

BS ISO/IEC 17826:2016



BSI Standards Publication

Information technology — Cloud Data Management Interface (CDMI)



National foreword

This British Standard is the UK implementation of ISO/IEC 17826:2016. It supersedes BS ISO/IEC 17826:2012 which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee IST/38, Cloud Computing and Distributed Platforms.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2016.
Published by BSI Standards Limited 2016

ISBN 978 0 580 92549 8
ICS 35.040

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 July 2016.

Amendments/corrigenda issued since publication

Date	Text affected
------	---------------

INTERNATIONAL STANDARD

ISO/IEC
17826

Second edition
2016-07-15

Information technology — Cloud Data Management Interface (CDMI)

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



Reference number
ISO/IEC 17826:2016(E)

AVA Documents
Document Sharing Hub
© ISO/IEC 2016



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2016, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

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

Foreword

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

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

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

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

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

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

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



Cloud Data Management Interface (CDMI™)

Version 1.1.1

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

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

SNIA Technical Position

March 19, 2015

USAGE

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

- 1 Any text, diagram, chart, table or definition reproduced shall be reproduced in its entirety with no alteration, and,
- 2 Any document, printed or electronic, in which material from this document (or any portion hereof) is reproduced shall acknowledge the SNIA copyright on that material, and shall credit the SNIA for granting permission for its reuse.

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

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

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

BSD 3-Clause Software License

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

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

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

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

DISCLAIMER

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

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

Copyright © 2015 SNIA. All rights reserved. All other trademarks or registered trademarks are the property of their respective owners.

Revision History

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

Contents

SECTION 1 - CDMI Preamble	1
Introduction	2
1 Scope	4
2 Normative references	4
3 Terms, acronyms, and definitions	6
4 Conventions	10
4.1 Interface format	10
4.2 Typographical conventions	10
4.3 Request and response body requirements	11
4.4 Key word requirements	11
5 Overview of cloud storage	13
5.1 Introduction	13
5.2 What is cloud storage?	13
5.3 Data storage as a Service	13
5.4 Data management for cloud storage	15
5.5 Data and container management	16
5.6 Reference model for cloud storage interfaces	16
5.7 Cloud Data Management Interface	17
5.8 Object model for CDMI	18
5.9 CDMI metadata	19
5.10 Object ID	20
5.11 CDMI object ID format	20
5.12 Security	21
5.12.1 Security objectives	21
5.12.2 HTTP security	22
5.12.3 Client authentication	22
5.12.4 Use of TLS and HTTP	23
5.12.5 Further information	23
5.13 Required HTTP support	23
5.13.1 RFC 2616 support requirements	23
5.13.2 Content-type negotiation	23
5.13.3 Range support	23
5.13.4 URI escaping	24
5.13.5 Use of URIs	24
5.13.6 Reserved characters	25
5.14 Time representations	25
5.15 Backwards compatibility	25
5.15.1 Value transfer encoding	25
5.15.2 Container export capabilities	26
5.16 Object references	26
SECTION 2 - Basic Cloud Storage	28
6 Data object resource operations using HTTP	29
6.1 Overview	29
6.2 Create a data object using HTTP	29
6.2.1 Synopsis	29

6.2.2	Capabilities	29
6.2.3	Request headers	30
6.2.4	Request message body	30
6.2.5	Response headers	30
6.2.6	Response message body	30
6.2.7	Response status	30
6.2.8	Example	31
6.3	Read a data object using HTTP	31
6.3.1	Synopsis	31
6.3.2	Capabilities	31
6.3.3	Request header	31
6.3.4	Request message body	31
6.3.5	Response headers	32
6.3.6	Response message body	32
6.3.7	Response status	32
6.3.8	Examples	32
6.4	Update a data object using HTTP	33
6.4.1	Synopsis	33
6.4.2	Capabilities	33
6.4.3	Request headers	33
6.4.4	Request message body	34
6.4.5	Response header	34
6.4.6	Response message body	34
6.4.7	Response status	34
6.4.8	Examples	34
6.5	Delete a data object using HTTP	35
6.5.1	Synopsis	35
6.5.2	Capability	35
6.5.3	Request headers	35
6.5.4	Request message body	35
6.5.5	Response headers	35
6.5.6	Response message body	35
6.5.7	Response status	36
6.5.8	Example	36
7	Container object resource operations using HTTP	37
7.1	Overview	37
7.2	Create a container object using HTTP	37
7.2.1	Synopsis	37
7.2.2	Capability	37
7.2.3	Request headers	38
7.2.4	Request message body	38
7.2.5	Response headers	38
7.2.6	Response message body	38
7.2.7	Response status	38
7.2.8	Example	38
7.3	Read a container object using HTTP	38
7.4	Update a container object using HTTP	38
7.5	Delete a container object using HTTP	39
7.5.1	Synopsis	39
7.5.2	Capability	39
7.5.3	Request headers	39
7.5.4	Request message body	39
7.5.5	Response headers	39
7.5.6	Response message body	39
7.5.7	Response status	40
7.5.8	Example	40
7.6	Create (POST) a new data object using HTTP	40

