BS ISO/IEC 17826:2022

BSI Standards Publication

Information technology — Cloud Data Management Interface (CDMITM) Version 2.0

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 17826:2022

National foreword

This British Standard is the UK implementation of ISO/IEC 17826:2022.

The UK participation in its preparation was entrusted to Technical Committee ICT/1, Information systems co-ordination.

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

Contractual and legal considerations

This publication has been prepared in good faith, however no representation, warranty, assurance or undertaking (express or implied) is or will be made, and no responsibility or liability is or will be accepted by BSI in relation to the adequacy, accuracy, completeness or reasonableness of this publication. All and any such responsibility and liability is expressly disclaimed to the full extent permitted by the law.

This publication is provided as is, and is to be used at the recipient's own risk.

The recipient is advised to consider seeking professional guidance with respect to its use of this publication.

This publication is not intended to constitute a contract. Users are responsible for its correct application.

© The British Standards Institution 2022
Published by BSI Standards Limited 2022

ISBN 978 0 539 18669 7

ICS 35.210

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 2022.

Amendments/corrigenda issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 17826:2022

INTERNATIONAL STANDARD

ISO/IEC 17826

Third edition
2022-05

Information technology — Cloud Data Management Interface (CDMI) Version 2.0.0

Technologies de l'information — Interface de management des données du nuage informatique (CDMI) Version 2.0.0

Reference number
ISO/IEC 17826:2022(E)

Documents
Document Sharing Hub
© ISO/IEC 2022

Full document details: BS ISO/IEC 17826:2022

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2022

All rights reserved. Unless otherwise specified, or required in the context of its implementation, 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
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 17826:2022

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.

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 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

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) or the IEC list of patent declarations received (see https://patents.iec.ch).

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

For an explanation of the voluntary nature of standards, 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 www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by SNIA (as Cloud Data Management Interface (CDMI™), Version 2.0.0) and drafted in accordance with its editorial rules. It was adopted, under the JTC 1 PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, Information technology.

This third edition cancels and replaces the second edition (ISO/IEC 17826:2016), which has been technically revised.

The main changes are as follows:

— Specification changes:

  — no longer a reliance on X-CDMI-SPECIFICATION-VERSION header, as per IETF guidelines;

  — support for PATCH HTTP verb, in addition of PUT, for resource updates;

  — use of /cdmi/v2.0.0/ root URI in examples;

  — new format for query parameters in URIs;

  — exports section has been re-written;

  — versioning examples have been updated;

  — audit and logging clauses have been removed (due to no vendor implementations);

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

  - Bibliography section has been re-written;
  - RFC references have been updated;
  - HTTPS is used instead of HTTP in examples;
  - references updated from CIFS to SMB;
  - document formatting cleanup.
- Incorporated Extensions into Specification:
  - Encrypted Object extension;
  - Delegated Access Control extension;
  - JSON Transfer Encoding extension.
- Incorporated Extensions into Annex A:
  - summary metadata for bandwidth;
  - expiring access control entries;
  - group storage system metadata;
  - header-based metadata;
  - immediate query.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

Table of Contents:

I CDMI Preamble 1

Clause 1: Scope 3

Clause 2: Normative references 4

Clause 3: Terms, acronyms, and definitions 6

Clause 4: Conventions 11
4.1 Interface format . . . 11
4.2 Typographical conventions . . . 12
4.3 Request and response body requirements . . . 13
4.4 Key Word requirements . . . 14

Clause 5: Overview of Cloud Storage 15
5.1 Overview . . . 15
5.2 Reference model for cloud storage interfaces . . . 18
5.3 Cloud data management interface . . . 19
5.4 Security . . . 23
5.5 Required HTTP support . . . 25
5.6 Time representations . . . 28
5.7 Backwards compatibility . . . 29
5.8 Object references . . . 30

II Basic Cloud Storage 32

Clause 6: Data Object Resource Operations using HTTP 33
6.1 Overview . . . 33
6.2 Create a data object using HTTP . . . 34
6.3 Read a data object using HTTP . . . 36
6.4 Update a data object using HTTP . . . 39
6.5 Delete a data object using HTTP . . . 41

Clause 7: Container Object Resource Operations using HTTP 43
7.1 Overview . . . 43
7.2 Create a container object using HTTP . . . 44
7.3 Read a container object using HTTP . . . 46
7.4 Update a container object using HTTP . . . 47
7.5 Delete a container object using HTTP . . . 48
7.6 Create (POST) a new data object using HTTP . . . 50

III CDMI Core 52

Clause 8: Data Object Resource Operations using CDMI 53
8.1 Overview . . . 53
8.2 Data object details . . . 54
8.3 Create a data object using CDMI . . . 56
8.4 Read a data object using CDMI . . . 67
8.5 Update a data object using CDMI . . . 76
8.6 Delete a data object using CDMI . . . 85

Clause 9: Container Object Resource Operations using CDMI 87
9.1 Overview . . . 87
9.2 Container object details . . . 88
9.3 Create a container object using CDMI . . . 90
9.4 Read a container object using CDMI . . . 97
9.5 Update a container object using CDMI . . . 102

Documents
Document Sharing Hub

© ISO/IEC 2022 – All rights reserved

Full document details: BS ISO/IEC 17826:2022

9.6 Delete a container object using CDMI . . . 107
9.7 Create (POST) a new data object using CDMI . . . 109
9.8 Create (POST) a new queue object using CDMI . . . 119

IV CDMI Advanced **125**

Clause 10: Domain object resource operations using CDMI **126**
10.1 Overview . . . 126
10.2 Domain object details . . . 128
10.3 Domain object summaries . . . 131
10.4 Domain object membership . . . 134
10.5 Create a domain object using CDMI . . . 137
10.6 Read a domain object using CDMI . . . 141
10.7 Update a domain object using CDMI . . . 145
10.8 Delete a domain object using CDMI . . . 149

Clause 11: Queue object resource operations using CDMI **151**
11.1 Overview . . . 151
11.2 Queue object details . . . 152
11.3 Create a queue object using CDMI . . . 155
11.4 Read a queue object using CDMI . . . 161
11.5 Update a queue object using CDMI . . . 168
11.6 Delete a queue object using CDMI . . . 172
11.7 Enqueue a new queue object value using CDMI . . . 174
11.8 Delete a queue object value using CDMI . . . 180

Clause 12: Capability object resource operations using CDMI **182**
12.1 Overview . . . 182
12.2 Capability object details . . . 183
12.3 Read a capabilities object using CDMI . . . 199

