in JWK format (See RFC 7517). This key is only disclosed when cdmi_enc_key_id is included in the DAC request and the DAC provider allows access.	Optional
dac_response_headers	JSON Object	A series of headers that start with "CDMI-DAC-" to be returned to the client. These headers can be used to pass information from the DAC provider back to the client.	Optional
dac_key_cache_expiry	JSON String	The complete date/time when the object key is no longer to be cached, specified in ISO 8601 date/time format. If this field is not included, the key shall not be cached.	Optional
dac_response_cache_expiry	JSON String	The complete date/time when the DAC response is no longer to be cached, specified in ISO 8601 date/time format. If this field is not included, the response shall not be cached.	Optional
dac_redirect_objectid	JSON String	Indicates an alternate CDMI Object ID used to access the requested object. If present, the CDMI server shall send an HTTP Redirect to the client.	Optional
dac_audit_uri	JSON String	Indicates a URI to a CDMI queue where audit logging messages associated with the operations shall be submitted. When present, audit logging messages shall be generated for receiving the response, performing the operation, and determining when to purge the key (see clause %s).	Optional

An example of a DAC response is shown below:

```
{
   "dac_response_version": "1",
   "dac_response_id": "037130fa-da72-44f0-8a31-62073263ac95",
   "dac_applied_mask": "ALL_PERMS",
   "dac_response_headers": {
      "CDMI-DAC-AuthInfo": "No key requested."
   },
   "dac_response_cache_expiry": "2017-04-06T15:06:01.554Z"
}
```

24.8 Packaged DAC Response

The above JSON (DAC response) shall be encrypted in JWE format where the recipient is the public key of the CDMI server certificate (as specified in the DAC request), and is JWS-signed using the private key of the DAC provider that corresponds to the DAC provider identity certificate associated with the object (`cdmi_dac_certificate`), or with a different signing, included in a jku/jwk/x5u or x5c JOSE header to allow retrieval of the public signing verification key.

The certificate of the CDMI server is then attached:

Table 128: Packaged DAC Response

Field Name	Type	Description	Requirement
dac_response	JSON Object	JOSE encrypted and signed response	Mandatory
dac_response_dest_certificate	JSON Object	The contents of the DAC request `server_identity` field.	Mandatory
dac_response_dest_uri	JSON String	The contents of the DAC request `dac_response_uri` field, if present	Optional

An example of a packaged DAC response is shown below[2]:


```
{
   "dac_response": {
      "protected":
         "eyJqd2siOiJ7XCJrdHlcIjpcIkVDXCIsXCJ4XCI6XCJnb3FoUmdNNGh5RWgxcC1m
          RDFvVTE1UUFnZEtYc0JaVFFfMEItSWdTejZNXCIsXCJ5XCI6XCJjZDhSVG04dUxU
          R2JsSXppb0F6djhkeklrTTg1YzA4bzIzZWtzSnJEdDJZXCIsXCJjcnZcIjpcIlAt
          MjU2XCJ9IiwiYWxnIjoiRVMyNTYifQ",
      "payload":
         "eyJwcm90ZWN0ZWQiOiJleUpoYkdjaU9pSkZRMFJJTFVWVElpd2laVzVqSWpvaVFU
          STFOa2REVFNJc0ltVndheUk2ZXlKcmRIa2lPaUpGUXlJc0luZ2lPaUpNVUVReWRX
          WmlkMUpmT0hoU2FWRlRNMWN3YUZSbU5tWnlXWEZDU0hWYU4xQTVUbEEzVFdaVFEy
          MDRJaXdpZVNJNklqWmhiMWgxUzJFeVVqZHNTMW93YlU5UlJUQmFlV0pQU2pKWlYy
          bzNOM1l3Wm5GWU1ESnBiRE5EVUVVaUxDSmpjbllpT2lKUUxUSTFOaUo5ZlEiLCJl
          bmNyeXB0ZWRfa2V5IjoiIiwiaXYiOiJYMFhTUDNZVTNBUkJwQ1NlIiwiY2lwaGVy
          dGV4dCI6Ik5DQXE1dnBCeUVaVERJcHlWem5Gemxtbm1JU09sVk5uNGpSUUtKWjB5
          c0s0dzZPcDYtNE94cGtvVVY5WUFvbDhmdUVOeFFMdjFBQUpDWXB3M0ZFelRrMEpG
          VmU1NWE0UlNIVkhnSlJmMEhiWjlxbk5aOHY0d1JUaXBGS0RsakpvLUhXOG82bzlM
          czV2YmRVTGJPRk9Db3RTTGZuekdSQ3lMV3Z2TUZaS3BHXzM1b21PeFpNcW1oN2Ro
          c3IxMmF6cHdkSnJKX084TTFkVHdDaWZxeURlWWFpNGM4M3U4TUhieDdETldRWkhH
          QnIzT1J0bDhaWGJTQW9OQ09fVWRpdU8zWXZmWmNiWU51TTY2UXBZbDFobENSaDJO
          eEZtLW12VUR0a1VoaXR5cTdyZ3BSbWZoYndKNklCaGdpdyIsInRhZyI6Ijh3YWx6
          T0Q4U3hWTC1STXY3OXlTZGcifQ",
      "signature":
         "8-09XlWUUDsXXqoEh5EKIAYEOTR-vtAYqauW1aNfdv2Io9B4RCuALl3zi7i27vbo
          TYvHxnFa7K6HJPygsAVn5g "
   },
   "dac_response_dest_certificate": {
      "kty": "EC",
      "x": "joyfi05KEI3hcOhJeOfny_TWsZ9FFS1zUydFQhm3G78",
      "y": "Nsk3jX1ph0FH8APR2k0XSu6pDZYyF7f_Okplf7hZ_8k",
      "crv": "P-256"
   }
}
```


Once created, the packaged DAC response shall be returned as the response to the HTTPS/HTTP request, or submitted using the DAC response URI specified in the DAC request, for example, as an HTTP PUT operation or via an SMTP email. The `dac_response_dest_certificate` and `dac_response_dest_uri` may also be used to route the request through intermediary hops if needed, as determined by the routing system, which is out of scope of this standard.

When the CDMI server receives a packaged DAC response message, it shall decrypt it using its private key and shall verify the signature. If the decryption and signature verification are successful, the CDMI server shall use the provided `dac_applied_mask` in place of the ACL computed mask.

[2] Decrypt with “d”: “huCoV1iC24rZ3uF5q-1HHlGb2UcC6Ue9oNezEQNzUB8”

If the CDMI server supports key or DAC response caching, cache expiry values shall be honored. Cached responses and keys can only be used for identical client operations, where the client identity, objectID, operation, and “CDMI-DAC-” request headers are identical. Otherwise, the cached response shall be expired. If an audit URI is present in the cached response, audit messages shall also be generated for all operations allowed using the cached response.

The CDMI server shall also implement audit logging when specified in the DAC response. If the CDMI server does not support audit logging and it is required by a DAC response, the operation shall be denied.

If a `dac_redirect_objectID` field is returned in the DAC response, the CDMI server shall return an HTTP redirect to the specified Object ID. This redirect allows a DAC provider to create a client-operation-specific instance of the object that is encrypted with a single-use key.

24.9 Error Handling

In the following scenarios, the following HTTP response codes shall be returned to a client:

- When a DAC response denies the requested operation, an HTTP status code of `403 Forbidden` shall be returned to the client along with any `dac_response_headers` included in the response.
- When a DAC response includes a `dac_redirect_objectID`, an HTTP status code of `302 Found` shall be returned to the client along with any `dac_response_headers` included in the response.
- When a DAC request to access or modify an encrypted object is allowed, but the key is not included in the DAC response, an HTTP status code of `401 Unauthorized` shall be returned to the client along with any `dac_response_headers` included in the response.
- When a DAC request to access or modify an encrypted object is allowed, but cannot be performed due to lack of support for an encryption algorithm, signing algorithm, or key type, an HTTP status code of `501 Not Implemented` shall be returned along with any `dac_response_headers` included in the response.
- When a DAC request times out, an HTTP status code of `500 Internal Server Error` shall be returned to the client.
- When a DAC request cannot be sent or routed because the DAC metadata is not supported or valid, an HTTP status code of `501 Not Implemented` shall be returned to the client.
- When a DAC request cannot be sent or routed because an upstream system is unavailable, an HTTP status code of `500 Internal Server Error` shall be returned to the client.

24.10 Examples

The following examples illustrate the primary ways that DAC requests are performed.

EXAMPLE 1: GET ciphertext of encrypted object with delegated access control

The following CDMI operation is performed against an encrypted CDMI object with delegated access control metadata:

```
GET /MyContainer/MyEncryptedObject.txt HTTP/1.1
Host: cloud.example.com
Accept: application/cms, application/jose+json
```

The CDMI server verifies local access controls and determines that the request can proceed. The following DAC request is generated:

```
{
   "dac_request_version": "1",
   "dac_request_id": "5b801b19-479e-446d-882a-8483f7c4905c",
   "server_identity": {
      "kty": "EC",
      "x": "joyfi05KEI3hcOhJeOfny_TWsZ9FFS1zUydFQhm3G78",
      "y": "Nsk3jX1ph0FH8APR2k0XSu6pDZYyF7f_Okplf7hZ_8k",
      "crv": "P-256"
   },
   "client_identity": {
      "acl_name": "anonymous",
      "acl_group": ["guest"]
   },
   "acl_effective_mask": "READ_ALL",
   "client_headers": {},
   "cdmi_objectID": "0000000800182F9E64313363323731622D363536662D3465",
   "cdmi_operation": "cdmi_read"
}
```

This request is first JWE encrypted with the key in cdmi_dac_certificate. The result is JWS signed, using either the key in server_identity, or a different key embedded in the JWS header.

The DAC provider verifies, decryptes and processes the request and returns the following DAC response:

```
{
   "dac_response_version": "1",
   "dac_response_id": "5b801b19-479e-446d-882a-8483f7c4905c",
      "dac_applied_mask": "ALL_PERMS",
   "dac_response_headers": {
      "CDMI-DAC-AuthInfo": "No key requested."
   }
}
```

The `CDMI-DAC-AuthInfo` indicates a custom header.

Since the operation is allowed by the DAC provider, the following response is sent:

```
HTTP/1.1 200 OK
Content-Type: application/jose+json
Content-Length: 290
CDMI-DAC-AuthInfo: No key requested.

<JOSE+JSON Encrypted Object>
```

EXAMPLE 2: GET ciphertext of encrypted object with passthrough key access

The following CDMI operation is performed against an encrypted CDMI object with delegated access control metadata:

```
GET /MyContainer/MyEncryptedObject.txt HTTP/1.1
Host: cloud.example.com
Accept: application/cms, application/jose+json
Authorization: Basic am9lOnBhc3N3b3Jk
CDMI-DAC-N: <vendor-specific header that indicates key passthrough>
```

The CDMI server verifies local access controls and determines that the request can proceed. The following DAC request is generated. The CDMI-DAC-N is a custom header that indicates that the client wants to obtain the object decryption key via header pass-through.

To demonstrate the power of such custom headers: the CDMI-DAC-N request header could contain a cell phone number. The matching response header would then contain a password-based encryption of the object key, while the password will be delivered via a message to the cell phone. It is up to the vendor to come up with and implement such mechanisms.

```
{
   "dac_request_version": "1",
   "dac_request_id": "77b54650-183f-4053-8512-be08f7c6c50e",
   "server_identity": {
      "kty": "EC",
      "x": "joyfi05KEI3hcOhJeOfny_TWsZ9FFS1zUydFQhm3G78",
      "y": "Nsk3jX1ph0FH8APR2k0XSu6pDZYyF7f_Okplf7hZ_8k",
      "crv": "P-256"
   },
   "client_identity": {
      "acl_name": "joe",
      "acl_group": ["users"]
   },
   "acl_effective_mask": "READ_ALL",
   "client_headers": {
      "CDMI-DAC-N": "<copy from headers>"
   },
   "cdmi_objectID": "0000000800182F9E64313363323731622D363536662D3465",
   "cdmi_operation": "cdmi_read"
}
```

This request is first JWE encrypted with the key in cdmi_dac_certificate. The result is JWS signed, either using the key in server_identity, or a different key embedded in the JWS header. Replication of these encrypted messages is not useful and will be skipped.

The DAC provider processes the request, obtains the object decryption key and embeds it as a dac_response_header, then returns the following DAC response:

```
{
   "dac_response_version": "1",
   "dac_response_id": "5b801b19-479e-446d-882a-8483f7c4905c",
   "dac_applied_mask": "ALL_PERMS",
   "dac_response_headers": {
      "CDMI-DAC-AuthInfo": "Key successfully retrieved from keyserver."
      "CDMI-DAC-N": "<vendor-specific decryption key info>"
   }
}
```

Since the operation is allowed by the DAC provider, the following response is sent:

```
HTTP/1.1 200 OK
Content-Type: application/jose+json
Content-Length: 290
CDMI-DAC-AuthInfo: Key successfully retrieved from keyserver.
CDMI-DAC-N: <vendor-specific decryption key info>

<JOSE+JSON Encrypted Object>
```

The client can now parse the key in the CDMI-DAC-N header and use it to decrypt the ciphertext.