7.6.1	Synopsis	40
7.6.2	Capabilities	41
7.6.3	Request headers	41
7.6.4	Request message body	41
7.6.5	Response header	41
7.6.6	Response message body	42
7.6.7	Response status	42
7.6.8	Examples	42
SECTION 3 - CDMI Core		43
8	Data object resource operations using CDMI	44
8.1	Overview	44
8.1.1	Data object metadata	45
8.1.2	Data object consistency	45
8.1.3	Data object representations	46
8.2	Create a data object using CDMI	46
8.2.1	Synopsis	46
8.2.2	Delayed completion of create	46
8.2.3	Capabilities	47
8.2.4	Request headers	47
8.2.5	Request message body	48
8.2.6	Response headers	51
8.2.7	Response message body	51
8.2.8	Response status	52
8.2.9	Examples	52
8.3	Read a data object using CDMI	55
8.3.1	Synopsis	55
8.3.2	Capabilities	55
8.3.3	Request headers	56
8.3.4	Request message body	56
8.3.5	Response headers	56
8.3.6	Response message body	57
8.3.7	Response status	59
8.3.8	Examples	59
8.4	Update a data object using CDMI	62
8.4.1	Synopsis	62
8.4.2	Capabilities	63
8.4.3	Request headers	63
8.4.4	Request message body	64
8.4.5	Response header	67
8.4.6	Response message body	67
8.4.7	Response status	67
8.4.8	Examples	67
8.5	Delete a data object using CDMI	71
8.5.1	Synopsis	71
8.5.2	Capability	71
8.5.3	Request header	71
8.5.4	Request message body	71
8.5.5	Response headers	71
8.5.6	Response message body	71
8.5.7	Response status	72
8.5.8	Example	72
9	Container object resource operations using CDMI	73
9.1	Overview	73

9.1.1	Container metadata	74
9.1.2	Reserved container names	74
9.1.3	Container object addressing	74
9.1.4	Container object representations	75
9.2	Create a container object using CDMI	75
9.2.1	Synopsis	75
9.2.2	Delayed completion of create	75
9.2.3	Capabilities	76
9.2.4	Request headers	76
9.2.5	Request message body	77
9.2.6	Response headers	79
9.2.7	Response message body	79
9.2.8	Response status	81
9.2.9	Examples	81
9.3	Read a container object using CDMI	83
9.3.1	Synopsis	83
9.3.2	Capabilities	83
9.3.3	Request headers	84
9.3.4	Request message body	84
9.3.5	Response headers	84
9.3.6	Response message body	84
9.3.7	Response status	86
9.3.8	Examples	86
9.4	Update a container object using CDMI	88
9.4.1	Synopsis	88
9.4.2	Delayed completion of snapshot	88
9.4.3	Capabilities	89
9.4.4	Request headers	89
9.4.5	Request message body	90
9.4.6	Response header	92
9.4.7	Response message body	92
9.4.8	Response status	92
9.4.9	Examples	92
9.5	Delete a container object using CDMI	93
9.5.1	Synopsis	93
9.5.2	Capability	93
9.5.3	Request header	94
9.5.4	Request message body	94
9.5.5	Response headers	94
9.5.6	Response message body	94
9.5.7	Response status	94
9.5.8	Example	94
9.6	Create (POST) a new data object using CDMI	95
9.6.1	Synopsis	95
9.6.2	Delayed completion of create	95
9.6.3	Capabilities	96
9.6.4	Request headers	97
9.6.5	Request message body	98
9.6.6	Response headers	101
9.6.7	Response message body	101
9.6.8	Response status	102
9.6.9	Examples	103
9.7	Create (POST) a new queue object using CDMI	105
9.7.1	Synopsis	105
9.7.2	Delayed completion of create	105
9.7.3	Capabilities	106
9.7.4	Request headers	107
9.7.5	Request message body	107

9.7.6	Response headers	108
9.7.7	Response message body	108
9.7.8	Response status	110
9.7.9	Example	110

SECTION 4 - CDMI Advanced 112

10	Domain object resource operations using CDMI	113
10.1	Overview	113
10.1.1	Domain object metadata	114
10.1.2	Domain object summaries	114
10.1.3	Domain object membership	117
10.1.4	Domain usage in access control	119
10.1.5	Domain object representations	120
10.2	Create a domain object using CDMI	120
10.2.1	Synopsis	120
10.2.2	Capabilities	120
10.2.3	Request headers	120
10.2.4	Request message body	121
10.2.5	Response headers	122
10.2.6	Response message body	122
10.2.7	Response status	123
10.2.8	Example	123
10.3	Read a domain object using CDMI	124
10.3.1	Synopsis	124
10.3.2	Capabilities	124
10.3.3	Request headers	124
10.3.4	Request message body	124
10.3.5	Response headers	125
10.3.6	Response message body	125
10.3.7	Response status	126
10.3.8	Examples	126
10.4	Update a domain object using CDMI	127
10.4.1	Synopsis	127
10.4.2	Capability	128
10.4.3	Request headers	128
10.4.4	Request message body	128
10.4.5	Response header	129
10.4.6	Response message body	129
10.4.7	Response status	130
10.4.8	Example	130
10.5	Delete a domain object using CDMI	130
10.5.1	Synopsis	130
10.5.2	Capability	131
10.5.3	Request header	131
10.5.4	Request message body	131
10.5.5	Response headers	131
10.5.6	Response message body	131
10.5.7	Response status	131
10.5.8	Example	132
11	Queue object resource operations using CDMI	133
11.1	Overview	133
11.1.1	Queue object metadata	134
11.1.2	Queue object addressing	134
11.1.3	Queue object representations	134