Clause 13: Exported protocols **204**
13.1 Overview . . . 204
13.2 Container object export details . . . 205
13.3 NFS exported protocol . . . 208
13.4 SMB exported protocol . . . 210
13.5 iSCSI exported protocol . . . 211
13.6 WebDAV exported protocol . . . 212
13.7 OCCI exported protocol . . . 213

Clause 14: CDMI snapshots **215**
14.1 Overview . . . 215
14.2 Creating a snapshot . . . 216
14.3 Deleting a snapshot . . . 217

Clause 15: Serialization/deserialization **218**
15.1 Overview . . . 218
15.2 Canonical format . . . 219
15.3 Exporting serialized data . . . 221
15.4 Importing serialized data . . . 222

Clause 16: Metadata **223**
16.1 Overview . . . 223
16.2 Support for storage system metadata . . . 224
16.3 Support for data system metadata . . . 226
16.4 Support for provided data system metadata . . . 234
16.5 Support for user metadata . . . 236
16.6 Metadata update operations . . . 237

Clause 17: Access control **238**
17.1 Overview . . . 238
17.2 Access control flow . . . 239

Clause 18: Retention and hold management **251**
18.1 Overview . . . 251
18.2 Retention management disciplines . . . 252
18.3 CDMI retention . . . 253
18.4 CDMI hold . . . 255
18.5 CDMI auto-deletion . . . 258

Documents
Document Sharing Hub

© ISO/IEC 2022 – All rights reserved

Full document details: BS ISO/IEC 17826:2022

18.6 Retention security considerations . . . 259

Clause 19: Scope specification **260**
19.1 Overview . . . 260
19.2 Examples . . . 261
19.3 Query matching expressions . . . 263

Clause 20: Results specification **266**
20.1 Overview . . . 266
20.2 Examples . . . 267

Clause 21: Notification queues **268**
21.1 Overview . . . 268
21.2 Metadata . . . 269

Clause 22: Query queues **273**
22.1 Overview . . . 273
22.2 Extending CDMI query . . . 275

Clause 23: Encrypted objects **276**
23.1 Overview . . . 276
23.2 Encryption operations . . . 277
23.3 Example uses of encrypted objects . . . 280
23.4 KMS integration . . . 281
23.5 CMS format . . . 282
23.6 JOSE format . . . 283
23.7 Signature/digest verification . . . 284
23.8 Error handling . . . 285

Clause 24: Delegated access control **286**
24.1 Overview . . . 286
24.2 Delegated access control (DAC) . . . 288
24.3 Delegated access control message exchange . . . 290
24.4 Client header passthrough . . . 292
24.5 DAC request . . . 293
24.6 Packaged DAC request . . . 295
24.7 DAC response . . . 296
24.8 Packaged DAC response . . . 297
24.9 Error handling . . . 299
24.10 Examples . . . 300

Clause 25: Data object versions **311**
25.1 Overview . . . 311
25.2 Traversing version-enabled data objects . . . 313
25.3 Concurrent updates and version-enabled data objects . . . 314
25.4 Capabilities for version-enabled data objects . . . 316
25.5 Updates triggering version creation . . . 317
25.6 Operations on version-enabled data objects . . . 318
25.7 Operations on data object versions . . . 319
25.8 Query of data object versions . . . 320
25.9 Version-enabled data object serialization . . . 321

V CDMI Annexes **323**

Clause 26: Extensions **324**
26.1 Overview . . . 324
26.2 Summary metadata for bandwidth . . . 325
26.3 Expiring access control entries (ACEs) . . . 327
26.4 Group storage system metadata . . . 328
26.5 Header-based metadata . . . 329
26.6 Immediate query . . . 337

VI References **340**

Bibliography **341**

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

List of Figures

Fig. 1: Existing data storage interface standards . . . 16
Fig. 2: Storage interfaces for object storage client data . . . 16
Fig. 3: Cloud storage reference model . . . 18
Fig. 4: CDMI object model . . . 20
Fig. 5: Object transitions between named and ID-only . . . 21
Fig. 6: CDMI URI Components . . . 26

Fig. 7: Hierarchy of domains . . . 126

Fig. 8: Hierarchy of capabilities . . . 183

Fig. 9: CDMI and OCCI in an integrated cloud computing environment . . . 213

Fig. 10: Snapshot container structure . . . 215

Fig. 11: Access control flow . . . 239

Fig. 12: Object retention . . . 253
Fig. 13: Object hold . . . 255
Fig. 14: Object hold on object with retention . . . 255
Fig. 15: Object with multiple holds . . . 256

Fig. 16: Encrypted object state transistions . . . 277

Fig. 17: Non-delegated (ACL-based) access control data flow . . . 286
Fig. 18: Delegated access control data flow example for non-encrypted object . . . 290
Fig. 19: Delegated access control data flow example for encrypted object . . . 291

Fig. 20: Updates to a non-version-enabled data object . . . 311
Fig. 21: Updates to a version-enabled data object . . . 312
Fig. 22: Linkages between a version-enabled data object and data object versions . . . 313
Fig. 23: Overlapping concurrent updates . . . 314
Fig. 24: Linkages for overlapping updates . . . 314
Fig. 25: Nested concurrent updates . . . 315
Fig. 26: Linkages for nested updates . . . 315
Fig. 27: Version to `capabilityURI` relationships . . . 316

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

List of Tables

Table 1: Overview of this document . . . 2

Table 2: Interface format . . . 11
Table 3: Key word requirements . . . 14

Table 4: Types of resources in the CDMI object model . . . 20
Table 5: Creation/consumption of storage system metadata . . . 21
Table 6: Object ID format . . . 22
Table 7: Relative URIs resolved against root URIs . . . 27

Table 8: Capabilities - Create a CDMI data object using HTTP . . . 34
Table 9: Request headers - Create a CDMI data object using HTTP . . . 34
Table 10: HTTP status codes - Create a data object using HTTP . . . 35
Table 11: Capabilities - Read a CDMI data object using HTTP . . . 36
Table 12: Request header - Read a CDMI data object using HTTP . . . 36
Table 13: Response headers - Read a CDMI Data Object using HTTP . . . 37
Table 14: HTTP status codes - Read a CDMI data object using HTTP . . . 37
Table 15: Capabilities - Update a CDMI data object using HTTP . . . 39
Table 16: Request headers - Update a CDMI data object using HTTP . . . 39
Table 17: Response header - Update a CDMI data object using HTTP . . . 40
Table 18: HTTP status codes - Update a CDMI data object using HTTP . . . 40
Table 19: Capabilities - Delete a CDMI data object using HTTP . . . 41
Table 20: HTTP status codes - Delete a CDMI data object using HTTP . . . 41