EXAMPLE 3: GET plaintext of encrypted object with delegated access control

The following CDMI operation is performed against an encrypted CDMI object with delegated access control metadata:

```
GET /MyContainer/MyEncryptedObject.txt HTTP/1.1
Host: cloud.example.com
Accept: */*
Authorization: Basic am9lOnBhc3N3b3Jk
```

The CDMI server verifies local access controls and determines that the request can proceed. The following DAC request is generated:

```
{
   "dac_request_version": "1",
   "dac_request_id": "b79d7619-1bbd-45a1-b2d3-5753f7fc5155",
   "server_identity": {
      "kty": "EC",
      "x": "joyfi05KEI3hcOhJeOfny_TWsZ9FFS1zUydFQhm3G78",
      "y": "Nsk3jX1ph0FH8APR2k0XSu6pDZYyF7f_Okplf7hZ_8k",
      "crv": "P-256"
   },
   "client_identity": {
      "acl_name": "joe",
      "acl_group": ["users"]
   },
   "acl_effective_mask": "READ_ALL",
   "client_headers": {},
   "cdmi_objectID": "0000000800182F9E64313363323731622D363536662D3465",
   "cdmi_operation": "cdmi_read",
   "cdmi_enc_key_id": "0000000800182F9E64313363323731622D363536662D3465"
}
```


The DAC provider processes the request, obtains the object decryption key and returns the following DAC response:


```
{
   "dac_response_version": "1",
   "dac_response_id": "b79d7619-1bbd-45a1-b2d3-5753f7fc5155",
   "dac_applied_mask": "ALL_PERMS",
   "dac_object_key": {
      "kty": "oct",
      "kid": "0000000800182F9E64313363323731622D363536662D3465",
      "use": "enc",
      "alg": "dir",
      "k": "vBX811eh8ydyI08by7L13kZKNmfRHTAMZa5vJqMCHQU"
   },
   "dac_response_headers": {
      "CDMI-DAC-AuthInfo": "Key successfully obtained from KMS."
   }
   "dac_key_cache_expiry": "2017-04-05T14:58:58Z",
   "dac_response_cache_expiry": "2017-04-05T14:58:58Z"
}
```


Since the operation is allowed by the DAC provider and the key is provided, the object is decrypted by the CDMI server and the following response is sent:

```
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 252

<Decrypted contents of Encrypted Value>
```

EXAMPLE 4: RSA Example

In this example, there are two hospitals (A and B), that both have CDMI servers, and federate objects between them. At some point, the following encrypted object has been made at hospital A. It containts a cdmi_dac_certificate and cdmi_dac_uri that indicate how access can be requested at hospital A. The cerificate contains a 2048-bit RSA encryption key, with a matching X.509 certification chain that can be used to verify the certificate.


```
{
   "objectType": "application/cdmi-object",
   "objectName": "MyEncryptedObject.txt",
   "capabilitiesURI": "/cdmi_capabilities/dataobject/",
   "objectID": "000000080018F34436313131393061372D613735302D3438",
   "mimetype": "application/jose+json",
   "metadata": {
      "cdmi_size": "306",
      "cdmi_dac_uri": "https://cdmi.hos-a.fr:9001/dac/",
      "cdmi_atime": "2017-04-06T14:06:34",
```


(continues on next page)

(continued from previous page)


```
      "cdmi_enc_key_id": "encryption_key_1",
      "cdmi_dac_certificate": {
         "kty": "RSA",
         "kid": "cdmi.hos-a.fr_encrypt_public",
         "key_ops": [
           "wrapKey",
           "unwrapKey",
           "encrypt",
           "decrypt"
         ],
         "n":
            "uL7ANgD80H5sNqo3nHzovPRxgncQLhz0oQvGMVvULCkrYXMaXZ5sNv7fT6UdMSZi
             T-e0sthapmEqrpeV9RKHsiF3COGl2YndUHixpEkHp8y1ggcH6iTzoBsgXMZ7OLW-
             mJ2RW3rodT7k-tcozYYsTSM5egMPQSAKgt0nMnPmdNRnEyA2_NJ8Y7lNkEXyja0Q
             JLstzkP8-cKS0BkEquLQEMbZVRM6U5uG69cj1i9OWvuRzPoaATKyt6Cc4f6PUu9L
             OyCBUAs9dXsRrt3B8H1qe7io7FAAcOpcUDKDnLFXS1THc37DK_zEyKZcMttjCvEl
             Ovt-cIaokdnxJeggv9AFGQ",
         "e": "AQAB",
         "x5c": [
           "MIIDMDCCAhigAwIBAgIBBDANBgkqhkiG9w0BAQsFADBCMQswCQYDVQQGEwJubDER
            MA8GA1UEChMIbGllc2RvbmsxDTALBgNVBAsTBGNkbWkxETAPBgNVBAMTCHJzYS1y
            b290MCAXDTE2MTAyNzEyNDUwMFoYDzk5OTkxMjMxMjM1OTU5WjA1MQswCQYDVQQG
            EwJmcjEOMAwGA1UEChMFaG9zLWExFjAUBgNVBAMTDWNkbWkuaG9zLWEuZnIwggEi
            MA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC4vsA2APzQfmw2qjecfOi89HGC
            dxAuHPShC8YxW9QsKSthcxpdnmw2/t9PpR0xJmJP57Sy2FqmYSqul5X1EoeyIXcI
            4aXZid1QeLGkSQenzLWCBwfqJPOgGyBcxns4tb6YnZFbeuh1PuT61yjNhixNIzl6
            Aw9BIAqC3Scyc+Z01GcTIDb80nxjuU2QRfKNrRAkuy3OQ/z5wpLQGQSq4tAQxtlV
            EzpTm4br1yPWL05a+5HM+hoBMrK3oJzh/o9S70s7IIFQCz11exGu3cHwfWp7uKjs
            UABw6lxQMoOcsVdLVMdzfsMr/MTIplwy22MK8SU6+35whqiR2fEl6CC/0AUZAgMB
            AAGjPDA6MAwGA1UdEwEB/wQCMAAwHQYDVR0OBBYEFBAIGICMR5H6KLKMlZAEhCCc
            KwE9MAsGA1UdDwQEAwIEMDANBgkqhkiG9w0BAQsFAAOCAQEAANYSSryUU6112pYM
            r83M3GWnjzu16B+4KgimZ8kbey94zNPdwmwQdSe0Xmg+1Otc6VUB40ouNnwK8efB
            aWBtXwCA7Nb715nTqo2+rn+X+A0mGrYaKkToPEe8ZYwDcOlOpNC9JFE+QgP9/CJa
            AaWrf95W+4kra2WnA4Bhqu2WWXnQkL47/nKcGVZgQAH+mVnxPaI0gELYdonXU/S2
            8HqxoyjpGL/vmyc46zUbxYsgx/jiE7J0fJVP6Yk/3dlNYCCpLtV8VmzFAQAeCcn8
            AWowFcd09a4SY09rn1MUv/rrvXpzflfn9j7PtRRFj2e/KhitmOH1zKDuYzREpUOu
            TDlPIQ==",
           "MIIDQDCCAiigAwIBAgIBATANBgkqhkiG9w0BAQsFADBCMQswCQYDVQQGEwJubDER
            MA8GA1UEChMIbGllc2RvbmsxDTALBgNVBAsTBGNkbWkxETAPBgNVBAMTCHJzYS1y
            b290MCAXDTE2MTAyNzEyNDQwMFoYDzk5OTkxMjMxMjM1OTU5WjBCMQswCQYDVQQG
            EwJubDERMA8GA1UEChMIbGllc2RvbmsxDTALBgNVBAsTBGNkbWkxETAPBgNVBAMT
            CHJzYS1yb290MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAsrUj46dx
            5ojlaZk7YtOL6e+Q6JoG7gVMaXkJn1Szlx9ND/8w4PeO1SQ2skukdOHAlQRmxdft
            zhccNTM5hmbcn8TAfWSYqQF1R7s78bVjtmat6AQP1vSgiyZ8Ak+iYZEq3c2zVyYQ
            HKKxWxmFZt1HT8/H/B3bXveXQcERKE+Tq66h8pqVcocQUtzRFsEYmv0bR1rghtoq
            H8nhB5xnebgVlXjApW+et2SE7r6Fjv1aAbGI89ouJ1gsMPeX56P8AUjacFtNKc44
            Obu6HRXY/jm6f2m1EUm84EUsJ+9b5+S2x4qPttJDfSCasWYYz4mFJ8MwmFiBGUwf
            geT2bUm6t7qqbQIDAQABoz8wPTAPBgNVHRMBAf8EBTADAQH/MB0GA1UdDgQWBBR+
            tEB2udkEXxX0k15GztF/4ol03jALBgNVHQ8EBAMCAQYwDQYJKoZIhvcNAQELBQAD
            ggEBAIjx1f9rJ2B+mDSA3L2GRhjrPRjfI6Un3Z51CeW9gO9PMQ5ws5pDJyB79dE/
            Q8Uf1e8pZyjchTsRa8GRdnKyndN2imayOVUvPoTd3/ZSmfkurcbj3I4VW8sjHP7C
            E8fmUS8Xprdpo2SxV7oneJC0vt5eyh8mgfJ/qSbwVaiXuH1Wxi6duAvdxddMXAxQ
            KPG1KKVM7CYfCdpX/HagCOHzcto+374zFqqnQ1Kx5rbgvxNSgm/PDDOMwP03+bbT
            R63KSK1VbdtLBuS4jgaPabwyxQz/FciwTu/HLOQn8TNqDWyoIbs+eQX2Mds2Apul
            8XH2+CakjBLMLL3Tlj2x+6tKR9o="
         ]
      },
      "cdmi_ctime": "2017-04-06T14:06:31"
   },
   "valueTransferEncoding": "json",
   "value": {
      "protected":
         "eyJraWQiOiJlbmNyeXB0aW9uX2tleV8xIiwiYWxnIjoiQTI1NktXIiwiY3R5Ijoi
          dGV4dC9wbGFpbiIsImVuYyI6IkEyNTZHQ00ifQ ",
      "encrypted_key":
         "329yyozEo3JPCpXGPKyI_fa5hhFH9dmfB7kulg1Q6NhoVAvdmDMclg",
      "iv": "9Gr5Hxzcs9hxPmPM",
      "ciphertext": "-sJkcHcdQUXChEBlZm7UZya1RR2_IcpRocC-BmQfAuA3",
```


(continues on next page)

(continued from previous page)

```
      "tag": "VIFJDcMdZngtpLWWDX8vFw"
   }
}
```

This encrypted object has been federated to the CDMI server at hospital B. Now, one of its clients wants to transparently access the plaintext of this object by performing the following operation:

```
GET /cdmi/MyContainer/MyEncryptedObject.txt HTTP/1.1
Host: cdmi.hos-b.us:9002
Accept: */*
```