11.2	Create a queue object using CDMI	134
11.2.1	Synopsis	134
11.2.2	Delayed completion of create	135
11.2.3	Capabilities	135
11.2.4	Request headers	136
11.2.5	Request message body	136
11.2.6	Response headers	138
11.2.7	Response message body	138
11.2.8	Response status	139
11.2.9	Examples	140
11.3	Read a queue object using CDMI	141
11.3.1	Synopsis	141
11.3.2	Capabilities	141
11.3.3	Request headers	142
11.3.4	Request message body	142
11.3.5	Response headers	142
11.3.6	Response message body	142
11.3.7	Response status	145
11.3.8	Examples	145
11.4	Update a queue object using CDMI	148
11.4.1	Synopsis	148
11.4.2	Capability	148
11.4.3	Request headers	148
11.4.4	Request message body	149
11.4.5	Response header	150
11.4.6	Response message body	150
11.4.7	Response status	150
11.4.8	Examples	150
11.5	Delete a queue object using CDMI	151
11.5.1	Synopsis	151
11.5.2	Capability	151
11.5.3	Request header	151
11.5.4	Request message body	152
11.5.5	Response headers	152
11.5.6	Response message body	152
11.5.7	Response status	152
11.5.8	Example	152
11.6	Enqueue a new queue value using CDMI	152
11.6.1	Synopsis	152
11.6.2	Capabilities	153
11.6.3	Request headers	153
11.6.4	Request message body	153
11.6.5	Response headers	155
11.6.6	Response message body	155
11.6.7	Response status	156
11.6.8	Examples	156
11.7	Delete a queue object value using CDMI	159
11.7.1	Synopsis	159
11.7.2	Capability	159
11.7.3	Request header	159
11.7.4	Request message body	159
11.7.5	Response headers	160
11.7.6	Response message body	160
11.7.7	Response status	160
11.7.8	Examples	160
12	Capability object resource operations using CDMI	161
12.1	Overview	161

12.1.1	Cloud storage system-wide capabilities	162
12.1.2	Storage system metadata capabilities	165
12.1.3	Data system metadata capabilities	165
12.1.4	Data object capabilities	168
12.1.5	Container capabilities	169
12.1.6	Domain object capabilities	171
12.1.7	Queue object capabilities	172
12.1.8	Capability object representations	172
12.2	Read a capabilities object using CDMI	173
12.2.1	Synopsis	173
12.2.2	Capability	173
12.2.3	Request headers	173
12.2.4	Request message body	173
12.2.5	Response headers	174
12.2.6	Response message body	174
12.2.7	Response status	175
12.2.8	Examples	175
13	Exported protocols	177
13.1	Overview	177
13.2	Exported protocol structure	178
13.2.1	Mapping names from CDMI to another protocol	179
13.2.1.1	Capabilities	179
13.2.1.2	Domains	179
13.2.1.3	Caching	179
13.2.1.4	Groups	180
13.2.1.5	Synopsis	180
13.2.2	Administrative users	181
13.2.3	User and groupname mapping syntax and evaluation rules	182
13.3	Discovering and mounting containers via foreign protocols	183
13.4	NFS exported protocol	184
13.5	CIFS exported protocol	186
13.6	OCCL exported protocol	187
13.7	iSCSI export modifications	187
13.7.1	Read container	187
13.7.2	Create and update containers	188
13.7.3	Modify an export	188
13.8	WebDAV exported protocol	188
14	CDMI snapshots	190
15	Serialization/deserialization	191
15.1	Overview	191
15.2	Exporting serialized data	191
15.3	Importing serialized data	191
15.3.1	Canonical format	192
15.3.2	Example JSON canonical serialized format	192
16	Metadata	194
16.1	Access control	194
16.1.1	ACL and ACE structure	194
16.1.2	ACE types	194
16.1.3	ACE who	194
16.1.4	ACE flags	195
16.1.5	ACE bit masks	196
16.1.6	ACL evaluation	198
16.1.7	Example ACE mask expressions	200
16.1.8	Canonical format for ACE hexadecimal quantities	201

16.1.9	JSON format for ACLs	201
16.2	Support for user metadata	202
16.3	Support for storage system metadata	202
16.4	Support for data system metadata	204
16.5	Support for provided data system metadata	210
16.6	Metadata update operations	211
17	Retention and hold management	212
17.1	Introduction	212
17.2	Retention management disciplines	212
17.3	CDMI retention	212
17.3.1	Overview	212
17.3.2	Examples	213
17.4	CDMI hold	214
17.4.1	Overview	214
17.4.2	Examples	216
17.5	CDMI auto-deletion	217
17.6	Retention security considerations	217
18	Scope specification	219
18.1	Introduction	219
18.2	Examples	219
18.3	Query matching expressions	221
19	Results specification	226
19.1	Introduction	226
19.2	Examples	226
20	Logging	228
20.1	Overview	228
20.2	Object logging	228
20.3	Security logging	228
20.4	Data management logging	229
20.5	Logging queues	229
20.6	Logging security considerations	231
21	Notification queues	232
21.1	Overview	232
21.2	Required metadata	232
21.3	System-created metadata	235
22	Query queues	236
22.1	Overview	236
22.2	Required metadata	236
22.3	System-created metadata	237
22.4	Extending CDMI query	238
SECTION 5 - CDMI Annexes	239	
Annex A		
(informative)		
Extensions	240	
A.1	Overview	240
A.2	Summary metadata for bandwidth	240
A.2.1	Overview	240