Table 21: Capabilities - Create a CDMI container object using HTTP . . . 44
Table 22: HTTP status codes - Create a container object using HTTP . . . 44
Table 23: Capabilities - Delete a CDMI container object using HTTP . . . 48
Table 24: HTTP status codes - Delete a CDMI container object using HTTP . . . 49
Table 25: Capabilities - Create a CDMI data object using HTTP POST . . . 50
Table 26: Request header - Create a new data object using HTTP . . . 50
Table 27: Response header - Create a new data object using HTTP . . . 51
Table 28: HTTP status codes - Create a new data object using HTTP . . . 51

Table 29: Capabilities - Create a CDMI data object using CDMI . . . 57
Table 30: Request headers - Create a CDMI data object using CDMI . . . 57
Table 31: Request message body - Create a data object using CDMI . . . 58
Table 32: Response headers - Create a data object using CDMI . . . 61
Table 33: Response message body - Create a data object using CDMI . . . 61
Table 34: HTTP status codes - Create a data object using CDMI . . . 62
Table 35: Capabilities - Read a CDMI data object using CDMI . . . 67
Table 36: Request headers - Read a CDMI data object using CDMI . . . 67
Table 37: Response headers - Read a CDMI data object using CDMI . . . 68
Table 38: Response message body - Read a CDMI data object using CDMI . . . 68
Table 39: HTTP status codes - Read a CDMI data object using CDMI . . . 70
Table 40: Capabilities - Update a CDMI data object using CDMI . . . 76
Table 41: Request headers - Update a CDMI data object using CDMI . . . 77
Table 42: Request message body - Update a CDMI data object using CDMI . . . 77
Table 43: Response header - Update a CDMI data object using CDMI . . . 80
Table 44: HTTP status codes - Update a CDMI data object using CDMI . . . 80
Table 45: Capabilities - Delete a CDMI data object using CDMI . . . 85
Table 46: HTTP status codes - Delete a CDMI data object using CDMI . . . 85

Table 47: Container metadata . . . 89
Table 48: Capabilities - Create a CDMI container object using CDMI . . . 91
Table 49: Request headers - Create a container object using CDMI . . . 91
Table 50: Request message body - Create a container object using CDMI . . . 91
Table 51: Response headers - Create a container object using CDMI . . . 93
Table 52: Response message body - Create a container object using CDMI . . . 93

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

Table 53: HTTP status codes - Create a CDMI container object using CDMI . . . 94
Table 54: Capabilities - Read a CDMI Container Object using CDMI . . . 97
Table 55: Request headers - Read a container object using CDMI . . . 97
Table 56: Response headers - Read a container object using CDMI . . . 98
Table 57: Response message body - Read a container object using CDMI . . . 98
Table 58: HTTP status codes - Read a container object using CDMI . . . 99
Table 59: Capabilities - Update a CDMI container object using CDMI . . . 103
Table 60: Request headers - Update a container object using CDMI . . . 103
Table 61: Request message body - Update a container object using CDMI . . . 103
Table 62: Response header - Update a container object using CDMI . . . 105
Table 63: HTTP status codes - Update a container object using CDMI . . . 105
Table 64: Capabilities - Delete a CDMI container object using CDMI . . . 107
Table 65: HTTP status codes - Delete a container object using CDMI . . . 108
Table 66: Capabilities - Create a CDMI data object using CDMI . . . 110
Table 67: Request headers - Create a new data object Using CDMI . . . 111
Table 68: Request message body - Create a new data object Using CDMI . . . 111
Table 69: Response headers - Create a new data object using CDMI . . . 115
Table 70: Response message body - Create a new data object using CDMI . . . 115
Table 71: HTTP status codes - Create a new data object using CDMI . . . 116
Table 72: Capabilities - Create a CDMI Queue object using CDMI . . . 120
Table 73: Request headers - Create a new queue object using CDMI . . . 120
Table 74: Request message body - Create a new queue object using CDMI . . . 121
Table 75: Response headers - Create a new queue object using CDMI . . . 122
Table 76: Response message body - Create a new queue object using CDMI . . . 122
Table 77: HTTP status codes - Create a new queue object using CDMI . . . 123

Table 78: Required metadata for a domain object . . . 129
Table 79: Contents of domain summary objects . . . 132
Table 80: Required settings for domain member user objects . . . 134
Table 81: Required settings for domain member delegation objects . . . 135
Table 82: Capabilities - Create a CDMI domain object using CDMI . . . 137
Table 83: Request headers - Create a domain object using CDMI . . . 137
Table 84: Request message body - Create a domain object using CDMI . . . 138
Table 85: Response headers - Create a domain object using CDMI . . . 139
Table 86: Response message body - Create a domain object using CDMI . . . 139
Table 87: HTTP status codes - Create a domain object using CDMI . . . 140
Table 88: Capabilities - Read a CDMI domain object using CDMI . . . 141
Table 89: Request headers - Read a domain object using CDMI . . . 141
Table 90: Response headers - Read a domain object using CDMI . . . 142
Table 91: Response message body - Read a domain object using CDMI . . . 142
Table 92: HTTP status codes - Read a domain object using CDMI . . . 143
Table 93: Capabilities - Update a CDMI domain object using CDMI . . . 145
Table 94: Request headers - Update a domain object using CDMI . . . 145
Table 95: Request message body - Update a domain object using CDMI . . . 146
Table 96: Response header - Update a domain object using CDMI . . . 147
Table 97: HTTP status codes - Update a domain object using CDMI . . . 147
Table 98: Capabilities - Delete a CDMI domain object using CDMI . . . 149
Table 99: HTTP status codes - Delete a domain object using CDMI . . . 150

Table 100: Capabilities - Create a CDMI queue object using CDMI . . . 155
Table 101: Request headers - Create a queue object Using CDMI . . . 156
Table 102: Request message body - Create a queue object using CDMI . . . 156
Table 103: Response headers - Create a queue object Using CDMI . . . 158
Table 104: Response message body - Create a queue object using CDMI . . . 158
Table 105: HTTP status codes - Create a queue object using CDMI . . . 159
Table 106: Capabilities - Read a CDMI queue object using CDMI . . . 161
Table 107: Request headers - Read a queue object using CDMI . . . 162
Table 108: Response headers - Read a queue object using CDMI . . . 162
Table 109: Response message body - Read a queue object using CDMI . . . 162
Table 110: HTTP status codes - Read a queue object using CDMI . . . 165
Table 111: Capabilities - Update a queue object using CDMI . . . 168
Table 112: Request headers - Update a queue object Using CDMI . . . 168
Table 113: Request message body - Update a queue object Using CDMI . . . 168
Table 114: Response header - Update a queue object Using CDMI . . . 170
Table 115: HTTP status codes - Update a queue object using CDMI . . . 170
Table 116: Capabilities - Delete a queue object using CDMI . . . 172
Table 117: HTTP status codes - Delete a queue object Using CDMI . . . 173
Table 118: Capabilities - Enqueue a new queue object value using CDMI . . . 174
Table 119: Request headers - Enqueue a new queue object value using CDMI . . . 174
Table 120: Request message body - Enqueue a new queue object value using CDMI . . . 175