The CDMI server at hospital B will look up the object and find out that it is an encrypted object with DAC information attached. As a result it will generate the following (plain) DAC request:

```
{
   "dac_request_version": "1",
   "dac_request_id": "73da04e1-2182-447e-8342-f4b9f06ec936",
   "server_identity": {
      "kty": "RSA",
      "kid": "cdmi.hos-b.us_encrypt_public",
      "key_ops": [
         "wrapKey",
         "unwrapKey",
         "encrypt",
         "decrypt"
      ],
      "n":
         "oQMQkY85UzwO7K6H0QQNfAiRMN3ZfhK0aXEKx7YwvrCU9IKOquZ1OYZ9Cv8556_8
          E8yZm02JDWOBOaSsGHU835jvXf12f4MywKGWj5FtIGL-j9kXF6SWq3zuLVYlXpMI
          KsJngHMVFca_-ZhZ2vLsrnDR1aCNEC48gR26ewp6WX1ptnSc1W4x3Mj-ONMVzxVE
          7XNlwYysTgDtonmTQD-YG6_KhhAPx0YowMbUWv_cMQvXsi7MMDyZn6fxfq4zQmQ2
          V5RtUy5msd6K3beDzS4LmZhsJmjU7YnhOj0pZby4Zckm43npjXPAuwPhzK2OW7qb
          fkv0qm4rsFWUcuNh8lBsDw",
      "e": "AQAB",
      "x5c": [
         "MIIDMDCCAhigAwIBAgIBBTANBgkqhkiG9w0BAQsFADBCMQswCQYDVQQGEwJubDER
          MA8GA1UEChMIbGllc2RvbmsxDTALBgNVBAsTBGNkbWkxETAPBgNVBAMTCHJzYS1y
          b290MCAXDTE2MTAyNzEyNDUwMFoYDzk5OTkxMjMxMjM1OTU5WjA1MQswCQYDVQQG
          EwJ1czEOMAwGA1UEChMFaG9zLWIxFjAUBgNVBAMTDWNkbWkuaG9zLWIudXMwggEi
          MA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQChAxCRjzlTPA7srofRBA18CJEw
          3dl+ErRpcQrHtjC+sJT0go6q5nU5hn0K\/znnr\/wTzJmbTYkNY4E5pKwYdTzfmO
          9d\/XZ\/gzLAoZaPkW0gYv6P2RcXpJarfO4tViVekwgqwmeAcxUVxr\/5mFna8uy
          ucNHVoI0QLjyBHbp7CnpZfWm2dJzVbjHcyP440xXPFUTtc2XBjKxOAO2ieZNAP5g
          br8qGEA\/HRijAxtRa\/9wxC9eyLswwPJmfp\/F+rjNCZDZXlG1TLmax3ordt4PN
          LguZmGwmaNTtieE6PSllvLhlySbjeemNc8C7A+HMrY5bupt+S\/SqbiuwVZRy42H
          yUGwPAgMBAAGjPDA6MAwGA1UdEwEB\/wQCMAAwHQYDVR0OBBYEFH7NJvMIfTQtZn
          nyiIdLNkjCgwSIMAsGA1UdDwQEAwIEMDANBgkqhkiG9w0BAQsFAAOCAQEAdiADIv
          0v09SUDcPL+BKysvchn\/Sgx5KBu7n9KFwE31Dhx2zvT6ruL8kXdekPH9cfrDafW
          6I\/vnbzAVj02i5pM2cHayj13fTOWSVwpcQuvkoIF9eVIWONkemMMf7M7jpTw07z
          7S2T5usaDmMNpqj8y5pRpQo3PnBVxpEZJ0XaSdfuiHtVLDq8gDZCq6Hc2tt7JM3W
          njnQgs+1lSGRuqWOcpmVONIoqvhiolNDZV35Z7puRwqck1N2f1qyHHGBWXfCN9U4
          ci6q1BnWBIFV+hURge8NSbpqawolaNueUbTcKjN3JsMC4ZxhMF9rN3uuPn+UAYkA
          yQkcSmGSMM07wcAkMg==",
         "MIIDQDCCAiigAwIBAgIBATANBgkqhkiG9w0BAQsFADBCMQswCQYDVQQGEwJubDER
          MA8GA1UEChMIbGllc2RvbmsxDTALBgNVBAsTBGNkbWkxETAPBgNVBAMTCHJzYS1y
          b290MCAXDTE2MTAyNzEyNDQwMFoYDzk5OTkxMjMxMjM1OTU5WjBCMQswCQYDVQQG
          EwJubDERMA8GA1UEChMIbGllc2RvbmsxDTALBgNVBAsTBGNkbWkxETAPBgNVBAMT
          CHJzYS1yb290MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAsrUj46dx
          5ojlaZk7YtOL6e+Q6JoG7gVMaXkJn1Szlx9ND\/8w4PeO1SQ2skukdOHAlQRmxdf
          tzhccNTM5hmbcn8TAfWSYqQF1R7s78bVjtmat6AQP1vSgiyZ8Ak+iYZEq3c2zVyY
          QHKKxWxmFZt1HT8\/H\/B3bXveXQcERKE+Tq66h8pqVcocQUtzRFsEYmv0bR1rgh
          toqH8nhB5xnebgVlXjApW+et2SE7r6Fjv1aAbGI89ouJ1gsMPeX56P8AUjacFtNK
          c44Obu6HRXY\/jm6f2m1EUm84EUsJ+9b5+S2x4qPttJDfSCasWYYz4mFJ8MwmFiB
          GUwfgeT2bUm6t7qqbQIDAQABoz8wPTAPBgNVHRMBAf8EBTADAQH\/MB0GA1UdDgQ
          WBBR+tEB2udkEXxX0k15GztF\/4ol03jALBgNVHQ8EBAMCAQYwDQYJKoZIhvcNAQ
          ELBQADggEBAIjx1f9rJ2B+mDSA3L2GRhjrPRjfI6Un3Z51CeW9gO9PMQ5ws5pDJy
          B79dE\/Q8Uf1e8pZyjchTsRa8GRdnKyndN2imayOVUvPoTd3\/ZSmfkurcbj3I4V
          W8sjHP7CE8fmUS8Xprdpo2SxV7oneJC0vt5eyh8mgfJ\/qSbwVaiXuH1Wxi6duAv
          dxddMXAxQKPG1KKVM7CYfCdpX\/HagCOHzcto+374zFqqnQ1Kx5rbgvxNSgm\/PD
```

(continues on next page)

(continued from previous page)


```
            DOMwP03+bbTR63KSK1VbdtLBuS4jgaPabwyxQz\/FciwTu\/HLOQn8TNqDWyoIbs
            +eQX2Mds2Apul8XH2+CakjBLMLL3Tlj2x+6tKR9o="
         ]
      },
      "client_identity": {
         "acl_name": "anonymous",
         "acl_group": ["guests"]
      },
      "acl_effective_mask": "READ_ALL",
      "client_headers": {},
      "cdmi_objectID": "000000080018F34436313131393061372D613735302D3438",
      "cdmi_operation": "cdmi_read",
      "cdmi_enc_key_id": "encryption_key_1"
}
```


This plain DAC request will be JWE encrypted using the key found in the object's `cdmi_dac_certificate` (key id 'cdmi.hos-a.fr_encrypt_public'). Then it will be JWS signed using hospital B's private signing key. Since this signing key is not equal to the encryption key (in `server_identity`) it is embedded in the JOSE protected header of the JWS (note: Base64 decode of the protected header reveals the signing key; Base64 decode of the payload reveals the JWE.)


```
{
   "dac_request": {
      "protected":
         "eyJraWQiOiJjZG1pLmhvcy1iLnVzX3NpZ25fcHJpdmF0ZSIsImp3ayI6IntcImt0
          eVwiOlwiUlNBXCIsXCJraWRcIjpcImNkbWkuaG9zLWIudXNfc2lnbl9wcml2YXRl
          XCIsXCJrZXlfb3BzXCI6W1widmVyaWZ5XCIsXCJzaWduXCJdLFwiblwiOlwicEVR
          aFFUMVF6QmdrV2RiVW56eVkwbkZmWjRVYXJnbFVPcGFxeGlXYXk5cGhnQ0x6Tmtj
          RHZ4eVdIdHFRSWE0ZHpvMDVaXzBiOXhNTE1rYUI5MTJheV83M1llOZHpmMWVfaUVX
          Mi1OdVB4MHVSaFV3SzQ4WUo2MFlwVTdpN2ZpQWNKeVJoU1dlWGtnQXQyRndUYnkt
          SjlsNW9DV1dZemRfc0U3a2NMSkc0QmkwSEtQbVhrUEVwbXpOamhsU0VsdnlodHFL
          djRERG1JRk1JTDNrUGJueGNfX0RwenAyaVVpdGhvUFhpY1pJQXMtUDIybGRGMkRE
          X0tzbW9SU3RQR2NuTEVYbWpKcXhoRU13Qm5UZE14TjdQNnh6bk5iQVNtdXNnR21F
          XzJXdVUyS09yLVBtYm5wTnNLcm14SHRhT2trc2pZdjFyVGhzRmkxNUZmSVQyQ1dU
          MnVRXCIsXCJlXCI6XCJBUUFCXCIsXCJ4NWNcIjpbXCJNSUlETURDQ0FoaWdBd0lC
          QWdJQkF6QU5CZ2txaGtpRzl3MEJBUXNGQURCQ01Rc3dDUVlEVlFRR0V3SnViREVS
          TUE4R0ExVUVDaE1JYkdsbGMyUnZibXN4RFRBTEJnTlZCQXNUQkdOa2JXa3hFVEFQ
          QmdOVkJBTVRDSEp6WVMxeWIyOTBNQ0FYRFRFMk1UQXlOekV5TkRVd01Gb1lEems1
          T1RreE1qTXhNak0xT1RVNVdqQTFNUXN3Q1FZRFZRUUdFd0oxY3pFT01Bd0dBMVVF
          Q2hNRmFHOXpMV0l4RmpBVUJnTlZCQU1URFdOa2JXa3VhRzl6TFdJdWRYTXdnZ0Vp
          TUEwR0NTcUdTSWIzRFFFQkFRVUFBNElCRHdBd2dnRUtBb0lCQVFDa1JDRkJQVkRN
          R0NSWjF0U2ZQSmpTY1Y5bmhScXVDVlE2bHFyR0packwybUdBSXZNMlJ3Ty9ISlll
          MnBBaHJoM09qVGxuL1J2M0V3c2lSb0gzWFpyTC92WmcxM04vVjcrSVJiYjQyNC9I
          UzVHRlRBcmp4Z25yUmlsVHVMdCtJQnduSkdGSlo1ZVNBQzNZWEJOdkw0bjJYbWdK
          WlpqTjMrd1R1Undza2JnR0xRY28rWmVROFNtYk0yT0dWSVNXL0tHMm9xL2dNT1ln
          VXdndmVROXVmRnovOE9uT25hSlNLMkdnOWVKeGtnQ3o0L2JhVjBYWU1QOHF5YWhG
          SzA4Wnljc1JlYU1tckdFUXpBR2ROMHpFM3MvckhPYzFzQkthNnlBYVlUL1phNVRZ
          bzZ2NCtadWVrMndxdWJFZTFvNlNTeU5pL1d0T0d3V0xYa1Y4aFBZSlpQYTVBZ01C
          QUFHalBEQTZNQXdHQTFVZEV3RUIvd1FDTUFBd0hRWURWUjBPQkJZRUZCeHdnVzB4
          TFV3QlRSaUlTMVZ2di9KVmNPMlFNQXNHQTFVZER3UUVBd0lIZ0RBTkJna3Foa2lH
          OXcwQkFRc0ZBQU9DQVFFQVI5bzMxQmR6N080d21GcFE0eEZjalFkSktSaFBiNndk
          RHdyOTM5OXd5OFUxV0VFOEpEQ0FvbW9nakJlQ2RLUWVqWE1oVjR1Vkx1YllpSitj
          WVRxZXh0NTJNb0pJRmUySnozNC9LbFVkYU5ENE5Jdm5teC9mWS83Qk9qQWFkY2Ny
          L0NQVUxmczE5OHUybG9GNUVSYWtPM2lGajhwWFY1MjlDSFFmekpsaGh0c0VoL3p2
          TXgydXpNTlpaWFlkVWxyQ0NBZGFGbWtFRDBHUHM4SGZDR1VXUytlQVNiN1ZnQXBC
          N0NYb1ZJLzVKa2JzL0ZreEF3TWlxSmE2RUpyYkRkSFl0N2prVktwcmVpMWlxRVBj
          Ri9MU0pNZXhGVjJVcDVISk9OMG1QMGFDclYwbFE2Nys3Mjg2Mkw1VmFjM0tjQTk0
          dVBZVTUxVEJITFNBVXRLTDI5Uk9oZz09XCIsXCJNSUlEUURDQ0FpaWdBd0lCQWdJ
          QkFUQU5CZ2txaGtpRzl3MEJBUXNGQURCQ01Rc3dDUVlEVlFRR0V3SnViREVSTUE4
          R0ExVUVDaE1JYkdsbGMyUnZibXN4RFRBTEJnTlZCQXNUQkdOa2JXa3hFVEFQQmdO
          VkJBTVRDSEp6WVMxeWIyOTBNQ0FYRFRFMk1UQXlOekV5TkRRd01Gb1lEems1T1Rr
          eE1qTXhNak0xT1RVNVdqQkNNUXN3Q1FZRFZRUUdFd0p1YkRFUk1BOEdBMVVFQ2hN
          SWJHbGxjMlJ2Ym1zeERUQUxCZ05WQkFzVEJHTmtiV2t4RVRBUEJnTlZCQU1UQ0hK
          ellTMXliMjkwTUlJQklqQU5CZ2txaGtpRzl3MEJBUUVGQUFPQ0FROEFNSUlCQ2dL
          Q0FRRUFzclVqNDZkeDVvamxhWms3WXRPTDZlK1E2Sm9HN2dWTWFYa0puMVN6bHg5
          TkQvOHc0UGVPMVNRMnNrdWtkT0hBbFFSbXhkZnR6aGNjTlRNNWhtYmNuOFRBZldT
          WXFRRjFSN3M3OGJWanRtYXQ2QVFQMXZTZ2l5WjhBaytpWVpFcTNjMnpWeVlRSEtL
          eFd4bUZadDFIVDgvSC9CM2JYdmVYUWNFUktFK1RxNjZoOHBxVmNvY1FVdHpSRnNF
```


(continues on next page)