A.2.2 Changes to CDMI 1.1	240
A.3 Expiring Access Control Entries (ACEs)	242
A.3.1 Overview	242
A.3.2 Changes to CDMI 1.1	242
A.4 Group storage system metadata	243
A.4.1 Overview	243
A.4.2 Changes to CDMI 1.1	243
A.5 Versioning	244
A.5.1 Overview	244
A.5.2 Changes to CDMI 1.1	244
Bibliography	259
Bibliography	264

Figures

Figure 1 – Existing data storage interface standards	14
Figure 2 – Storage interfaces for object storage client data	15
Figure 3 – Cloud storage reference model	16
Figure 4 – CDMI object model	18
Figure 5 – Object transitions between named and ID-only	19
Figure 6 – Object ID format	20
Figure 7 – Hierarchy of capabilities	161
Figure 8 – CDMI and OCCI in an integrated cloud computing environment	177
Figure 9 – Snapshot container structure	190
Figure 10 – Object retention	213
Figure 11 – Object hold	215
Figure 12 – Object hold on object with retention	215
Figure 13 – Object with multiple holds	215
Figure 14 – Updates to a non-version-enabled data object	249
Figure 15 – Updates to a version-enabled data object	250
Figure 16 – Linkages between a version-enabled data object and data object versions	251
Figure 17 – Overlapping concurrent updates	252
Figure 18 – Linkages for overlapping updates	252
Figure 19 – Nested concurrent updates	253
Figure 20 – Linkages for nested updates	253
Figure 21 – Version to capabilityURI relationships	254

Tables

Table 1 – Interface format	10
Table 2 – Key word requirements	11
Table 3 – Types of resources in the model	18
Table 4 – Creation/consumption of storage system metadata	19
Table 5 – Relative URIs resolved against root URIs	24
Table 6 – Request headers - Create a data object using HTTP	30
Table 7 – HTTP status codes - Create a data object using HTTP	30
Table 8 – Request header - Read a data object using HTTP	31
Table 9 – Response headers - Read a data object using HTTP	32
Table 10 – HTTP status codes - Read a data object using HTTP	32
Table 11 – Request headers - Update a data object using HTTP	33
Table 12 – Response header - Update a data object using HTTP	34
Table 13 – HTTP status codes - Update a data object using HTTP	34
Table 14 – HTTP status codes - Delete a data object using HTTP	36
Table 15 – HTTP status codes - Create a container object using HTTP	38
Table 16 – HTTP status codes - Delete a container object using HTTP	40
Table 17 – Request headers - Create a new data object using HTTP	41
Table 18 – Response header - Create a new data object using HTTP	41
Table 19 – HTTP status codes - Create a new data object using HTTP	42
Table 20 – Request headers for creating a data object using CDMI	47
Table 21 – Request message body - Create a data object using CDMI	48
Table 22 – Response headers - Create a data object using CDMI	51
Table 23 – Response message body - Create a data object using CDMI	51
Table 24 – HTTP status codes - Create a data object using CDMI	52
Table 25 – Request headers - Read a data object using CDMI	56
Table 26 – Response headers - Read a data object using CDMI	56
Table 27 – Response message body - Read a data object using CDMI	57
Table 28 – HTTP status codes - Read a data object using CDMI	59
Table 29 – Request headers - Update a data object using CDMI	63
Table 30 – Request message body - Update a data object using CDMI	64
Table 31 – Response header - Update a data object using CDMI	67
Table 32 – HTTP status codes - Update a data object using CDMI	67
Table 33 – Request header - Delete a data object using CDMI	71
Table 34 – HTTP status codes - Delete a data object using CDMI	72
Table 35 – Container metadata	74
Table 36 – Request headers - Create a container object using CDMI	76
Table 37 – Request message body - Create a container object using CDMI	77
Table 38 – Response headers - Create a container object using CDMI	79
Table 39 – Response message body - Create a container object using CDMI	79
Table 40 – HTTP status codes - Create a container object using CDMI	81
Table 41 – Request headers - Read a container object using CDMI	84
Table 42 – Response headers - Read a container object using CDMI	84
Table 43 – Response message body - Read a container object using CDMI	84
Table 44 – HTTP status codes - Read a container object using CDMI	86
Table 45 – Request headers - Update a container object using CDMI	89
Table 46 – Request message body - Update a container object using CDMI	90
Table 47 – Response header - Update a container object using CDMI	92
Table 48 – HTTP status codes - Update a container object using CDMI	92
Table 49 – Request header - Delete a container object using CDMI	94
Table 50 – HTTP status codes - Delete a container object using CDMI	94
Table 51 – Request headers - Create a new data object using CDMI	97
Table 52 – Request message body - Create a new data object using CDMI	98
Table 53 – Response headers - Create a new data object using CDMI	101
Table 54 – Response message body - Create a new data object using CDMI	101
Table 55 – HTTP status codes - Create a new data object using CDMI	102
Table 56 – Request headers - Create a new queue object using CDMI	107