Documents
Document Sharing Hub

© ISO/IEC 2022 – All rights reserved

Full document details: BS ISO/IEC 17826:2022

Table 121: HTTP status codes - Enqueue a new queue object value Using CDMI . . . 177
Table 122: Capabilities - Delete a queue object value using CDMI . . . 180
Table 123: HTTP status codes - Delete a queue object value using CDMI . . . 181

Table 124: System-wide capabilities . . . 185
Table 125: Capabilities for storage system metadata . . . 189
Table 126: Capabilities for data system metadata . . . 190
Table 127: Capabilities for data objects . . . 193
Table 128: Capabilities for container objects . . . 194
Table 129: Capabilities for domain objects . . . 196
Table 130: Capabilities for queue objects . . . 198
Table 131: Capabilities - Read a capabilities object using CDMI . . . 199
Table 132: Request headers - Read a capabilities object using CDMI . . . 199
Table 133: Response headers - Read a capabilities object Using CDMI . . . 200
Table 134: Response message body - Read a capabilities object using CDMI . . . 200
Table 135: HTTP status codes - Read a capabilities object using CDMI . . . 201

Table 136: Elements of the NFS protocol export structure . . . 208
Table 137: Elements of the SMB protocol export structure . . . 210
Table 138: Elements of the iSCSI protocol export structure . . . 211
Table 139: Elements of the WebDAV protocol export structure . . . 212

Table 140: Serialization import behaviour . . . 222

Table 141: Storage system metadata . . . 224
Table 142: Data system metadata . . . 226
Table 143: Provided values of data system metadata . . . 234

Table 144: ACE types . . . 241
Table 145: Who identifiers . . . 241
Table 146: ACE flags . . . 241
Table 147: ACE masks bits . . . 243
Table 148: ACE bit mask/string . . . 249

Table 149: Query matching expressions . . . 263

Table 150: Required metadata for a notification queue . . . 269
Table 151: Notification status metadata . . . 272

Table 152: Required metadata for a query queue . . . 273
Table 153: Query status metadata . . . 274

Table 154: Access modes for DAC . . . 288
Table 155: DAC request . . . 293
Table 156: Packaged DAC request . . . 295
Table 157: DAC response . . . 296
Table 158: Packaged DAC response . . . 297

Table 159: Version-enabled data object metadata items . . . 313

Table 161: Response headers - Inspect a data object using HTTP . . . 330
Table 162: HTTP status codes - Inspect a data object using HTTP . . . 331
Table 163: Request headers - Create a container object using HTTP . . . 333
Table 164: Response Headers - Inspect a container object using HTTP . . . 334
Table 165: HTTP status codes - Inspect a container object using HTTP . . . 334
Table 167: Required metadata for a query queue . . . 338

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 17826:2022

Part I

CDMI Preamble

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

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

This document is organized as follows:

Table 1: Overview of this document

Clause 1	Scope	Defines the scope of this document
Clause 2	Normative references	Lists the normative references for this document
Clause 3	Terms	Provides terminology used in this document
Clause 4	Conventions	Describes the conventions used in presenting the interfaces and the typographical conventions used in this document
Clause 5	Overview of Cloud Storage	Provides a brief overview of cloud storage and details the philosophy behind this International Standard as a model for the operations
Clause 6	Data Object Resource Operations using HTTP	Provides the normative standard of data object resource operations using HTTP
Clause 7	Container Object Resource Operations using HTTP	Provides the normative standard of container object resource operations using HTTP
Clause 8	Data Object Resource Operations using CDMI	Provides the normative standard of data object resource operations using CDMI
Clause 9	Container Object Resource Operations using CDMI	Provides the normative standard of container object resource operations using CDMI
Clause 10	Domain Object Resource Operations using CDMI	Provides the normative standard of domain object resource operations using CDMI
Clause 11	Queue Object Resource Operations using CDMI	Provides the normative standard of queue object resource operations using CDMI
Clause 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 can 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 21	Notification Queues	Describes how CDMI clients can efficiently discover what changes have occurred to the system
Clause 22	Query Queues	Describes how CDMI clients can 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
Clause 24	Delegated Access Control	Describes how to delegate access control to external systems
Clause 25	Data Object Versions	Describes how to work with versioned data objects
Clause 26	Extensions	Provides informative vendor extensions. Each extension is added to the standard when at least two vendors implement the extension.

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

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.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

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.

ISO 3166:2013, *Codes for the representation of names of countries and their subdivisions (Parts 1, 2 and 3)* - see [35][36][37]

ISO 4217:2015, *Codes for the representation of currencies and funds* - see [38]

ISO 8601:2019, *Data elements and interchange formats – Information interchange – Representation of dates and times* - see [32][33]

ISO/IEC 9594-8:2017, *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks* - see [43]

ISO 14721:2012, *Space data and information transfer systems – Open archival information system (OAIS) – Reference model* - see [34]

ISO/IEC 14776-414:2009, *SCSI Architecture Model - 4 (SAM-4)* - see [29]

ISO/IEC 17788:2014, *Information technology – Cloud computing – Overview and vocabulary* - see [31]

ISO/IEC 20648, *TLS specification for storage systems* - see [39]

ISO/IEC 27040:2015, *Information technology – Security techniques – Storage security* - see [30]

ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards, 6th edition, 2011* - see [28]

IEEE 1003.1-2017, *IEEE Standard for Information Technology–Portable Operating System Interface (POSIX(R)) Base Specifications, Issue 7* - see [41]

RFC 1867, *Form-based File Upload in HTML* - see [20]

RFC 2045, *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies* - see [9]

RFC 2046, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types* - see [10]

RFC 2119, *Key Words for Use in RFCs to Indicate Requirement Levels* - see [3]

RFC 2578, *Structure of Management Information Version 2 (SMIv2)* - see [21]

RFC 2616, *Hypertext Transfer Protocol – HTTP/1.1* - see [23]