(continued from previous page)


```
        WW12MGJSMXJnaHRvcUg4bmhCNXhuZWJnVmxYakFwVytldDJTRTdyNkZqdjFhQWJH
        STg5b3VKMWdzTVBlWDU2UDhBVWphY0Z0TktjNDRPYnU2SFJYWS9qbTZmMm0xRVVt
        ODRFVXNKKzliNStTMng0cVB0dEpEZlNDYXNXWVl6NG1GSjhNd21GaUJHVXdmZ2VU
        MmJVbTZ0N3FxYlFJREFRQUJvejh3UFRBUEJnTlZIUk1CQWY4RUJUQURBUUgvTUIw
        R0ExVWREZ1FXQkJSK3RFQjJ1ZGtFWHhYMGsxNUd6dEYvNG9sMDNqQUxCZ05WSFE4
        RUJBTUNBUVl3RFFZSktvWklodmNOQVFFTEJRQURnZ0VCQUlqeDFmOXJKMkIrbURT
        QTNMMkdSaGpyUFJqZkk2VW4zWjUxQ2VXOWdPOVBNUTV3czVwREp5Qjc5ZEUvUThV
        ZjFlOHBaeWpjaFRzUmE4R1Jkbkt5bmROMmltYXlPVlV2UG9UZDMvWlNtZmt1cmNi
        ajNJNFZXOHNqSFA3Q0U4Zm1VUzhYcHJkcG8yU3hWN29uZUpDMHZ0NWV5aDhtZ2ZK
        L3FTYndWYWlYdUgxV3hpNmR1QXZkeGRkTVhBeFFLUEcxS0tWTTdDWWZDZHBYL0hh
        Z0NPSHpjdG8rMzc0ekZxcW5RMUt4NXJiZ3Z4TlNnbS9QRERPTXdQMDMrYmJUUjYz
        S1NLMVZiZHRMQnVTNGpnYVBhYnd5eFF6L0ZjaXdUdS9ITE9RbjhUTnFEV3lvSWJz
        K2VRWDJNZHMyQXB1bDhYSDIrQ2FrakJMTUxMM1RsajJ4KzZ0S1I5bz1cIl19Iiwi
        YWxnIjoiUlMyNTYifQ",
    "payload":
       "eyJwcm90ZWN0ZWQiOiJleUpyYVdRaU9pSmpaRzFwTG1odmN5MWhMbVp5WDJWdVkz
        SjVjSFJmY0hWaWJHbGpJaXdpWVd4bklqb2lVbE5CTFU5QlJWQWlMQ0psYm1NaU9p
        SkJNalUyUjBOTkluMCIsImVuY3J5cHRlZF9rZXkiOiJ0WkQ4ZkpUT1hGZ3NTVi11
        RXN1ZS1VYURnVVpQX3FFZWFVUFFyQmpObUF3UlpMSXlONklUc3hhclRqbDR3YjdH
        UXltN0prOWw4QXlld0Fkakltby1yOUNULUhSbmR5Nlc2Nkd4bW11TGhwOV9ncjl2
        WEtjeXh3QUhHeGIyZGo3dmlIQjlIRElNazZEQzB1Ri1naWZlM3VaeEhjdVVfYkRG
        VGZBU0M2TEwyLTlIQ29OMzlSM2VZcC1FeWc0clhwZ2t6SkpnLTdRSUpHV3pvRFF1
        d2VfeEM3Vmh5Q25vdDc2bW9RbEpSM1I5Snh6M2M4TE96ckp0YTY3YjdhS3Zodzk5
        eUNPX3REb3d0WDZpNUd1cmZVbE5GN2VucmUwOWd3cUlLT3ZxQWRDTVdlSU5YWnVO
        WjlZX3Y5RjVvWlF1M21BcEdib0dVdVRvRWxUMFFKd1dwaUR6bUEiLCJpdiI6InVJ
        ODQyQ3ZsRUpBYTd4Y1YiLCJjaXBoZXJ0ZXh0IjoiVjBCcWtLTV93TmV6UVJuOGxv
        b3VOdFhPem9tSjJLbElYaU1NT0VQSk1NTVZpd0lKY05IWHdBV01wUnBoMlRrMnlS
        dUoweHBZbmRPdVBtUHdPVGljMWR4YXZvaGJEdHloTEFQS09TNWIzSk9lQTgxbWU3
        enRaeFdxOFRDVVRFVkhBVTM5dERzZUQyZ0tUME0yOFR1dXpEZHhJcEtUMXJQRXZr
        WThIQVBjakhZQ2pTeVpmcEhJSlpkUUhEclo0YUZBN0xETkkxWkxPOEhXMmtJbFB6
        UzQ0WFdKQzYwS243THJvSXRyZ0hCWnJfZEFpNWlIQTdmU0VCSFVSRFpPQ0pfdzlr
        TldLMDZfVTN3S2xVNU9DeFFYNzFMRGxCQllENlpaa2lwaExoRl91RUxvZFdKREo1
        Zmg4VnhDRUJ6ZVdqckR1M3UzMXNldHJEbmRIaFNsT2VyeEswNFZuQXNETlczRXNT
        a2twLXp5cFh4d2ZBRVQ0VWxiR3V0UV9FX2xsX2M4VVZjSnJhRXJoZFgzX2VNajdS
        YThud2VEdGxZWjlDLWVlQkdJYzE1OTZLaVZzQlJ0c1dPZm5SLWQzdVZFREMwZnVv
        alp1U3M1Mm1mWjh5dU1yeE9QWDJodzFVVGlCeHRuODdyQjRNc1ppSkwwc2VIelIz
        T3d4eEFMUFJRSzFzQVNweXJRaW1jTVJBSUtGVzJUUDFjSjRmYTlIdmNTbzN0eTUx
        cENSVGlmUV9CSFlLM01LaGdWMWVnZ0I2Mmw4SEs2NkE0dFlpcjJyUnhkNEZPb1U0
        bXN0bVQ3aFpmOVhBRzFBMzNwSHNxSEFINng4LVJDc0FwOHlGeHd1SHg3X0s2ZzNo
        ZzJNSTNHZUpBTGxMRXhyeUJ5NEF4OXQ5ZEg0SWkzZWZjZlh2LVROcmFIS0tHd3VM
        QUItby1jVG1WUWZfU3dGSEtHZGNjSWgwazFYT3VXdkFmekducjJKMnhBVjBUQzdz
        YWNyZHBWNzgyWkM2T2dSSVliTEs0RnhUYXpuMXY1dlZjRms4dU40NFgxV2VuM09f
        RmR1UllLZ0ZPbUFyTHd4WXBMYXJSS1QzX3FPcjlOQnYtQ3FycEJPV2EwVlVtaXF5
        TWxTb2VHR2plaHcwLVNuMVBTa3dWb25jNTVYTWxYcEVZc2hJb3dHN0xnb0dBakxl
        MU13cjkxNWxmWnZsajFpckljbGk5MGtIWVBKTjdZVTJEMVhaM2hEV3FLTTRLWG1w
        RnJaVk5qdi0yUWN0ZGt1Vk1MaHQwN2JESU51aUpDZENsS0llbWVVZlhZMW1IakxB
        N1J6Tjg0SjN4UTIzVXN5eFdWLTFYRTJueEFmcmYzMG5KZG5HTi1KbEhqdFMtQW5E
        cXljUGQ4VjhBcUw5UURrUm1tZmsxRnBwTUdEaW9XS1lVM2JTTGhyTHh0RzMweUQ2
        bzVlb0UtUFN0aW9XcGVsN1lZcVg0aGxlNGNoNWVhWkQyRF93Ni1VOUxLbmQ2NnZq
        OVptNjU2YVZrYy1SSGhNWEpQczhMeElCM2hrQ1BSMi1KN2lUaTBZYTh2d016OFJ5
        LVQ2Z285b3VVU2F3MHJYbi0wNm9HNC00aXZhMnZ2UkZGaXFOS25KYTdrM1BrWDF2
        cXVqQUl5Ti1ESkI2cmZNZnZxTlZmV0phNXBuNTZLY1g4TFBPeUFiSGo4WUNWT3Fo
        QldtbGt1SjM4WUtpY1dWRkJfTlpSZEpseDRTUXdoQ2pLYlJtdmxENXJqeEk4NTRY
        NTdzQkNQbmpaTzdvZDRpZ2lVbnlfZjNyRkZQd1lieVFmdWxkM3hMRVdSWWZCWUpu
        YWRvNzB0S1VXMllOcWFzUzR3SW9ibFhFSnR5a2RPTHdQLVkybGRVMXdVbThsUHdp
        VXpNZGtrY2xDWkQ2Wkx6LU9GdDdLaE9EbUxPTUdPNWlhczlzeG9DaG4zYzFMcnFQ
        amd1Ui1RWFNoaUZrNmtYVU5CMnlIVEt3dmZZT0ZjUjhEbDdleVJCQ3NVUUY4STdT
        ZkFWNkJrMm82ZU1jdEJGTXluWFgydU9nWXEwOWtUM0ppckhzbGl3dThZbzM1bHpW
        ZGViWnlqREIxbWl2LUNKR091ZjRxbEZxRzNiYVBCQmhNa0ZFS2lWU0FJZmdDaFlu
        dUc3WkxvNmVWMW4zMnRGeDFzdEdUcnpuMk13alNzektPMTJIY3VkcUZnTUREQ28w
        T19OREZ4Qmh1RFB3aVNadnBGNHJWV201QktEbjBOQzNFcy1zOExGaDZJUFVTMW1l
        VFB2Umx0cERJZTljQTZqdHhCQURacVphdGtreXB5ZzZGRDZzUjhUQVRDRGo0WWhT
        Rzk4MVBmRFZ1S09FVjlqaTdzd3lkLW9ST2Q1UXZCa1lKUkxhckpEOGhEdWZIcHdy
        MlViNENBb1JKcnJWdGM4TmJDcmlLbmZGWVBQal8yN3p1RFZKZXBxQTdpOVVTY0ZD
        T25vUEx3NEFpVmxqQVRPVU93MmNzR0FqM19tLThPR2cxM3BTUHNjWDlSUm5TV1lY
        cjdFdnFJaUVXNDZaNTR6NlJTNHg0Z1dqZWFzNUpHdmRsQkhoQk42bFd4QTFLa0Vs
        el90NlVOSVhGSWpZRTdyVF9oUzJsWmstWGNPelZuM0gxTllZU3NpTVVLVDNvWm4t
        UmliVVhoMGZjbVNWTHFRMnVuX2dQV3MyT0VUeHB3U1Y2dk9leWRjOFcwQVdhSVph
```


(continues on next page)