Table 57 – Request message body - Create a new queue object using CDMI	107
Table 58 – Response headers - Create a new queue object using CDMI	108
Table 59 – Response message body - Create a new queue object using CDMI	108
Table 60 – HTTP status codes - Create a new queue object using CDMI	110
Table 61 – Required metadata for a domain object	114
Table 62 – Contents of domain summary objects	115
Table 63 – Required settings for domain member user objects	118
Table 64 – Required settings for domain member delegation objects	119
Table 65 – Request headers - Create a domain object using CDMI	120
Table 66 – Request message body - Create a domain object using CDMI	121
Table 67 – Response headers - Create a domain object using CDMI	122
Table 68 – Response message body - Create a domain object using CDMI	122
Table 69 – HTTP status codes - Create a domain object using CDMI	123
Table 70 – Request headers - Read a domain object using CDMI	124
Table 71 – Response headers - Read a domain object using CDMI	125
Table 72 – Response message body - Read a domain object using CDMI	125
Table 73 – HTTP status codes - Read a domain object using CDMI	126
Table 74 – Request headers - Update a domain object using CDMI	128
Table 75 – Request message body - Update a domain object using CDMI	128
Table 76 – Response header - Update a domain object using CDMI	129
Table 77 – HTTP status codes - Update a domain object using CDMI	130
Table 78 – Request header - Delete a domain object using CDMI	131
Table 79 – HTTP status codes - Delete a domain object using CDMI	131
Table 80 – Request headers - Create a queue object using CDMI	136
Table 81 – Request message body - Create a queue object using CDMI	136
Table 82 – Response headers - Create a queue object using CDMI	138
Table 83 – Response message body - Create a queue object using CDMI	138
Table 84 – HTTP status codes - Create a queue object using CDMI	139
Table 85 – Request headers - Read a queue object using CDMI	142
Table 86 – Response headers - Read a queue object using CDMI	142
Table 87 – Response message body - Read a queue object using CDMI	142
Table 88 – HTTP status codes - Read a queue object using CDMI	145
Table 89 – Request headers - Update a queue object using CDMI	148
Table 90 – Request message body - Update a queue object using CDMI	149
Table 91 – Response header - Update a queue object using CDMI	150
Table 92 – HTTP status codes - Update a queue object using CDMI	150
Table 93 – Request header - Delete a queue object using CDMI	151
Table 94 – HTTP status codes - Delete a queue object using CDMI	152
Table 95 – Request headers - Enqueue a new queue object value using CDMI	153
Table 96 – Request message body - Enqueue a new queue object value using CDMI	153
Table 97 – HTTP status codes - Enqueue a new queue object value using CDMI	156
Table 98 – Request header - Delete a queue object value using CDMI	159
Table 99 – HTTP status codes - Delete a queue object value using CDMI	160
Table 100 – System-wide capabilities	162
Table 101 – Capabilities for storage system metadata	165
Table 102 – Capabilities for data system metadata	166
Table 103 – Capabilities for data objects	168
Table 104 – Capabilities for containers	169
Table 105 – Capabilities for domain objects	171
Table 106 – Capabilities for queue objects	172
Table 107 – Request headers - Read a capabilities object using CDMI	173
Table 108 – Response headers - Read a capabilities object using CDMI	174
Table 109 – Response message body - Read a capabilities object using CDMI	174
Table 110 – HTTP status codes - Read a capabilities object using CDMI	175
Table 111 – Required members of the NFS protocol structure	184
Table 112 – Optional NFS export parameters	184
Table 113 – Required members of the CIFS protocol structure	186
Table 114 – ACE types	194

Table 115 – Who identifiers	195
Table 116 – ACE flags	195
Table 117 – ACE bit masks	196
Table 118 – Storage system metadata	202
Table 119 – Data system metadata	204
Table 120 – Provided values of data systems metadata items	210
Table 121 – Query matching expressions	221
Table 122 – Required metadata for a logging queue	230
Table 123 – Logging status metadata	231
Table 124 – Required metadata for a notification queue	232
Table 125 – Notification status metadata	235
Table 126 – Required metadata for a query queue	236
Table 127 – Query status metadata	237

Section I

CDMI Preamble

Introduction

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

This document is organized as follows:

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

20 - Logging	Describes CDMI functional logging for object functions, security events, data management events, and queues
21 - Notification Queues	Describes how CDMI clients may efficiently discover what changes have occurred to the system
22 - Query Queues	Describes how CDMI clients may efficiently discover what content matches a given set of metadata query criteria or full-content search criteria
Annex A - (informative) Extensions	Provides informative vendor extensions. Each extension is added to the standard when at least two vendors implement the extension.
Bibliography	Provides informative references that may contain additional useful information

1 Scope

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

2 Normative references

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3 Terms, acronyms, and definitions

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

3.1

Access Control List

ACL

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

3.2

API

Application Programming Interface

3.3

CDMI™

Cloud Data Management Interface

3.4

CDMI capability

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

3.5

CDMI container

object that stores zero or more children objects and associated metadata

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

3.6

CDMI data object

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

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

3.7

CDMI domain

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

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

3.8

CDMI object

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

3.9

CDMI queue

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

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

3.10

CIFS

Common Internet File System

3.11

cloud storage

see 3.4 Data Storage as a Service

3.12

CRC

cyclic redundancy check

3.13

CRUD

create, retrieve, update, delete

3.14

Data Storage as a Service

DSaaS

delivery of appropriately configured virtual storage and related data services over a network, based on a request for a given service level

3.15

domain

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

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

3.16

eventual consistency

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

3.17

FC

Fibre Channel

3.18

FCoE

Fibre Channel over Ethernet

3.19

HTTP

HyperText Transfer Protocol

3.20

iSCSI

Internet Small Computer Systems Interface

3.21

JSON

JavaScript Object Notation

3.22

LDAP

Lightweight Directory Access Protocol

3.23

LUN

Logical Unit Number

3.24

metadata

data about other data (see [ISO 14701:2012](#))

3.25

MIME

Multipurpose Internet Mail Extensions

3.26

NFS

Network File System

3.27

object

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

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

3.28

object identifier

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

3.29

OCCI

Open Cloud Computing Interface

3.30

POSIX

Portable Operating System Interface

3.31

Representational State Transfer

REST

specific set of principles for defining, addressing, and interacting with resources addressable by URIs

3.32

RPO

recovery point objective

3.33

RTO

recovery time objective

3.34

service level

performance targets for a service

3.35

SNMP

Simple Network Management Protocol

3.36

thin provisioning

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

3.37

Uniform Resource Identifier

URI

compact sequence of characters that identifies an abstract or physical resource

3.38

VIM

Vendor Interface Module

3.39

virtualization

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

3.40

WebDAV

Web Distributed Authoring and Versioning

3.41

XAM

eXtensible Access Method

4 Conventions

4.1 Interface format

Each interface description has nine components, as described in [Table 1](#).