RFC 2617, *HTTP Authentication: Basic and Digest Access Authentication* - see [8]

RFC 3280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile* - see [13]

RFC 3530, *Network File System (NFS) Version 4 Protocol* - see [1]

RFC 7143, *Internet Small Computer System Interface (iSCSI) Protocol (Consolidated)* - see [4]

RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax* - see [2]

RFC 4627, *The Application/JSON Media Type for JavaScript Object Notation (JSON)* - see [5]

RFC 4648, *The Base16, Base32, and Base64 Data Encodings* - see [19]

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

RFC 4918, *HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)* - see [6]

RFC 5246, *The Transport Layer Security (TLS) Protocol Version 1.2* - see [25]

RFC 6068, *The 'mailto' URI Scheme* - see [27]

RFC 5652, *Cryptographic Message Syntac (CMS)* - see [12]

RFC 6208, *Cloud Data Management Interface (CDMI) Media Types* - see [26]

RFC 6839, *Additional Media Type Structured Syntax Suffixes* - see [11]

RFC 7515, *JSON Web Signatures* - see [17]

RFC 7516, *JSON Web Encryption* - see [18]

RFC 7518, *JSON Web Algorithms* - see [15]

RFC 8446, *The Transport Layer Security (TLS) Protocol Version 1.3* - see [24]

SNIA TLS, *TLS Specification for Storage Systems, version 1.1.0* - see [42]

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 17826:2022

Clause 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)
a 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 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`.

3.5

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

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

3.6

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

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

3.7

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`.

3.8

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

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

3.9

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`.

3.10

CIFS
Common Internet File System (See SMB)

3.11

cloud storage
See Data storage as a Service

3.12

CRC
cyclic redundancy check

3.13

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

3.14

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

3.15

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

3.16

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

3.17

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

3.18

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

3.19

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

3.20

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.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

3.21

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

3.22

FC
Fibre Channel

3.23

FCoE
Fibre Channel over Ethernet

3.24

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

3.25

HTTP
HyperText Transfer Protocol

3.26

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

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

3.27

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

3.28

iSCSI
Internet Small Computer Systems Interface (see RFC 7143 [4])

3.29

JOSE
JavaScript Object Signing and Encryption

3.30

JWA
JSON Web Algorithm

3.31

JWE
JSON Web Encryption

3.32

JWS
JSON Web Signing

3.33

JSON
JavaScript Object Notation

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

3.34

LDAP
Lightweight Directory Access Protocol

3.35

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

3.36

metadata
data about other data (see [34])

3.37

MIME
Multipurpose Internet Mail Extensions (see RFC 2045 [9])

3.38

NFS
Network File System (see RFC 3530 [1])

3.39

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.

3.40

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

3.41

OCCI
Open Cloud Computing Interface (see [40])

3.42

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

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

3.43

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

3.44

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

Private clouds are created due to issues of trust.

3.45

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

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

3.46

Representational State Transfer (REST)
a specific set of principles for defining, addressing, and interacting with resources addressable by URIs (see [7])

3.47

RPO
recovery point objective

3.48

RTO
recovery time objective

3.49

service level
performance targets for a service

3.50

Server Message Block
A network file system access protocol designed primarily used by Windows clients to communicate file access requests to Windows servers. (Also see CIFS)

3.51

SNMP
Simple Network Management Protocol

3.52

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.

3.53

Uniform Resource Identifier (URI)
compact sequence of characters that identifies an abstract or physical resource (see RFC 3986 [2])

3.54

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

3.55

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

3.56

WebDAV
Web Distributed Authoring and Versioning (see RFC 4918 [6])

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

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, PATCH, 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

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

4.2 Typographical conventions

All code text and HTTP status codes are shown in a fixed-width font.

API resquests also include a prefix to indicate the source of the request or reply:

- Client-initiated requests are prefixed with '–>'
- Server-initiated replies are prefixed with '<–'

An example is included below:

```
--> PUT /cdmi/2.0.0/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"
--> }

<-- HTTP/1.1 202 Accepted
<-- Location: https://cloud.example.com/cdmi/2.0.0/MyContainer/MyDataObject.txt
```

Similarly, HTTP status codes are shown in a fixed-width font, as in:

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

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

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 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.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

4.4 Key Word requirements

In this International Standard, the key words in Table 3 shall be interpreted as described in ISO/IEC Directives, Part 2. Table 2 — Key word requirements

Table 3: Key word requirements

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

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

Clause 5

Overview of Cloud Storage

5.1 Overview

5.1.1 General Context

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 can 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.

5.1.2 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).

5.1.3 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 SMB/NFS or WebDAV for file network storage, as shown in Fig. 1.

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.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

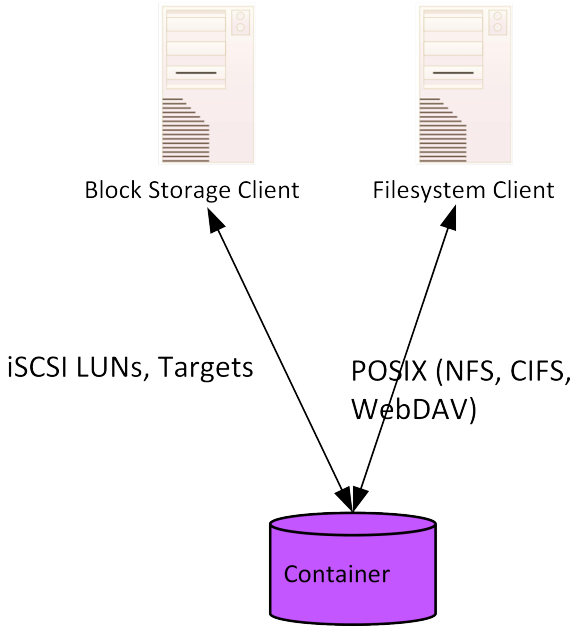

Fig. 1: Existing data storage interface standards

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 Fig. 2).

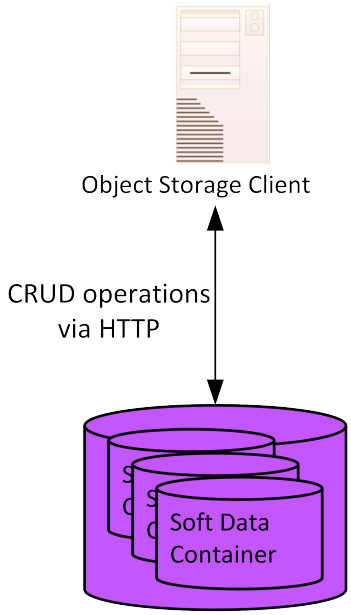

Fig. 2: Storage interfaces for object storage client data

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

5.1.4 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 16.5.