(continued from previous page)


```
        Vm43bW1MNnY2aEctTVFLSl92S3pfaTNxbkRxRUVzVjhDQVE3NTN1cWpHUVBEMUs4
        VGtZeTNucnhncEh1ODY3UDJNUGpVQWFxeldTV1dNN2FQeVhCTkZNeFQ5V3Q5ZmVw
        VEFheDdhLVo1VGcwS1pxS0lDQVVNYUo4OWJveG1ncG9rcGFRVWtQS1pFMjR2NGph
        ZE5pT21SSHJMeE11VzVrdG5kQk5Bb0dHbm9HRTNzYWZlc24wX2NPc3M2TTVXcEhR
        Z1N6VWxacF9aREJ5Q09ldFo1TmEwMjVfb2t1T2drUklKVjN4MmZPbWM2U0wwU3Fy
        R1lWSEhORmZKN2lLeVNXTk5Sdkl4UmhWTmJYVXFMRl9OYUMxZDdNQmpkTHY3WlFp
        Ym0yT3BmSWtmSlNNUzFKcFVwSVNRakR4djk2TFhIOHBhQXBTVDZtWnlEa0VrTW1O
        b2ZXSktwM0pQNUFTTXZ2S3dsbkgxQU9rSWJYZjdRLXlyYkthY1BtWEhkZ2hjRmZa
        Vi1ldU9ERzhfdTRWdGZwOEF0LVBRSkJyX1V0cVZJbHpReDBnUE5MVVdMbzdlRmlv
        dEtPTEx3Vjh2MGpxcDdUUlBIVzBHc3pVWmFZb0RjYUZMaXZDaFJwVDktQks3d29w
        RXhfN1FxUWUxSVhuMVNKQWRtZFpvZmdfSF9ZeWU0MVJjSnZGeWp6SmZBOGlranBs
        STMxVW9WSzVaTllXSVExZ2lvUFh3U3JUVW1GeE45X3Z0aHl5Z1pYTXQzVkZiUTNB
        eFBBZjlSSE5ENlpBOGFValk2R0pNb1BvX3FySkMyLW9hQzd0cTc0SURBekk3bmpM
        S09VMG9IZ3RtaGhGUVB6cloyWjZOYUlLaTJIWm9yLWc3LWctN1N3ZGRxd0RDSnlO
        ZXdEb3pJN3NNRFM3XzM1RkoydnRnaGZLUHFTLWhFd3JlZGtaNE1ORUg1dnRXU3F3
        NHRmZDFrQmNROExtRlZmM2hfV1FWZjVqQ0lGcEZNOV9MYnV6eWVmeU9IMENSLXlJ
        YlE0QXFleS1ZclA0YWJTdXF2VXBlMEFyVEdZS3pMWk5FZHhFVzliemxsYUVJaVNU
        MTRBNXAzam9CUVNOQmtLRUdPMVh2RTdSVnJ3T1kyemItWWFIRkxjU0gzdDRtMFlZ
        aVducFZ1RnBRZjExWVlqbTgzLXFEbHZuYUN5T19GQXBTMy1SRnhrVTZFVzRYcDZ3
        eVlxNDF1amxGVHp4Q1RnMExxeklqTGlyRm1VSGpKZUlhR3lVSzZ5TGpkYkhtR0Ns
        LW15T3FZR3hwNERybHVITHBkaWZHenhmQnd6OUM4MXNNbkFxYmNRWGMwcXdIMVdO
        al9uX0tMblJFdUhRWFBleDV2VmZ6eWRlZUZfNE93WExSdXZ1M3Zlm3Z3VldWamRP
        ZWN2a05nbkVFUDdocVhkbXpRbVlKZndxSGRwZnNSRExESDloOWpRcEI0LTNMRjg4
        U2NSa1RteFlNU291bFVqcHdia25hTWZQRVMtZkgtTjVwQXNqVE0wOVFTV0Y5NENE
        bHdZR1ZhZzY2UkdLcGdwU1ptMGF1dGNsSG1KOGVRMjRuRzNKdEF6bGNkVC0zNXNC
        TlFsZHRkYlhkRzFxb2ZlOUpUNE5iSll1VXdwemlhNnJYNGQ5NkRqdm1VdWpqanNl
        NjhCYkx2bzJQM1VyTkxUWDBjNmUtb3lHbzcwdURTN1lxb2ZIUUxhU05ORHRnZ3d1
        V1IyZFRVUDdLRkk5N29Qb0ZiX1JjS3BQVVU4cDZsTjJRYkVOYy00TWNleWJYVS1U
        ODR1ZnpqZkJkdHdMdkYtQ3d6Nk9lS1V1NnBPeVN2MGVnbFNJUWlmb3JPU3FGVl9y
        QWdvX3NuNDVHVVhKU0dNc0FFT0NXN0lQX0E4Y1hvaFpGbi1NNW1UTU83b01fSWN1
        Q0dVclNuREV2X3NNdXFLdFk0cExbHRMT2tCTXVHU292Mm1DVHBmQjVnVnBvaGZf
        UzBaUW9CWlV2UDkwU3ZLUVRuTG12QnZnVTd3OVVmY1dRTzBrNUZZSjNfREZxNHVL
        VWJpNXB5cFVCT0txd0Zla3ljLXZXVGRPTGdLOW40ZkdUcjlzOG9ZejczdC1tVXZ5
        a3pfdkRzWWFpWHZQbkd3My00SDhVWllXZy1GSHJfSHlLTlFQNklGSTVGdEFneW5s
        QjF6SEQ2a0xXbGtGaWdfb0FjdXJ6enE3WXJJaUlOcnZFUzFUZUgwRXBwczVjZHpF
        WWh0M3ZHNFMtQTJqcE5PN1pJUklzUmFUMHkxbFBOLVZEdWJxOHBFZmhyX1B0ZVM0
        T21zNGNvLWxiOUJNR1BmeEp0cVRDUWJGSnJLTFBDbDJKNk5VVGtnTy0zVlgzUkIw
        Rk9uUmdWRl96aVQ0amF1UFFxZDRsQ3BOME12S2RCb29BWGtYNzkxU2JjbmJoMUhS
        cFlhblNVZENiZmZ5MGg3NTJ1MldmV3JjSUVDa29GWkRqelUtbG04bnF0OTZnS1ZH
        OFpHejF1bGhIZzVfVVIxVnlZNElROVBjNHVCTk1yazRUVThNRjF6R1RsalhnT25t
        UVZld0tPWmtkcFB4eWhGdExtalUzeWtoejBKMm9oMzJpaVhpbHFFeVQyQjM2Nzhz
        NHF3NHNkalNTdXdCWXZZZmtRNl80SnIzNnNrUm5CQ1ZLa21mdlFtbld4S3NheWQ2
        WnRPcllwZ1hjTlRfOUpYXzE5RFZiekdYbW1zTExONzRPQWg0QjZUVXciLCJ0YWci
        OiJnb1QzT0V0XzZ5eVJXdEtPdFJpUTdBIn0",
    "signature":
       "a9idqlHPlsfvg7IEKECf_XNeKKpU5jioicQTxvpQywpT8DM5M7sy0PfrOSEL4iKJ
        hGd32xQ-JwaIgT5RR0QwAM3cy15y0IMh-KkHh1H3Amsy2RIsi2jkZVpIN24OF0T4
        Wf6fXfuuXjBydT-PjIjQako2Lsic-nKAJukm0f3uwHxGXN3JGlIzi4_QrYeRjI4V
        b9ymaZW6soF1VaDqXulfaL7RtN_B46-Yg_N8XK5XWuIW5PZ1ZEkYFaqGzPaoMi31
        ipA1vx-VlQdVxCATOx2uQOT-um6NJAlvAigyWn45CRM5ETXja_V8WkpDPlHH4A5G
        SEzKUGz3tYC85Agd0-Aoig"
},
"dac_request_dest_certificate": {
    "kty": "RSA",
    "kid": "cdmi.hos-a.fr_encrypt_public",
    "key_ops": [
      "wrapKey",
      "unwrapKey",
      "encrypt",
      "decrypt"
    ],
    "n":
       "uL7ANgD80H5sNqo3nHzovPRxgncQLhz0oQvGMVvULCkrYXMaXZ5sNv7fT6UdMSZi
        T-e0sthapmEqrpeV9RKHsiF3COGl2YndUHixpEkHp8y1ggcH6iTzoBsgXMZ7OLW-
        mJ2RW3rodT7k-tcozYYsTSM5egMPQSAKgt0nMnPmdNRnEyA2_NJ8Y7lNkEXyja0Q
        JLstzkP8-cKS0BkEquLQEMbZVRM6U5uG69cj1i9OWvuRzPoaATKyt6Cc4f6PUu9L
        OyCBUAs9dXsRrt3B8H1qe7io7FAAcOpcUDKDnLFXS1THc37DK_zEyKZcMttjCvEl
        Ovt-cIaokdnxJeggv9AFGQ",
```


(continues on next page)

(continued from previous page)


```
      "e": "AQAB",
      "x5c": [
        "MIIDMDCCAhigAwIBAgIBBDANBgkqhkiG9w0BAQsFADBCMQswCQYDVQQGEwJubDER
         MA8GA1UEChMIbGllc2RvbmsxDTALBgNVBAsTBGNkbWkxETAPBgNVBAMTCHJzYS1y
         b290MCAXDTE2MTAyNzEyNDUwMFoYDzk5OTkxMjMxMjM1OTU5WjA1MQswCQYDVQQG
         EwJmcjEOMAwGA1UEChMFaG9zLWExFjAUBgNVBAMTDWNkbWkuaG9zLWEuZnIwggEi
         MA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC4vsA2APzQfmw2qjecfOi89HGC
         dxAuHPShC8YxW9QsKSthcxpdnmw2/t9PpR0xJmJP57Sy2FqmYSqul5X1EoeyIXcI
         4aXZid1QeLGkSQenzLWCBwfqJPOgGyBcxns4tb6YnZFbeuh1PuT61yjNhixNIzl6
         Aw9BIAqC3Scyc+Z01GcTIDb80nxjuU2QRfKNrRAkuy3OQ/z5wpLQGQSq4tAQxtlV
         EzpTm4br1yPWL05a+5HM+hoBMrK3oJzh/o9S70s7IIFQCz11exGu3cHwfWp7uKjs
         UABw61xQMoOcsVdLVMdzfsMr/MTIplwy22MK8SU6+35whqiR2fEl6CC/0AUZAgMB
         AAGjPDA6MAwGA1UdEwEB/wQCMAAwHQYDVR0OBBYEFBAIGICMR5H6KLKM1ZAEhCCc
         KwE9MAsGA1UdDwQEAwIEMDANBgkqhkiG9w0BAQsFAAOCAQEAANYSSryUU6112pYM
         r83M3GWnjzu16B+4KgimZ8kbey94zNPdwmwQdSe0Xmg+1Otc6VUB40ouNnwK8efB
         aWBtXwCA7Nb715nTqo2+rn+X+A0mGrYaKkToPEe8ZYwDcOlOpNC9JFE+QgP9/CJa
         AaWrf95W+4kra2WnA4Bhqu2WWXnQkL47/nKcGVZgQAH+mVnxPaI0gELYdonXU/S2
         8HqxoyjpGL/vmyc46zUbxYsgx/jiE7J0fJVP6Yk/3dlNYCCpLtV8VmzFAQAeCcn8
         AWowFcd09a4SY09rn1MUv/rrvXpzflfn9j7PtRRFj2e/KhitmOH1zKDuYzREpUOu
         TDlPIQ==",
        "MIIDQDCCAiigAwIBAgIBATANBgkqhkiG9w0BAQsFADBCMQswCQYDVQQGEwJubDER
         MA8GA1UEChMIbGllc2RvbmsxDTALBgNVBAsTBGNkbWkxETAPBgNVBAMTCHJzYS1y
         b290MCAXDTE2MTAyNzEyNDQwMFoYDzk5OTkxMjMxMjM1OTU5WjBCMQswCQYDVQQG
         EwJubDERMA8GA1UEChMIbGllc2RvbmsxDTALBgNVBAsTBGNkbWkxETAPBgNVBAMT
         CHJzYS1yb290MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAsrUj46dx
         5ojlaZk7YtOL6e+Q6JoG7gVMaXkJn1Szlx9ND/8w4PeO1SQ2skukdOHAlQRmxdft
         zhccNTM5hmbcn8TAfWSYqQF1R7s78bVjtmat6AQP1vSgiyZ8Ak+iYZEq3c2zVyYQ
         HKKxWxmFZt1HT8/H/B3bXveXQcERKE+Tq66h8pqVcocQUtzRFsEYmv0bR1rghtoq
         H8nhB5xnebgVlXjApW+et2SE7r6Fjv1aAbGI89ouJ1gsMPeX56P8AUjacFtNKc44
         Obu6HRXY/jm6f2m1EUm84EUsJ+9b5+S2x4qPttJDfSCasWYYz4mFJ8MwmFiBGUwf
         geT2bUm6t7qqbQIDAQABoz8wPTAPBgNVHRMBAf8EBTADAQH/MB0GA1UdDgQWBBR+
         tEB2udkEXxX0k15GztF/4ol03jALBgNVHQ8EBAMCAQYwDQYJKoZIhvcNAQELBQAD
         ggEBAIjx1f9rJ2B+mDSA3L2GRhjrPRjfI6Un3Z51CeW9gO9PMQ5ws5pDJyB79dE/
         Q8Uf1e8pZyjchTsRa8GRdnKyndN2imayOVUvPoTd3/ZSmfkurcbj3I4VW8sjHP7C
         E8fmUS8Xprdpo2SxV7oneJC0vt5eyh8mgfJ/qSbwVaiXuH1Wxi6duAvdxddMXAxQ
         KPG1KKVM7CYfCdpX/HagCOHzcto+374zFqqnQ1Kx5rbgvxNSgm/PDDOMwP03+bbT
         R63KSK1VbdtLBuS4jgaPabwyxQz/FciwTu/HLOQn8TNqDWyoIbs+eQX2Mds2Apul
         8XH2+CakjBLMLL3Tlj2x+6tKR9o="
        ]
      },
   "dac_request_dest_uri": "https://cdmi.hos-a.fr:9001/dac/"
}
```


The DAC provider at hospital A will retrieve the signing key from the JOSE protected header, validate it using the included X.509 certificates, and then verify/decrypt. It creates the following (plain) DAC response. Note that it included the object decryption key.


```
{
   "dac_response_version": "1",
   "dac_response_id": "73da04e1-2182-447e-8342-f4b9f06ec936",
   "dac_applied_mask": "ALL_PERMS",
   "dac_object_key": {
      "kty": "oct",
      "kid": "encryption_key_1",
      "use": "enc",
      "alg": "A256KW",
      "k": "1mk_8n9GZJTLDUEuxBYT-9GO8bC_fR2qqt03rVSRFak"
   },
   "dac_response_headers": {
      "CDMI-DAC-AuthInfo": "Key successfully obtained from KMS."
   },
   "dac_key_cache_expiry": "2017-04-06T14:42:47.393Z",
   "dac_response_cache_expiry": "2017-04-06T14:42:47.393Z"
}
```


As before, DAC response will be JWE encrypted using the key in server_identity. The result will be:


```
{
   "dac_response": {
```


(continues on next page)