Table 1 — Interface format

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

4.2 Typographical conventions

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

EXAMPLE 1

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

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

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

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

4.3 Request and response body requirements

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

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

4.4 Key word requirements

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

Table 2 — Key word requirements

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

Table 2 — Key word requirements

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

5 Overview of cloud storage

5.1 Introduction

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

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

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

5.2 What is cloud storage?

The use of the term *cloud* in describing these new models arose from architecture drawings that typically used a cloud as the icon for a network. The cloud represents any-to-any network connectivity in an abstract way. In this abstraction, the network connectivity in the cloud is represented without concern for how it is made to happen.

The cloud abstraction of complexity produces a simple base on which other features can be built. The general cloud model extends this base by adding a pool of resources. An important part of the cloud model is the concept of a pool of resources that is drawn from, on demand, in small increments. A relatively recent innovation that has made this possible is virtualization.

Thus, cloud storage is simply the delivery of virtualized storage on demand. The formal term that is used for this is *Data storage as a Service (DaaS)*.

5.3 Data storage as a Service

By abstracting data storage behind a set of service interfaces and delivering it on demand, a wide range of actual cloud services and implementations are possible. The only type of storage that is excluded from this definition is that which is delivered in fixed-capacity increments instead of that which is based on demand.

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

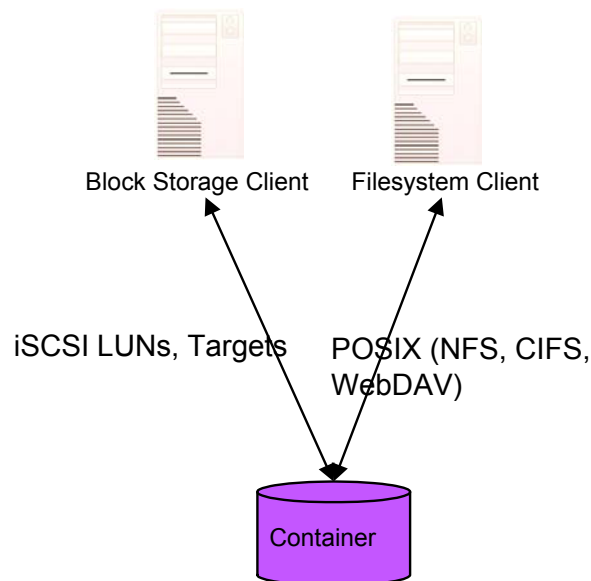


Figure 1 — Existing data storage interface standards

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

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

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

Each data object is created, retrieved, updated, and deleted as a separate resource. In this type of interface, a container, if used, is a simple grouping of data objects for convenience. Nothing prevents the

concept of containers from being hierarchical, although any given implementation might support only a single level (see [Figure 2](#)).

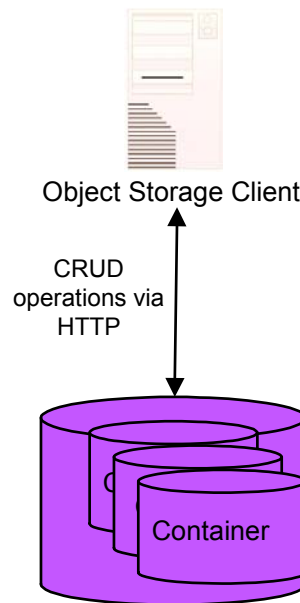


Figure 2 — Storage interfaces for object storage client data

5.4 Data management for cloud storage

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

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

By supporting metadata in a cloud storage interface and prescribing how the storage system and data system metadata is interpreted to meet the requirements of the data, the simplicity required by the cloud storage model can be maintained while still addressing the requirements of enterprise applications and their data.

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

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

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

5.5 Data and container management

There is no reason that managing data and managing containers should involve different interfaces. Therefore, the use of metadata is extended from applying to individual objects to applying to containers of objects as well. Thus, any data placed into a container inherits the data system metadata of the container into which it was placed. When creating a new container within an existing container, the new container would similarly inherit the metadata settings of its parent's data system metadata. After an object is created, the data system metadata can be overridden at the container or individual object level, as desired.

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

5.6 Reference model for cloud storage interfaces

The cloud storage reference model is shown in Figure 3.