Storage system metadata is produced/interpreted by the cloud service provider and basic storage functions (e.g., modification and access statistics, access control). For more information on support for storage system metadata, see 16.2.

Data system metadata is interpreted by the cloud service provider 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, if the cloud service provider supports this level of granularity. For more information on support for data system metadata, see 16.3.

5.1.5 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 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 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., SMB, NFSv4), these extended attributes may be used to specify the data system metadata to override that specified for the container.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

5.2 Reference model for cloud storage interfaces

The cloud storage reference model is shown in Fig. 3.

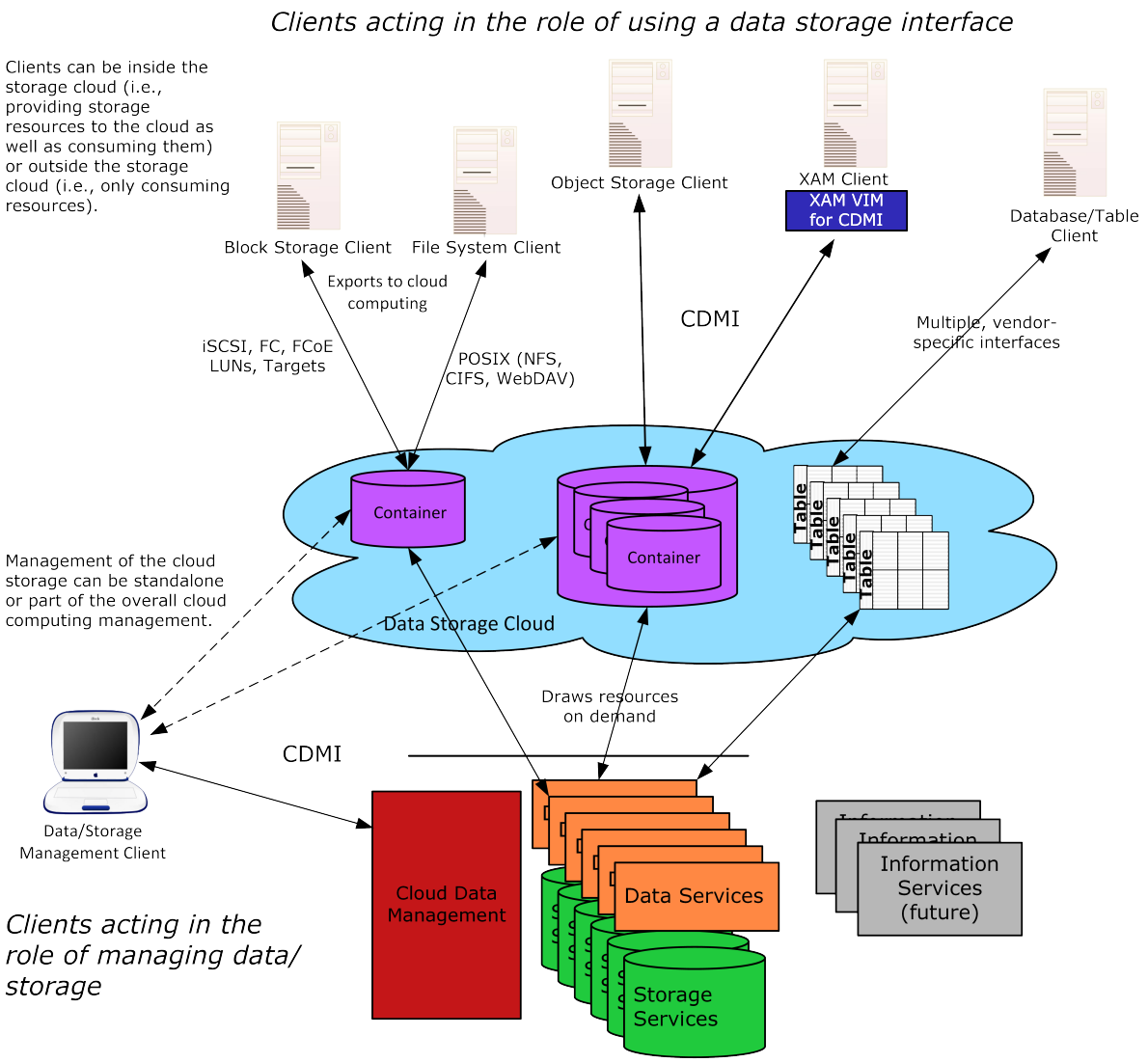

Fig. 3: Cloud storage reference model

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).

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

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 service 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 service provider provides.

Conformant cloud service providers 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 [7]).

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 5.3.3, 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.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

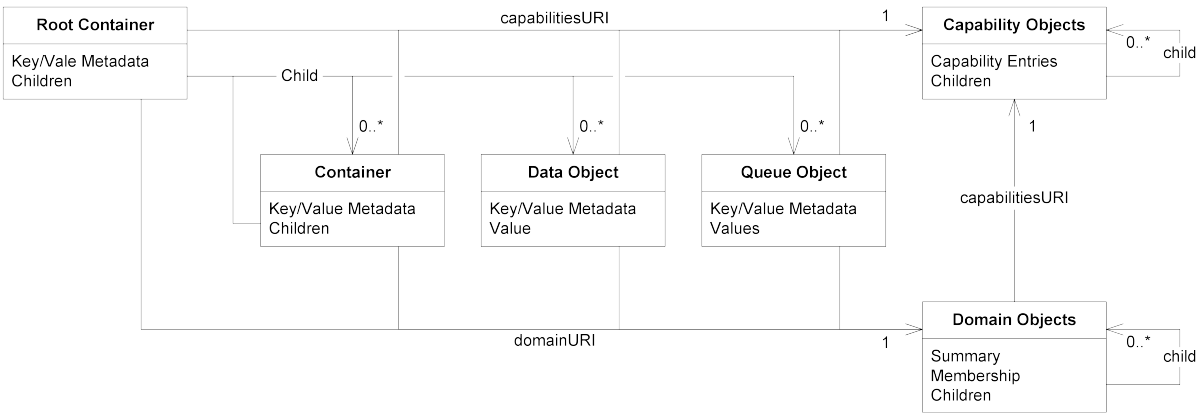

Fig. 4: CDMI object model

Table 4: Types of resources in the CDMI object model

Resource type	Description	Reference
Data objects	Data objects are used to store data and associated metadata, and provide functionality similar to files in a file system.	See clause 8.
Container objects	Container objects have zero or more children objects, and store metadata associated with the container as a whole. Container objects do not store data directly. 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 pieces of data, and store metadata associated with the queue as a whole. Enqueued values 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 5.1.5). Using the CDMI object model (see Fig. 4), the client can 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 can send a PUT to create a data object within the newly created container.