(continued from previous page)


```
"protected":
   "eyJraWQiOiJjZG1pLmhvcy1hLmZyX3NpZ25fcHJpdmF0ZSIsImp3ayI6IntcImt0
    eVwiOlwiUlNBXCIsXCJraWRcIjpcImNkbWkuaG9zLWEuZnJfc2lnbl9wcml2YXRl
    XCIsXCJrZXlfb3BzXCI6W1widmVyaWZ5XCIsXCJzaWduXCJdLFwiblwiOlwiblQ2
    Rjc1aVZGZkdQLXdramdneVhacFdvRWhxNmZTWkNfRENTYXNLLUdVdXdWaUhETUJG
    cXVreUM4ZmsxajNBY0JyTTlIZERIbnFrVzRfM2YzOVdZMlMtRFY5YjhkWWpRRHJl
    TDFfcHBNRVg2enJnN1hBWEJJa1ViT2h1dXJOTVNBNENlQXlCY2xqWUJpQ3dvTWxv
    aUNGb2RsbFYzUDZwekVLNjduTHNfYWVfTHVaUmRzaFhXakotYm9qQzZiNGhJcWVx
    UFh0dXBPekl5MDBKLVZydHNGUTBZaDN2dWRmbVVJTkZSTFl6c1lzY1Y3TVR4TUd5
    S1hIMXE0ekdHSHZmSlpTS0MxTEtYYk9Dc3JhaHVrUFVhY0tBSFBqbTNKaHpONTR6
    amVYTGxnaDZPT2x4X2Eza1V4YlBFTl8zYjhHTDhOVUhuWFljcUNjVDVXUUV2ZUZw
    OFlRXCIsXCJlXCI6XCJBUUFCXCIsXCJ4NWNcIjpbXCJNSUlETURDQ0FoaWdBd0lC
    QWdJQkFqQU5CZ2txaGtpRzl3MEJBUXNGQURCQ01Rc3dDUVlEVlFRR0V3SnViREVS
    TUE4R0ExVUVDaE1JYkdsbGMyUnZibXN4RFRBTEJnTlZCQXNUQkdOa2JXa3hFVEFQ
    QmdOVkJBTVRDSEp6WVMxeWIyOTBNQ0FYRFRFMk1UQXlOekV5TkRVd01Gb1lEems1
    T1RreE1qTXhNak0xT1RVNVdqQTFNUXN3Q1FZRFZRUUdFd0ptY2pFT01Bd0dBMVVF
    Q2hNRmFHOXpMV0V4RmpBVUJnTlZCQU1URFdOa2JXa3VhRzl6TFdFdVpuSXdnZ0Vp
    TUEwR0NTcUdTSWIzRFFFQkFRVUFBNElCRHdBd2dnRUtBb0lCQVFDZFBvWHZtSlVW
    OFkvN0NTT0NESmRtbGFnU0dycDlKa0w4TUpKcXdyNFpTN0JXSWNNd0VXcTZUSUx4
    K1RXUGNCd0dzejBkME1lZXFSYmovZC9mMVpqWkw0Tlgxdngxa U5BT3Q0dlgrbWt3
    UmZyT3VEdGNCY0VpUlJzNkc2NnMweElEZ0o0RElGeVdOZ0dJTENneVdpSUlXaDJX
    VlhjL3FuTVFycnVjdXo5cDc4dTVsRjJ5RmRhTW41dWlNTHB2aUVpcDZvOWUyNms3
    TWpMVFFuNVd1MndWRFJpSGUrNTErWlFnMFZFdGpPeGl4eFhzeFBFd2JJcGNmV3Jq
    TVlZZTk4bGxJb0xVc3BkczRLeXRxRzZROVJwd29BYytPYmNtSE0zbmpPTjVjdVdD
    SG80NlhIOXJlUlRGczhRMy9kdndZdncxUWVkZGh5b0p4UGxaQVM5NFdueGhBZ01C
    QUFHalBEQTZNQXdHQTFVZEV3RUIvd1FDTUFBd0hRWURWUjBPQkJZRUZNWThmOW9h
    aXhYTkFWSW1Nby9heE9kN205clFNQXNHQTFVZER3UUVBd0lIZ0RBTkJna3Foa2lH
    OXcwQkFRc0ZBQU9DQVFFQWJPejR5akdLektpM0pVQTJ0WVVOeFhoQ1RKbnBVUGZB
    WDhRdGs5ZEJXTzZhVXNsSzJlTkh1TzZjL2tKMG0vKzR4LzJiczJlU0dISmRldU9w
    dzd3QmQwTDB6L1hUeStFVHRQM0k1TEovYzhNdmN6NTQyZCtMQUpXdlZtU1hIQU82
    QjZRZDJtSkJ5aGFMU3k2YWUxK3BwcmE5Yk1aYjFTVWVxc3ZtZ1R5Z2h2Y3NkVzky
    TUhBUHlxU1hxTmthaTNhNW5kbE14YjdqcWFhMzNCZkxTOHhVcE1FQ0wvaGJ4L0dH
    WXcxZXIzSUFCVDBCeG9JQXp0c0tHQTFUOWh2cnA4NCtFNjJhcnZrU1lHUk8zQmpu
    SGRHcGdnRUhRMmhGNkR1UFIvRGVyV2d2R0MyREN0cFNxVzJTTklaUFprb2tsZ3pE
    QnVXdVlHYjl4d1RWa0psWVRpSzFOdz09XCIsXCJNSUlEUURDQ0FpaWdBd0lCQWdJ
    QkFUQU5CZ2txaGtpRzl3MEJBUXNGQURCQ01Rc3dDUVlEVlFRR0V3SnViREVSTUE4
    R0ExVUVDaE1JYkdsbGMyUnZibXN4RFRBTEJnTlZCQXNUQkdOa2JXa3hFVEFQQQmdO
    VkJBTVRDSEp6WVMxeWIyOTBNQ0FYRFRFMk1UQXlOekV5TkRRd01Gb1lEems1T1Rr
    eE1qTXhNak0xT1RVNVdqQkNNUXN3Q1FZRFZRUUdFd0p1YkRFUk1BOEdBMVVFQ2hN
    SWJHbGxjMlJ2Ym1zeERUQUxCZ05WQkFzVEJHTmtiV2t4RVRBUEJnTlZCQU1UQ0hK
    ellTMXliMjkwTUlJQklqQU5CZ2txaGtpRzl3MEJBUUVGQUFPQ0FROEFNSUlCQ2dL
    Q0FRRUFzclVqNDZkeDVvamxhWms3WXRPTDZlK1E2Sm9HN2dWTWFYa0puMVN6bHg5
    TkQvOHc0UGVPMVNRMnNrdWtkT0hBbFFSbXhkZnR6aGNjTlRNNWhtYmNuOFRBZldT
    WXFRRjFSN3M3OGJWanRtYXQ2QVFQMXZTZ2l5WjhBaytpWVpFcTNjMnpWeVlRSEtL
    eFd4bUZadDFIVDgvSC9CM2JYdmVYUWNFUktFK1RxNjZoOHBxVmNvY1FVdHpSRnNF
    WW12MGJSMXJnaHRvcUg4bmhCNXhuZWJnVmxYakFwVytldDJTRTdyNkZqdjFhQWJH
    STg5b3VKMWdzTVBlWDU2UDhBVWphY0Z0TktjNDRPYnU2SFJYWS9qbTZmMm0xRVVt
    ODRFVXNKKzliNStTMng0cVB0dEpEZlNDYXNXWVl6NG1GSjhNd21GaUJHVXdmZ2VU
    MmJVbTZ0N3FxYlFJREFRQUJvejh3UFRBUEJnTlZIUk1CQWY4RUJUQURBUUgvTUIw
    R0ExVWREZ1FXQkJSK3RFQjJ1ZGtFWHhYMGsxNUd6dEYvNG9sMDNqQUxCZ05WSFE4
    RUJBTUNBUVl3RFFZSktvWklodmNOQVFFTEJRQURnZ0VCQUlqeDFmOXJKMkIrbURT
    QTNMMkdSaGpyUFJqZkk2VW4zWjUxQ2VXOWdPOVBNUTV3czVwREp5Qjc5ZEUvUThV
    ZjFlOHBaeWpjaFRzUmE4R1Jkbkt5bmROMmltYXlPVlV2UG9UZDMvWlNtZmt1cmNi
    ajNJNFZXOHNqSFA3Q0U4Zm1VUzhYcHJkcG8yU3hWN29uZUpDMHZ0NWV5aDhtZ2ZK
    L3FTYndWYWlYdUgxV3hpNmR1QXZkeGRkTVhBeFFLUEcxS0tWTTdDWWZDZHBYL0hh
    Z0NPSHpjdG8rMzc0ekZxcW5RMUt4NXJiZ3Z4TlNnbS9QRERPTXdQMDMrYmJUUjYz
    S1NLMVZiZHRMQnVTNGpnYVBhYnd5eFF6L0ZjaXdUdS9ITE9RbjhUTnFEV3lvSWJz
    K2VRWDJNZHMyQXB1bDhYSDIrQ2FrakJMTUxMM1RsajJ4KzZ0S1I5bz1cIl19Iiwi
    YWxnIjoiUlMyNTYifQ",
"payload":
   "eyJwcm90ZWN0ZWQiOiJleUpyYVdRaU9pSmpaRzFwTG1odmN5MWlMblZ6WDJWdVkz
    SjVjSFJmY0hWaWJHbGpJaXdpWVd4bklqb2lVbE5CTFU5QlJWQWlMQ0psYm1NaU9p
    SkJNalUyUjBOTkluMCIsImVuY3J5cHRlZF9rZXkiOiJLZ1lXVDVDOVBiN0FEOVZH
    bTczSDRQSkFJUHNnbTN2ampNbmZJcXZBOEFLOEtNREowMTlNQkVHeC1tTUpIQTBo
    dGhyb1F0UXA3NklDR09GLXJaNnBxajNTYm1iNFNUc3FhUjQ5aTN2Y2x1dUhRSlZM
    WVNOVklWOHBJYzZLbllDWFNGMzNldHlIZ0p0bFZjXzE2RTIyUGxEVE9IWjhPRzZz
    dklWa1o1cktnLVNGS1ZZbVlWeC05UC04NU1ueThRMkYtR3VBcDNiOHVZZzhFNXdx
    Z1AwY2NHYVNjZF82cFUzdk83aUZGWlVLLV8ycHlzQWdRMlo3Zkl0a0lSWnVPdFFf
```


(continues on next page)