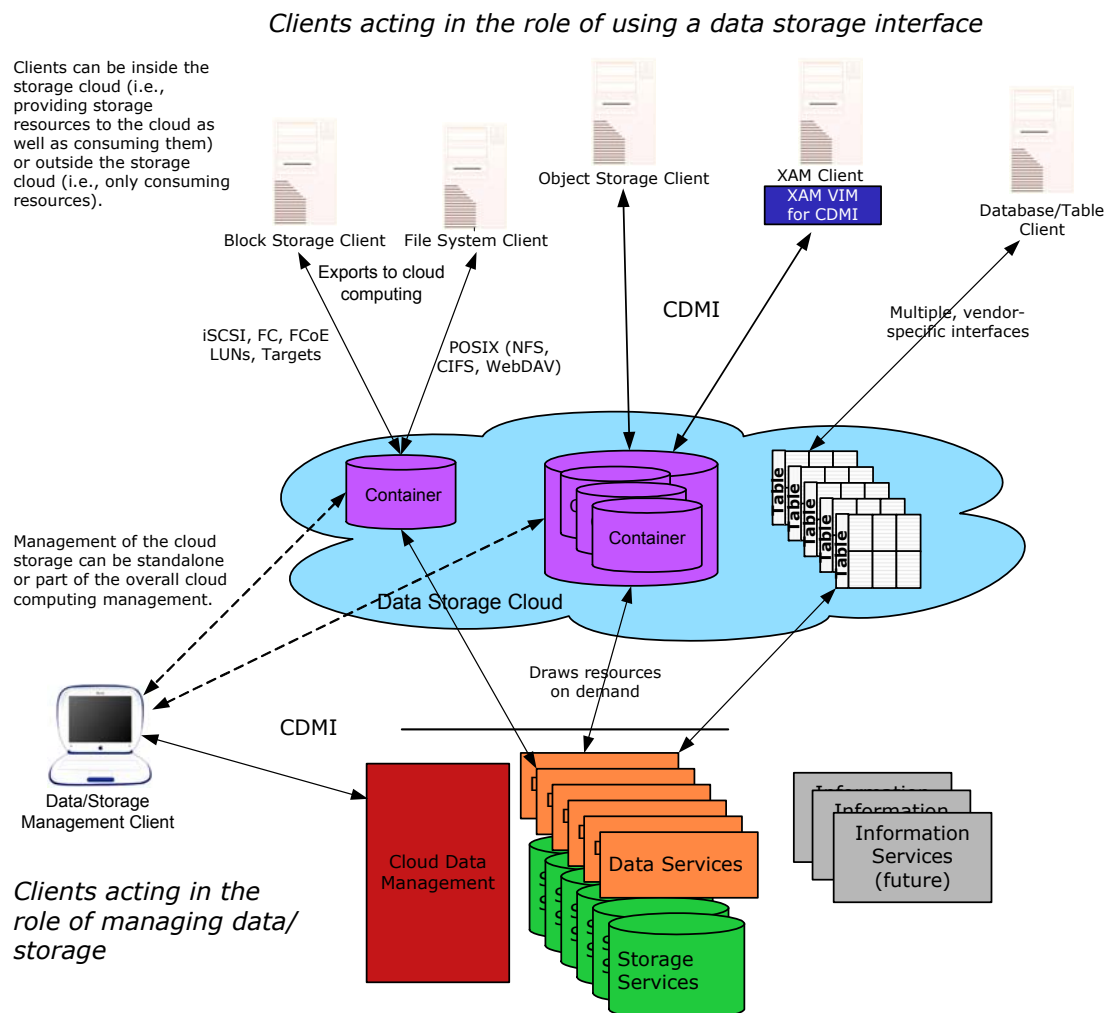


Figure 3 — Cloud storage reference model

This model shows multiple types of cloud data storage interfaces that are able to support both legacy and new applications. All of the interfaces allow storage to be provided on demand, drawn from a pool of resources. The storage capacity is drawn from a pool of storage capacity provided by storage services.

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

5.7 Cloud Data Management Interface

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

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

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

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

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

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

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

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

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

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

- determine how user credentials are mapped to principals used in an Access Control List (ACL),
- allow granting of special cloud-related privileges, and
- allow delegation to external user authorization systems (e.g., LDAP or Active Directory).

Domains may also be hierarchical, allowing for corporate domains with multiple children domains for departments or individuals. The domain concept is also used to aggregate usage data that is used to bill, meter, and monitor cloud use.

Finally, capabilities allow a client to discover the capabilities of a CDMI implementation. Requirements throughout this International Standard shall be understood in the context of CDMI capabilities. Mandatory requirements on functionality that is conditioned on a CDMI capability shall not be interpreted to require implementation of that capability, but rather shall be interpreted to apply only to implementations that support the functionality required by that capability.

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

5.8 Object model for CDMI

The model for CDMI is shown in Figure 4.