Queue objects are also defined (see Fig. 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 9.7). Subsequent to creation, objects are modified by performing PUTs using the object ID assigned by the CDMI server, using the "`/cdmi_objectid/`" URI (see 8.5). 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 8.3). Subsequent to creation, objects are modified by performing PUTs using the object path specified by the client (see 8.5). 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.

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

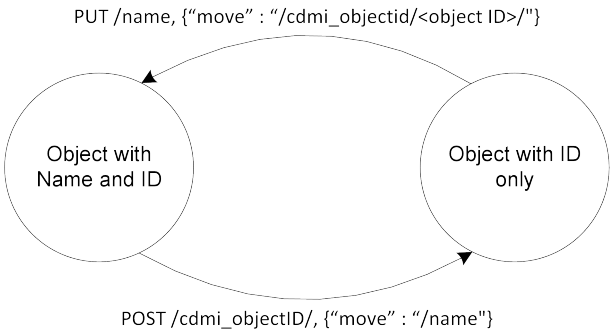

Fig. 5: Object transitions between named and ID-only

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 CDMI Servers 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 URI (see 5.5.5). If the data object "`MyDataObject.txt`", stored in the root container "`/`" with a root path of "`/cdmi/2.0.0/`", has an object ID of "`00006FFD001001CCE3B2B4F602032653`", the following pair of URIs access the same data object:

- `https://cloud.example.com/cdmi/2.0.0/MyDataObject.txt`
- `https://cloud.example.com/cdmi/2.0.0/cdmi_objectid/00006FFD001001CCE3B2B4F602032653`

If containers are supported, they shall also be accessible by object ID. If the container "MyContainer", stored in the root container "`/`" with a root path of "`/cdmi/2.0.0/`", has an object ID of "`00006FFD0010AA33D8CEF9711E0835CA`", the following pairs of URIs access the same object:

- `https://cloud.example.com/cdmi/2.0.0/MyContainer/`
- `https://cloud.example.com/cdmi/2.0.0/cdmi_objectid/00006FFD0010AA33D8CEF9711E0835CA/`

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

- `https://cloud.example.com/cdmi/2.0.0/MyContainer/MyDataObject.txt`
- `https://cloud.example.com/cdmi/2.0.0/cdmi_objectid/00006FFD0010AA33D8CEF9711E0835CA/MyDataObject.txt`

5.3.4 CDMI object ID format

The CDMI Server 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 CDMI Servers 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 [21] and https://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 CRC algorithm 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.
- 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 [19] and shall be case insensitive.

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

5.4 Security

5.4.1 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 shall be implmented, 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 12.2.7 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).

5.4.2 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” [42]; 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;
- 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 are required.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

5.4.3 Client Authentication

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

CDMI clients shall 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 [23] and RFC 2617 [8] 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 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.

5.4.4 Use of TLS and HTTP

Recommendations for using HTTP and TLS are as follows:

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

5.4.5 Further information

For further information pertaining to storage security techniques, see the latest version of ISO 20648.

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

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 [23] (i.e., HTTP 1.1). The subclauses below list the sections of RFC 2616 [23] 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 [26]. All CDMI representations follow the rules established for `application/json` as defined in RFC 4627 [5]. The use of the CDMI media types with the `+json` suffix shall be supported as defined in RFC 6839 [11].

A client can optionally supply an HTTP `Accept` header, as per section 14.1 of RFC 2616 [23]. 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 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 [23]. 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 9.4, 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 [23]).

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 [23].

5.5.4 URI escaping

Percent escaping of reserved characters specified in RFC 3986 [2] 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 and reply:

```
--> GET /cdmi/2.0.0/%40MyContainer/?objectName&metadata=%40user HTTP/1.1
--> Host: cloud.example.com
--> Accept: application/cdmi-container

<-- HTTP/1.1 200 OK
<-- Content-Type: application/cdmi-container
<--
<-- {
<--   "objectName": "@MyContainer/",
<--   "metadata": {
<--     "@user": "test",
<--     ...
<--   }
<-- }
```

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

5.5.5 Use of URIs

The format and syntax of URIs are defined by RFC 3986 [2].

This International Standard splits the RFC 3986 path into two parts: The “root path” and the “CDMI path”, as shown in Fig. 6. The URI containing only the root path is called the “root URI”.

```
 https://example.com:443/cdmi/2.0.0/myContainer/myDataObject?value
 \___/   \_____________/\__________|_________________________/ \___/
   |           |              |                |                |
scheme     authority      root path        CDMI path         query
   |                          |
   \__________________________/
              |
          root URI
```

Fig. 6: CDMI URI Components

The container at the start of the CDMI path is the root container. For example, in Fig. 6, the root container is named “`myContainer`”.

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 [2]. Every CDMI client shall maintain one or more root URIs that each correspond to a root CDMI 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.

This International Standard places no additional restrictions on root URIs beyond those specified for the “path component” in RFC 3986.

Industry conventions for RESTful APIs suggest root URIs end in “`/cdmi/<version>/`”, where <version> is in the form of “`<major>.<minor>.<micro>`”, where <major>, <minor> and <micro> are integers indicating the version of the CDMI interface specification. All examples in this specification use a root URI of `https://cloud.example.com/cdmi/2.0.0/`.

The properties of the root URI determine the `parentID` and `parentURI` fields of an root CDMI container:

- If the root path is “`/`”, the root container shall not include the `parentID` field and shall populate an empty string (“”) for the value of the `parentURI` field.
- If the root path is not “`/`” and the last entity in the root path is a CDMI container, the root container shall populate `parentID` field with the CDMI object ID of the CDMI container corresponding to the parent path entity, and shall populate the `parentURI` field with the URI of the parent path.
- If the root path is not “`/`” and the last entity in the root path is not a CDMI container, the root container shall not include the `parentID` field, and shall populate the `parentURI` field with the URI of the parent path.
- If the root path is not “`/`” and the last entry in the root path is not accessible via the scheme, root container may omit the `parentID` field and may populate `parentURI` field with an empty string (“”).