(continued from previous page)


```
            RkpNaThLcFdKc3lMNFNDanZSNkxFSVBIekdTMUtkTmNEVTVrU28yc3pkanQ0WENH
            Sk9WSkc4dFhfWmZRTjh6RzdpTktJcXhiTVYwRUZVOWRjUzBkeWciLCJpdiI6Im9n
            Z3hKM3g4YjBhb2xlSDQiLCJjaXBoZXJ0ZXh0IjoiLWhkM1A3UGM5emtpQ0hhYnda
            ZVhWU29rYmNXcmdqQ1QyU0FZaFFGYzNMNFY4b2ZLYVNYR3UtazF3T014OHhZY1FH
            Slo4Mk12b2tEUXNGRnZaQjBNUG9hdEljQWtmTms0VURqREcxNi1zVGc3QURKeV9p
            TkdmZ1dkUy1XU3QxWW1PZW5HaXBYNEZWM3ZyNU9URlE4X3hCRHdYSHRiU05KX3Jy
            RlVWX2R3TnJQdUdiSlYzSXhOYi01RFJ1Rkl1Z0ptY0JXelJYRW1LeUEyZEdZTG1V
            eEF3WFpWT18wYm1KVkxzZzlKa1puczU3YXNqR1Fna0c4ZzhCNDlnTEl0QWRFQXpJ
            SFlrNTVpUkRvQk9WRnhGR284Vlc1ZFNrM05ldi1mZ1JxYzFDZEdodEZuTnlzUjUy
            akhtMFprMzRoeTJRLWZVNnNTVXBwMU5vVmw3YTd1NWMxdEVZSGRNVU9fWWtHV0g0
            dFIxeWdmbG94dFY3ODRMYXdObGNCYlV6NFpvWXFKeXg2amEzdGxHaUpxaWdLZ043
            WENTT0R6RXF0QVRQeXBITnVfYXJHRm9LNmk3cGdKZXcwYUxMQWJQOFA0SFJHTVZ6
            Q1V6dktDbDdKYS1XTE81U1UwS0ViLVhJZGdHX195a005WTJqUVQzdi1MUjNNRG1q
            WlE0ZDM4cG9BS1lfSHJCWDg1Z1dMQXFzY3MyRHhwRTlIZm9xaXk0T2tscVlaa3Js
            MUpNT1dTQVZkWWNqckhjc3Y3VjZGSUJGV2Vjc1Nwb185ZUsxOCIsInRhZyI6IkJ6
            NFZZeHdtNFkyU1Y4bDVUckMwUEEifQ",
      "signature":
         "RjUCI3Q_zfBJyeHjYfldsd6MppSDNUAIzC77lsbM1MiKfLDi8oN0999gpByS7Sx6
          kCXqsNkV4T0z_qaqy4UY9JrdjMRTNFPXJMwhqbBem-s6dJT6VquF3GBQTU8wb4OK
          5E8rGvTcWw-Hd0SfpjGoJtgv5RmpzfVgdvANZcJfST-rOra3EnPitOf8dJ95Db3t
          78mEbMfqdoobk1Dnc39DvpnzD61TIoxWoZj3UGcBcvNPsl2XijS6yZlgAsrQbGnX
          xWx-PCwEACZoVekzt-YV5QUFH2Jqb1pGeUUCwoFv1ON90iXVusIdWWnJO51gSKwg
          i80ZxOSSBwF6b9WIeXHe5g"
   },
   "dac_response_dest_certificate": {
      "kty": "RSA",
      "kid": "cdmi.hos-b.us_encrypt_public",
      "key_ops": [
         "wrapKey",
         "unwrapKey",
         "encrypt",
         "decrypt"
      ],
      "n":
         "oQMQkY85UzwO7K6H0QQNfAiRMN3ZfhK0aXEKx7YwvrCU9IKOquZ1OYZ9Cv8556_8
          E8yZm02JDWOBOaSsGHU835jvXf12f4MywKGWj5FtIGL-j9kXF6SWq3zuLVYlXpMI
          KsJngHMVFca_-ZhZ2vLsrnDR1aCNEC48gR26ewp6WX1ptnSc1W4x3Mj-ONMVzxVE
          7XNlwYysTgDtonmTQD-YG6_KhhAPx0YowMbUWv_cMQvXsi7MMDyZn6fxfq4zQmQ2
          V5RtUy5msd6K3beDzS4LmZhsJmjU7YnhOj0pZby4Zckm43npjXPAuwPhzK2OW7qb
          fkv0qm4rsFWUcuNh8lBsDw",
      "e": "AQAB",
      "x5c": [
         "MIIDMDCCAhigAwIBAgIBBTANBgkqhkiG9w0BAQsFADBCMQswCQYDVQQGEwJubDER
          MA8GA1UEChMIbGllc2RvbmsxDTALBgNVBAsTBGNkbWkxETAPBgNVBAMTCHJzYS1y
          b290MCAXDTE2MTAyNzEyNDUwMFoYDzk5OTkxMjMxMjM1OTU5WjA1MQswCQYDVQQG
          EwJ1czEOMAwGA1UEChMFaG9zLWIxFjAUBgNVBAMTDWNkbWkuaG9zLWIudXMwggEi
          MA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQChAxCRjzlTPA7srofRBA18CJEw
          3dl+ErRpcQrHtjC+sJT0go6q5nU5hn0K\/znnr\/wTzJmbTYkNY4E5pKwYdTzfmO
          9d\/XZ\/gzLAoZaPkW0gYv6P2RcXpJarfO4tViVekwgqwmeAcxUVxr\/5mFna8uy
          ucNHVoI0QLjyBHbp7CnpZfWm2dJzVbjHcyP440xXPFUTtc2XBjKxOAO2ieZNAP5g
          br8qGEA\/HRijAxtRa\/9wxC9eyLswwPJmfp\/F+rjNCZDZXlG1TLmax3ordt4PN
          LguZmGwmaNTtieE6PSllvLhlySbjeemNc8C7A+HMrY5bupt+S\/SqbiuwVZRy42H
          yUGwPAgMBAAGjPDA6MAwGA1UdEwEB\/wQCMAAwHQYDVR0OBBYEFH7NJvMIfTQtZn
          nyiIdLNkjCgwSIMAsGA1UdDwQEAwIEMDANBgkqhkiG9w0BAQsFAAOCAQEAdiADIv
          0v09SUDcPL+BKysvchn\/Sgx5KBu7n9KFwE31Dhx2zvT6ruL8kXdekPH9cfrDafW
          6I\/vnbzAVj02i5pM2cHayj13fTOWSVwpcQuvkoIF9eVIWONkemMMf7M7jpTw07z
          7S2T5usaDmMNpqj8y5pRpQo3PnBVxpEZJ0XaSdfuiHtVLDq8gDZCq6Hc2tt7JM3W
          njnQgs+1lSGRuqWOcpmVONIoqvhiolNDZV35Z7puRwqck1N2f1qyHHGBWXfCN9U4
          ci6q1BnWBIFV+hURge8NSbpqawolaNueUbTcKjN3JsMC4ZxhMF9rN3uuPn+UAYkA
          yQkcSmGSMM07wcAkMg==",
         "MIIDQDCCAiigAwIBAgIBATANBgkqhkiG9w0BAQsFADBCMQswCQYDVQQGEwJubDER
          MA8GA1UEChMIbGllc2RvbmsxDTALBgNVBAsTBGNkbWkxETAPBgNVBAMTCHJzYS1y
          b290MCAXDTE2MTAyNzEyNDQwMFoYDzk5OTkxMjMxMjM1OTU5WjBCMQswCQYDVQQG
          EwJubDERMA8GA1UEChMIbGllc2RvbmsxDTALBgNVBAsTBGNkbWkxETAPBgNVBAMT
          CHJzYS1yb290MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAsrUj46dx
          5ojlaZk7YtOL6e+Q6JoG7gVMaXkJn1Szlx9ND\/8w4PeO1SQ2skukdOHAlQRmxdf
          tzhccNTM5hmbcn8TAfWSYqQF1R7s78bVjtmat6AQP1vSgiyZ8Ak+iYZEq3c2zVyY
          QHKKxWxmFZt1HT8\/H\/B3bXveXQcERKE+Tq66h8pqVcocQUtzRFsEYmv0bR1rgh
```


(continues on next page)

(continued from previous page)

```
            toqH8nhB5xnebgVlXjApW+et2SE7r6Fjv1aAbGI89ouJ1gsMPeX56P8AUjacFtNK
            c44Obu6HRXY\/jm6f2m1EUm84EUsJ+9b5+S2x4qPttJDfSCasWYYz4mFJ8MwmFiB
            GUwfgeT2bUm6t7qqbQIDAQABoz8wPTAPBgNVHRMBAf8EBTADAQH\/MB0GA1UdDgQ
            WBBR+tEB2udkEXxX0k15GztF\/4ol03jALBgNVHQ8EBAMCAQYwDQYJKoZIhvcNAQ
            ELBQADggEBAIjx1f9rJ2B+mDSA3L2GRhjrPRjfI6Un3Z51CeW9gO9PMQ5ws5pDJy
            B79dE\/Q8Uf1e8pZyjchTsRa8GRdnKyndN2imayOVUvPoTd3\/ZSmfkurcbj3I4V
            W8sjHP7CE8fmUS8Xprdpo2SxV7oneJC0vt5eyh8mgfJ\/qSbwVaiXuH1Wxi6duAv
            dxddMXAxQKPG1KKVM7CYfCdpX\/HagCOHzcto+374zFqqnQ1Kx5rbgvxNSgm\/PD
            DOMwP03+bbTR63KSK1VbdtLBuS4jgaPabwyxQz\/FciwTu\/HLOQn8TNqDWyoIbs
            +eQX2Mds2Apul8XH2+CakjBLMLL3Tlj2x+6tKR9o="
        ]
    }
}
```

The CDMI server at Hospital B can now decrypt this message, process the access control decision, and use the object key to decrypt the encrypted object:

```
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 33

This is an unencrypted text file.
```

Clause 25

Data Object Versions

25.1 Overview

Version-enabled data objects allow the previous state of a data object to be retained when an update is performed. In a non-version-enabled data object, each update changes the state of the object, and the previous state is lost. This state change is shown in Fig. 17.

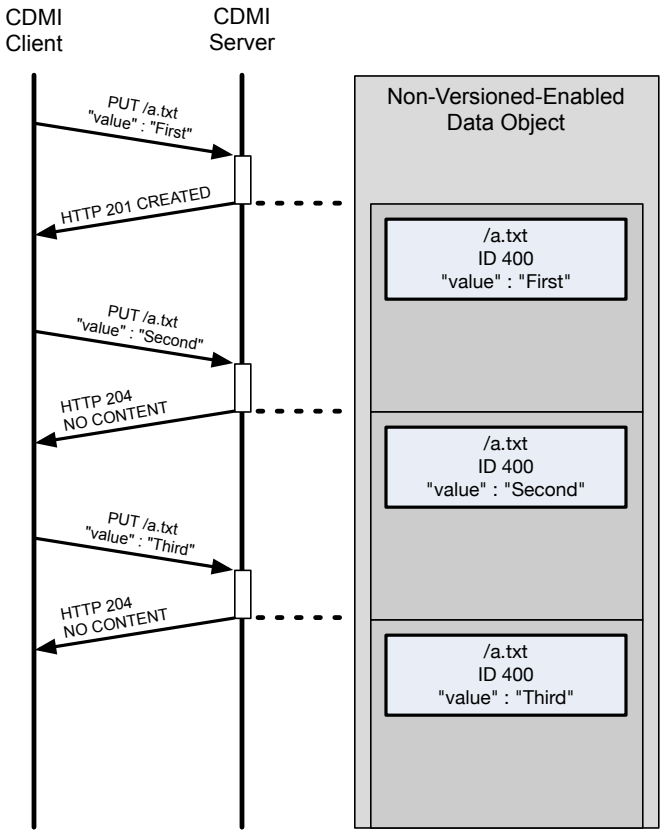


Fig. 17: Updates to a Non-Version-Enabled Data Object

When a data object has versioning enabled, each update creates a new “current version” with the same contents of the version-enabled data object, and the previous current version becomes a historical version. All versions can be accessed via separate URIs and are immutable. The version-enabled data object continues to be mutable and has the

same behaviors to clients as a non-version-enabled data object. This behavior is shown in Fig. 18 from the perspective of a client.

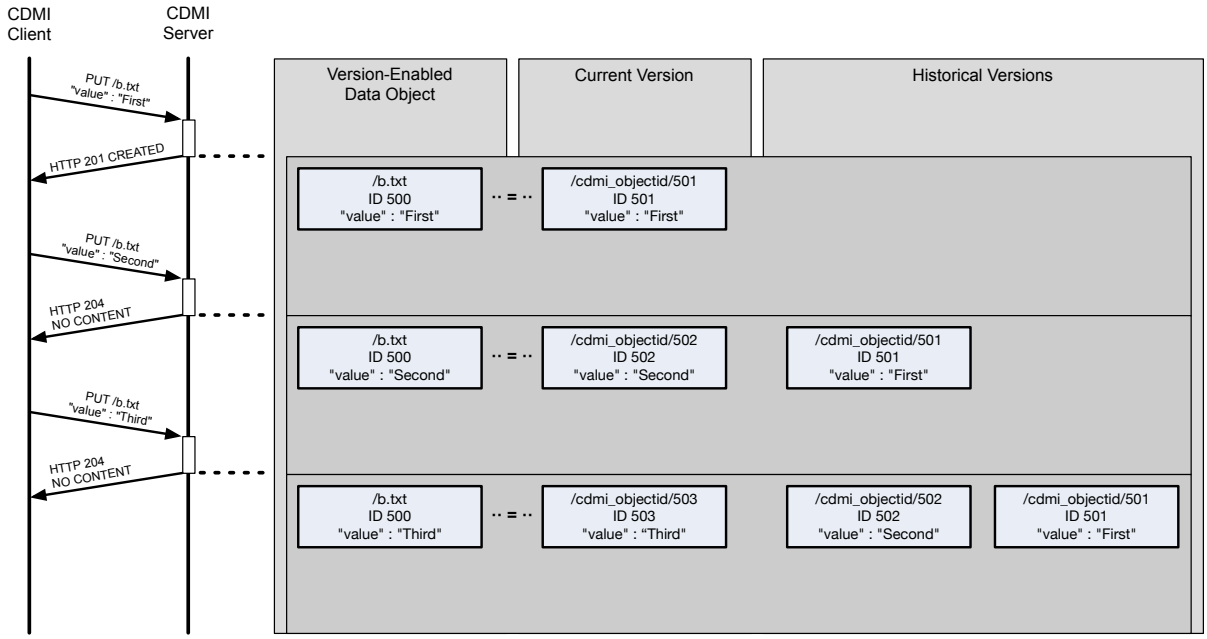


Fig. 18: Updates to a Version-Enabled Data Object

Using this approach, CDMI clients that are not aware of versioning can continue to access version-enabled data objects the same way as non-version-enabled data objects, while CDMI clients that are aware of versioning can access and manage the immutable versions associated with the version-enabled data object.

Versioning is enabled for a data object by adding a data system metadata item that indicates that versioning is desired.

Version-enabled data objects and all associated versions contain additional storage system metadata items. These metadata items allow a client to discover the versions that are associated with a version-enabled data object and to iterate through these versions.

The maximum number of versions to be retained, maximum age of versions to be retained, and the maximum space that can be consumed by versions is controlled by data system metadata.

When a data object is version enabled, it always contains at least one version, the “current version”. The current version has the same contents as the version-enabled data object but has a different identifier (URI and Object Identifier) and is immutable. When a version-enabled data object is changed, a new current version is created, and the previous current version becomes a historical version.

Versioning has multiple client use cases:

- Clients that need to preserve all data written to a data object over time can use versions to retain all updates made to a data object.
- Clients can restore the contents of a historical version by copying it to the version-enabled data object.
- Clients that retrieve a large data object across multiple parallel or sequential transactions or that need to be able to resume a retrieval at a later time can retrieve the URI for the current version of the data object. Clients can then use that URI to retrieve the data object itself. As the current version is immutable and retains its identifier, even if an update occurs (where the current version becomes a historical version), the client will always receive the same results and will not receive a mixture of the older and newer data object contents.
- Clients can iterate through historical versions to detect where concurrent updates have occurred and can access any overwritten data.
- Distributed CDMI implementations can also use versions to merge concurrent changes made on different, eventually consistent nodes without resulting in data loss.

25.2 Traversing Version-Enabled Data Objects

Version-enabled data objects have multiple metadata items that allow a client to traverse through the data object versions.

When a client enables versioning for a data object, the following metadata items shall be added to the version-enabled data object:

- a `cdmi_version_object` metadata item that contains the URI to the corresponding version-enabled data object. This metadata item allows a client to detect that a given object is a version-enabled data object and not a data object version.
- a `cdmi_version_current` field that contains the URI to the current version of the version-enabled data object.
- a `cdmi_version_oldest` field that contains the URI of one or more of the oldest versions. More than one version can exist in this metadata item as explained in Section 25.3.