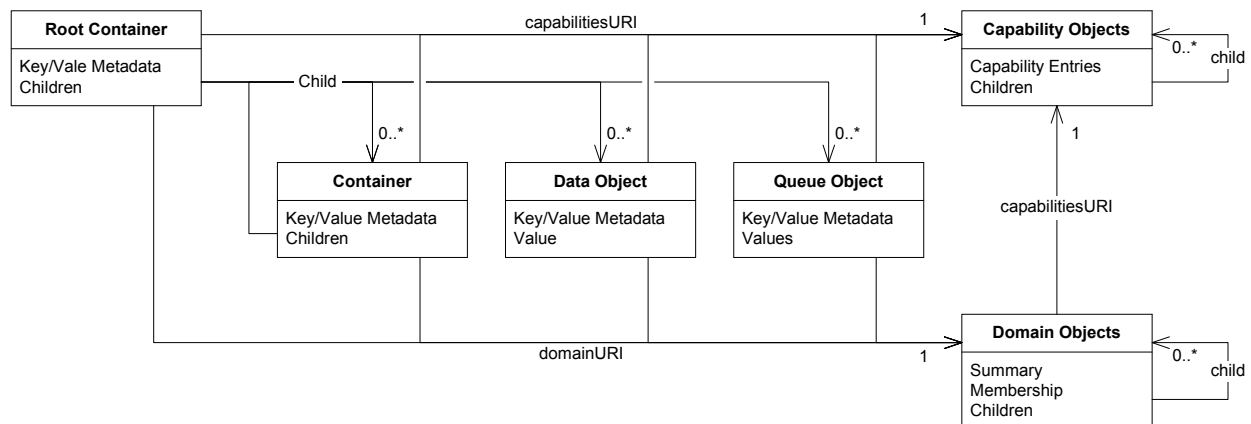


Figure 4 — CDMI object model

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

Table 3 — Types of resources in the model

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

For data storage operations, the client of the interface only needs to know about container objects and data objects. All data path implementations are required to support at least one level of containers (see 5.5). Using the CDMI object model (see Figure 4), the client can send a PUT via CDMI (see 5.6) to the new container URI and create a new container with the specified name. Container metadata are optional and are expressed as a series of name-value pairs. After a container is created, a client can send a PUT to create a data object within the newly created container. A subsequent GET will fetch the data object, including the value field.

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

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

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

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

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

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

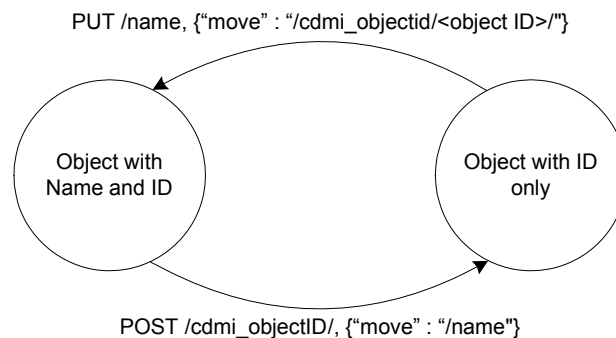


Figure 5 — Object transitions between named and ID-only

5.9 CDMI metadata

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

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

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

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

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

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

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

Table 4 — Creation/consumption of storage system metadata

	Created by user	Created by system
Consumed by user	User metadata	Storage system metadata
Consumed by system	Data system metadata	N/A

5.10 Object ID

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

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

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

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

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

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

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

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

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

5.11 CDMI object ID format

The cloud service shall create the object ID, which identifies an object. The object ID shall be globally unique and shall conform to the format defined in Figure 6. The native format of an object ID is a variable-length byte sequence and shall be a maximum length of 40 bytes. A client should treat object IDs as opaque byte strings.

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

Figure 6 — Object ID format

The fields shown in Figure 6 are defined as follows.

- The reserved bytes shall be set to zero.
- The Enterprise Number field shall be the SNMP enterprise number of the offering organization that developed the system that created the object ID, in network byte order. See RFC 2578 and <http://www.iana.org/assignments/enterprise-numbers>. 0 is a reserved value.
- The byte at offset 5 shall contain the full length of the object ID, in bytes.
- The CRC field shall contain a 2-byte (16-bit) CRC in network byte order. The CRC field enables the object ID to be validated for integrity. The CRC field shall be generated by running the

algorithm (see [ISO 14701:2012](#)) across all bytes of the object ID, as defined by the Length field, with the CRC field set to zero. The CRC function shall have the following fields:

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

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

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

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

5.12 Security

5.12.1 Security objectives

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

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

Security measures within CDMI are summarized as:

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

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

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

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

5.12.2 HTTP security

HTTP is the mandatory transport mechanism for this version of CDMI. It is important to note that HTTP, by itself, offers no confidentiality or integrity protections. As CDMI is built on top of HTTP, HTTP over Transport Layer Security (TLS) (i.e., HTTPS) is the mechanism that is used to secure the communications between CDMI clients and servers.

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

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

When TLS is used to secure HTTP, the client and server typically perform some form of entity authentication. However, the specific nature of this entity authentication depends on the cipher suite negotiated; a cipher suite specifies the encryption algorithm and digest algorithm to use on a TLS connection. A very common scenario involves using server-side certificates, which the client trusts, as the basis for unidirectional entity authentication. It is possible that mutual authentication involving both client-side and server-side certificates are required.

5.12.3 Client authentication

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

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

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

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

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

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

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

5.12.4 Use of TLS and HTTP

Recommendations for using HTTP and TLS are as follows:

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

5.12.5 Further information

For further information pertaining to storage security techniques, see [ISO 14701:2012](#).

5.13 Required HTTP support

5.13.1 RFC 2616 support requirements

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

5.13.2 Content-type negotiation

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

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

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

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

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

5.13.3 Range support

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

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

5.13.4 URI escaping

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

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

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

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

The response shall be:

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

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

5.13.5 Use of URIs

The format and syntax of URIs are defined by [RFC 3986](#).

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

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

[Table 5](#) shows how relative URIs are resolved against root URIs.

Table 5 — Relative URIs resolved against root URIs

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

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

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

5.13.6 Reserved characters

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

5.14 Time representations

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

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

5.15 Backwards compatibility

5.15.1 Value transfer encoding

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

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

5.15.2 Container export capabilities

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

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

5.16 Object references

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

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

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

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

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

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

The following shows the response.

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

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

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

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

The following shows the response.

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

EXAMPLE 3 PUT to create a reference:

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