Table 7 shows how CDMI paths (relative URIs) are resolved with root URIs

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

Table 7: Relative URIs resolved against root URIs

Root URI	+ CDMI Path	=> Resolved URI
https://cloud.example.com/		https://cloud.example.com/
https://cloud.example.com/	/	https://cloud.example.com/
https://cloud.example.com/	myCDMIcontainer/testObject	https://cloud.example.com/myCDMIcontainer/testObject
https://cloud.example.com/	myCDMIcontainer/testObject	https://cloud.example.com/container/testObject
https://cloud.example.com/myNonCDMIentity/	myCDMIcontainer/testObject	https://cloud.example.com/myNonCDMIentity/myCDMIcontainer/testObject
https://cloud.example.com/myNonCDMIentity/	myCDMIcontainer/testObject	https://cloud.example.com/myCDMIcontainer/testObject
https://cloud.example.com/cdmi/2.0.0/	myCDMIcontainer/testObject	https://cloud.example.com/cdmi/2.0.0/myCDMIcontainer/testObject
https://cloud.example.com/cdmi/2.0.0/	myCDMIcontainer/testObject	https://cloud.example.com/myCDMIcontainer/testObject

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.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

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`".

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

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.

See the CDMI 1.1.1 Specification for details on backwards compatibility specific to the 1.x versions of CDMI.

5.7.1 Specification version detection

CDMI 2.x clients shall not include the `X-CDMI-Specification-Version` custom header. When a CDMI 2.x client performs an operation against a CDMI 1.x Server, the absence of this header shall result in an error response from the CDMI 1.x server. The client may use the presence of the `X-CDMI-Specification-Version` header in an error response as an indication to use CDMI 1.x (which mandates the use of this custom header), if supported.

CDMI 2.x servers may use the presence of the `X-CDMI-Specification-Version` custom header from a CDMI 1.x client as an indication to use CDMI 1.x, if supported.

5.7.2 JSON value transfer encoding

CDMI 2.x servers may support the “json” value transfer encoding. When a CDMI server supports both CDMI 2.x and CDMI 1.x, data objects with a value transfer encoding of json shall be made accessible to CDMI 1.x clients using a value transfer encoding of UTF-8, with the server adding in the required escaping.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

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 /cdmi/2.0.0/MyContainer/MyDataObject.txt HTTP/1.1
--> Host: cloud.example.com
--> Accept: application/cdmi-object

<-- HTTP/1.1 302 Found
<-- Location: https://cloud.example.com/cdmi/2.0.0/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/2.0.0/cdmi_objectid/00006FFD0010AA33D8CEF9711E0835CA HTTP/1.1
--> Host: cloud.example.com
--> Accept: application/cdmi-object

<-- HTTP/1.1 302 Found
<-- Location: https://archive.example.com/cdmi/2.0.0/cdmi_objectid/
↪00006FFD0010AA33D8CEF9711E0835CA
```

EXAMPLE 3: PUT to create a reference:

```
--> PUT /cdmi/2.0.0/MyContainer/MyDataObject.txt HTTP/1.1
--> Host: cloud.example.com Accept: application/cdmi-object
--> Content-Type: application/cdmi-object
-->
--> {
--> "reference": "https://cloud.example.com/cdmi/2.0.0/MyContainer/
↪MyOtherDataObject.txt"
--> }

<-- HTTP/1.1 201 Created
```


© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

EXAMPLE 4: POST to create a reference:

```
--> POST /cdmi/2.0.0/cdmi_objectid/ HTTP/1.1
--> Host: cloud.example.com Accept: application/cdmi-object
--> Content-Type: application/cdmi-object
-->
--> {
--> "reference": "https://cloud.example.com/cdmi/2.0.0/MyContainer/
↪MyOtherDataObject.txt"</P>
--> }

<-- HTTP/1.1 201 Created
<-- Location: https://cloud.example.com/cdmi/2.0.0/cdmi_objectid/
↪00007ED90010DF417BAD70A0C7F5CDDA
```

EXAMPLE 5: DELETE to delete a reference:

```
--> DELETE /cdmi/2.0.0/MyContainer/MyDataObject.txt HTTP/1.1
--> Host: cloud.example.com

<-- HTTP/1.1 204 No Content
```

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

Part II

Basic Cloud Storage

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

Clause 6

Data Object Resource Operations using HTTP

6.1 Overview

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

As CDMI builds on top of, and is compatible with, the HTTP standard (RFC 2616 [23]), 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 [26] 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.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

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

Capabilities that indicate which operations are supported are shown in Table 8.

Table 8: Capabilities - Create a CDMI data object using HTTP

Capability	Location	Description
`cdmi_create_dataobject`	Parent Container	Ability to create a new data object
`cdmi_create_value_range`	System Wide Capability	Ability to create a data object using a specified byte range

6.2.3 Request headers

The HTTP request headers for creating a CDMI data object using HTTP are shown in Table 9.

Table 9: 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 in this header shall be converted to lower case and stored in the `mimetype` field of the CDMI data object. • If the `Content-Type` header includes the charset parameter as defined in RFC 2616 [23] 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 [23] Section 14.35.1)	Optional

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

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 10.

Table 10: HTTP status codes - Create a data object using HTTP

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

6.2.8 Examples

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

```
--> PUT /cdmi/2.0.0/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


 <-- HTTP/1.1 201 Created
```

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

```
--> PUT /cdmi/2.0.0/MyContainer/MyEncryptedObject.txt HTTP/1.1
--> Host: cloud.example.com
--> Content-Type: application/cms
--> Content-Length: 1425
-->
--> <CMS Encrypted Object>

<-- HTTP/1.1 201 Created
```

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

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022

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

Capabilities that indicate which operations are supported are shown in Table 11.

Table 11: Capabilities - Read a CDMI data object using HTTP

Capability	Location	Description
`cdmi_read_value`	Data Object	Ability to read the value of an existing data object
`cdmi_read_value_range`	Data Object	Ability to read a sub-range of the value of an existing data object
`cdmi_object_access_by_ID`	System Wide Capability	Ability to access the object by ID

6.3.3 Request header

The HTTP request header for reading a CDMI data object using HTTP is shown in Table 12.

Table 12: Request header - Read a CDMI data object using HTTP

Header	Type	Description	Requirement
`Range`	Header string	A valid ranges-specifier (see RFC 2616 [23] Section 14.35.1)	Optional
`Accept`	Header string	"*/*" or a value as described in: 5.5.2. • 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 header 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 clause 23 – Encrypted Objects) • If the Accept header 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 header mimetype list excludes "`\*/\*`", decrypted plaintext shall only be returned if the encapsulated mimetype is included in the Accept header mimetype list.	Optional

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 17826:2022