Each data object version shall contain the above three fields, with the same values as found in the version-enabled data object. Each data object version shall also contain the following two fields:

- a `cdmi_version_parent` field that contains the URI of the previous version. If the data object version does not have a parent, this field is omitted.
- `cdmi_version_children` field that contains the URIs of the versions created by modifying this version. If the data object version does not have any children, this metadata item shall be empty.

To visualize how these fields allow a client to traverse data object versions, the linkages between the version-enabled data object and data object versions in the final state of Fig. 18 is shown in Fig. 19.

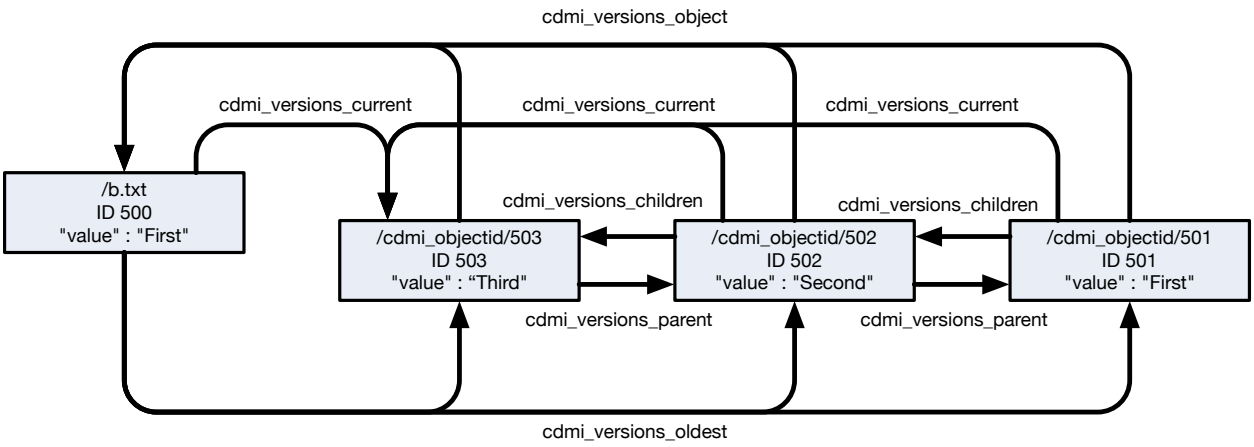


Fig. 19: Linkages Between a Version-Enabled Data Object and Data Object Versions

A client accessing the version-enabled data object (`/b.txt`) can traverse to the current version and to the oldest version.

A client accessing a data object version can traverse to the version-enabled data object, to the current version, to the parent version, to child versions, and to the oldest version.

25.3 Concurrent Updates and Version-Enabled Data Objects

When multiple concurrent updates are performed against a version-enabled data object, each update is performed against the state of the object at the time the update starts. The change to the state resulting from the update to the object becomes visible to clients at the time the update completes.

Two different types of concurrent updates can occur: overlapping updates and nested updates. Fig. 20 and Fig. 21 show the update sequence and resulting version linkages for overlapping updates:

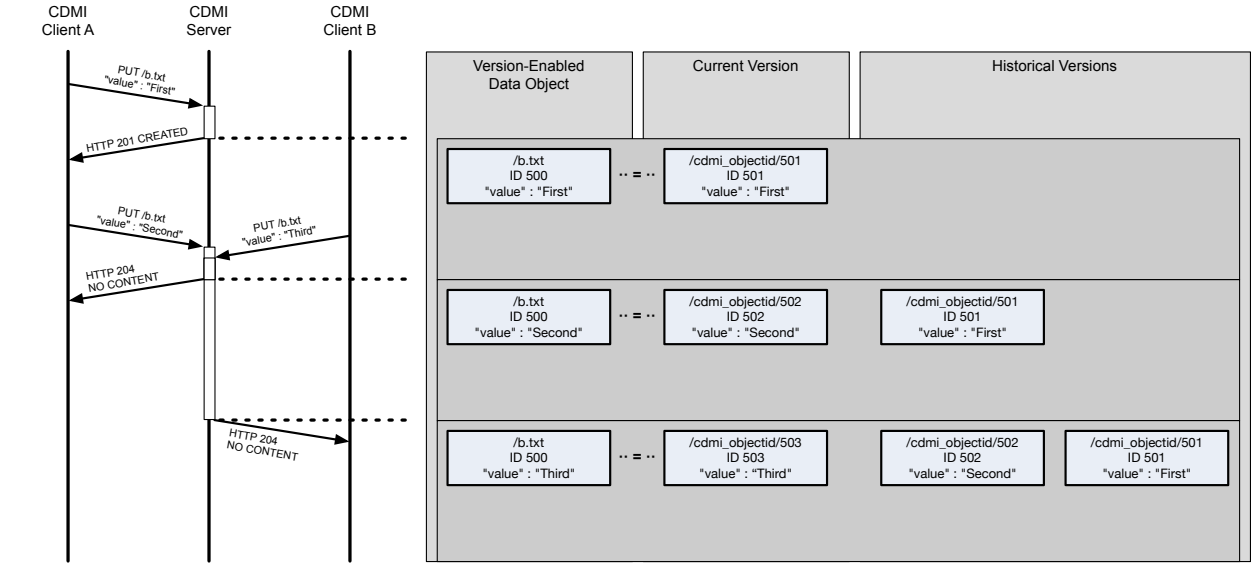


Fig. 20: Overlapping Concurrent Updates

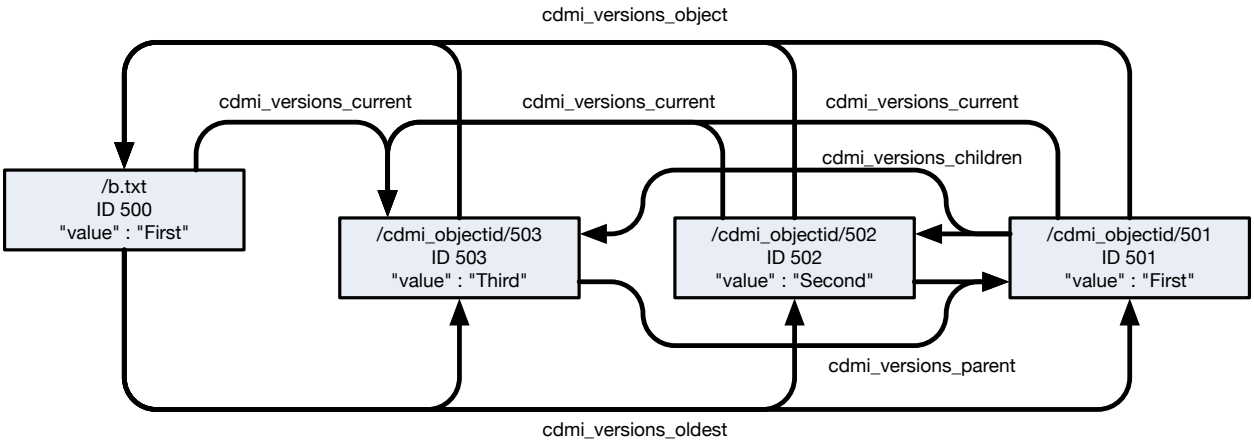


Fig. 21: Linkages for Overlapping Updates

In the sequence shown in Fig. 20, both the “Second” and “Third” updates are performed against the “First” state. As the “Third” update completes last, it becomes the current version. In this example, historical version 501 would have two children, versions `502` and `503`. Both versions `502` and `503` would have the same parent `501`.

4121 Fig. 22 and Fig. 23 show the update sequence and resulting version linkages for nested updates:

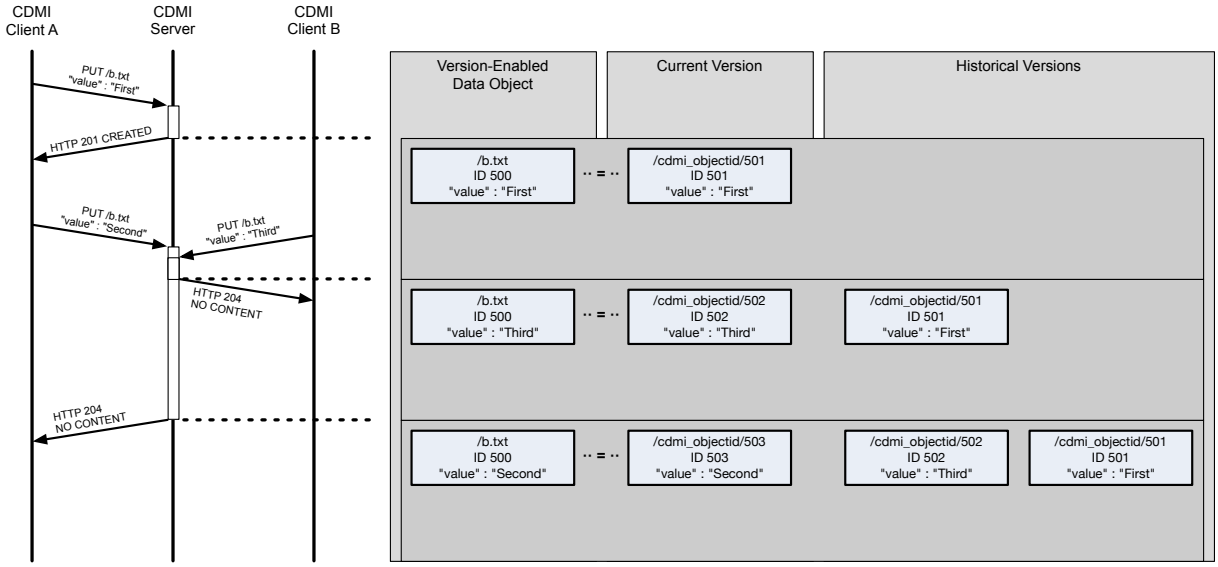


Fig. 22: Nested Concurrent Updates

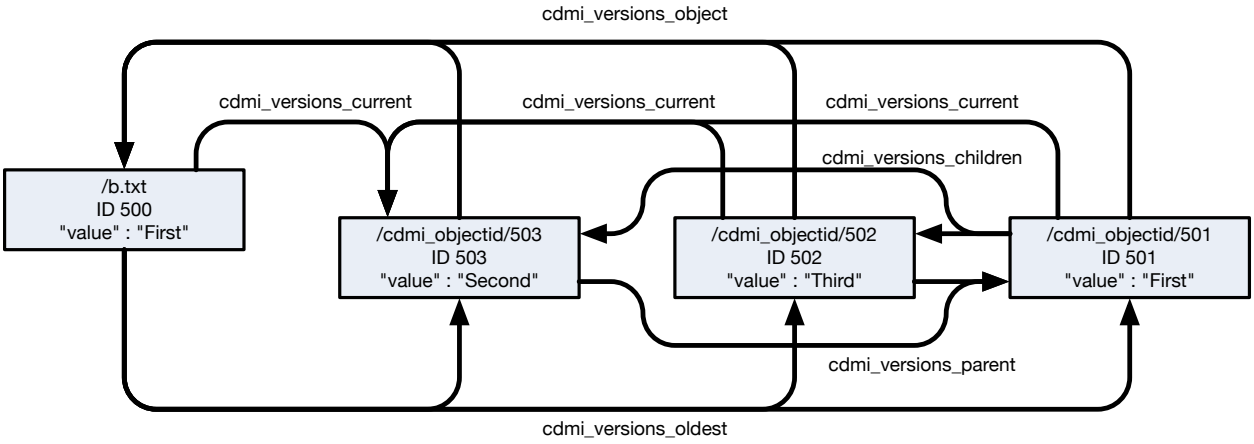


Fig. 23: Linkages for Nested Updates

4122 In the sequence shown in Figure 16, both the “Second” and “Third” updates are performed against the “First” state. As
4123 the “Second” update completes last, it becomes the current version. In this example, historical version `501` would have
4124 two children, versions `502` and `503`. Both versions `502` and `503` would have the same parent `501`.

4125 Both of these data structures are equivalent, with the only difference being which update completed last.

25.4 Capabilities for Version-Enabled Data Objects

The relationship between version-enabled data objects, data object versions, and capabilities is shown in `ref_version_to_capabilityuri_relationships`.

Section V

CDMI Annexes

Clause 26

(Informative) Extensions

26.1 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.

26.2 Summary Metadata for Bandwidth

26.2.1 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.

26.2.2 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 62 in `ref_domain_object_summaries` as follows:

4155

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

26.3 Expiring Access Control Entries (ACEs)

26.3.1 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.

26.3.2 Changes to CDMI 1.1

1. Insert into Section 17.1.6:
   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 Section 17.1.6:
   Original text:

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

   Revised text:

   ACE = { acetype , identifier , aceflags , acemask , acetime, expiration }
3. Insert into Section 17.1.6 after "`acemask = uint_t | acemaskstring`":
   expiration = uint_t
4. Insert into Section 17.1.6 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 Section 5.6.
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 Section 5.6, which specifies that the ACE is no longer valid and shall be ignored after the time specified.

26.4 Group Storage System Metadata

26.4.1 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.

26.4.2 Changes to CDMI 1.1

1. Add a table entry to the end of Table 99 in Section 12.1.9.

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 111 of Section 16.1.

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

S

U

V

W

X