SNIA CDMI™ (Cloud Data Management Interface)

MQTT Exports CDMI Extension

Version 0.0.1

ABSTRACT: This CDMI Extension is intended for CDMI 2.0 developers who are considering a standardized way to specify how to export a CDMI queue using the MQTT protocol. When multiple compatible implementations are demonstrated and approved by the Technical Working Group, this extension will be incorporated into the CDMI standard.

Working Draft

June 14, 2026

USAGE

DISCLAIMER

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

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

Clause 1

MQTT exports

1.1 Overview

MQTT queue exports are an optional part of the CDMI specification that allow a cloud data management client to configure a CDMI queue object so that the cloud storage system connects to an external MQTT broker on the client's behalf and publishes queue values as MQTT messages. When an MQTT export is specified, the cloud storage system establishes and maintains an outbound MQTT client connection to the designated broker and publishes CDMI queue values to the configured topic.

Support for MQTT queue exports is indicated to CDMI-enabled clients by the presence of the `"cdmi_export_mqtt"` system-wide capability and the `"cdmi_export_mqtt"` per-object capability on queue objects.

When a cloud data management client wants to export a CDMI queue via MQTT, the client creates a new CDMI queue object with an `exports` field containing an MQTT export entry, or adds an MQTT export entry to the `exports` field of an existing CDMI queue object.

When a cloud storage system is notified that an MQTT export is requested, it shall connect outbound to the MQTT broker identified by `broker_uri` and shall maintain that connection, automatically reconnecting using the specified reconnection strategy following any loss of connectivity. The CDMI server shall act as the MQTT client in all cases; the CDMI server shall not expose an MQTT broker endpoint.

As each value is enqueued into the CDMI queue object, the cloud storage system shall publish that value as a single MQTT PUBLISH message on the configured `topic` at the configured QoS level. The `dequeue_on_publish` field allows a per-export control over queue value deletion upon delivery, with queue values only removed from the CDMI queue after the delivery. Unreachability or backpressure from MQTT servers may result in a backlog of pending messages. CDMI clients may dequeue a queue value at any time, but dequeuing shall not prevent delivery of pending MQTT messages.

When an MQTT export is removed, the CDMI server shall send a clean MQTT DISCONNECT, and any pending MQTT messages shall be removed from the cloud storage system.

If an MQTT export cannot be established, the CDMI queue object containing the `exports` field shall remain accessible via CDMI, the export shall be reported as disconnected in the CDMI server populated `connected` field, and the failure reasons shall be recorded in the `last_problems` field.

If a cloud storage system supports MQTT exports created by other data access or management protocols, these exports shall be visible to CDMI-enabled clients through the presence of an `exports` entry in the CDMI queue object at the appropriate location. This allows cloud data management clients to discover all active MQTT exports.

Supporting MQTT queue exports enables the following cloud data management client value:

- Cloud data management clients can discover which MQTT export capabilities are supported.
- Cloud data management clients can discover which MQTT exports are configured on a queue object.
- Cloud data management clients can request that a CDMI queue object be exported to an external MQTT broker and topic.
- Cloud data management clients can configure authentication and TLS security for the MQTT broker connection.
- Cloud data management clients can modify or remove an MQTT queue export.

1.2 MQTT export field format

1.2.1 Top-level `exports` field format

When a CDMI server supports MQTT queue exports, each CDMI queue object shall contain an `exports` JSON object at the top level of the queue object.

```
{
    "exports": {

    }
}
```

The `exports` field contains zero or more named MQTT export entries.

1.2.2 MQTT export object

Each MQTT export entry is a user-named JSON object within the `exports` field.

```
{
    "exports": {
        "mqtt_primary": {
        },
        "mqtt_secondary": {
        }
    }
}
```

Multiple MQTT export entries may coexist within the same `exports` object, allowing a single CDMI queue to publish to multiple MQTT brokers or topics simultaneously. The CDMI server shall maintain an independent MQTT client connection for each export entry.

Each MQTT export entry contains the following fields:

Table 1.1: MQTT Export Object Fields

Field Name	Type	Description	Requirement
`type`	JSON String	The export type. Shall be set to `"MQTT"`.	Mandatory
`broker_uri`	JSON String	The URI of the MQTT broker to which the CDMI server shall connect. The URI scheme shall be `"mqtt"` for an unencrypted connection on the default port (1883), or `"mqtts"` for a TLS-secured connection on the default port (8883). A non-default port may be specified using standard URI port notation (e.g., `"mqtt://broker.example.com:1883"`).	Mandatory
`topic`	JSON String	The MQTT topic name to which the CDMI server shall publish queue values. Topics shall follow MQTT topic syntax as defined in the MQTT specification. Wildcard characters (`+` and `#`) shall not be used in publish topics.	Mandatory
`client_id`	JSON String	The MQTT client identifier the CDMI server shall present to the broker during connection. The value shall conform to the character and length constraints defined in the MQTT specification. The client ID/broker URI combination shall be unique across all connections to the broker. If a client ID/broker URI combination is already in use by another CDMI MQTT export provided by the CDMI server, the configuration shall not be considered an allowable configuration, and the CDMI server shall return an HTTP status code of `400 Bad Request` to the client.	Mandatory

continues on next page

Table 1.1 – continued from previous page

Field Name	Type	Description	Requirement
protocol_version	JSON String	The MQTT protocol version the CDMI server shall negotiate with the broker. Permitted values are "3.1.1" and "5.0". Fields marked **(MQTT 5.0 only)** in this table are ignored when this field is not "5.0". The default value shall be "3.1.1".	Optional
reconnect_strategy	JSON String	The strategy used when connectivity between the CDMI server and the MQTT broker is lost. Permitted values are "fixed" (attempt to reconnect every ten seconds), "linear" (reconnect with a linearly increasing delay starting from one second, incremented by one second up to 256 seconds), and "exponential" (reconnect with an exponential delay starting from one second, increasing by powers of 2 up to 256 seconds). The server shall add jitter equal to +/- 25% of the reconnect delay. The default value shall be "exponential".	Optional
qos	JSON String	The MQTT Quality of Service level for all PUBLISH operations. Permitted values are "0" (at most once delivery), "1" (at least once delivery), and "2" (exactly once delivery). The default value shall be "1".	Optional
retain	JSON String	When set to "true", published MQTT messages shall have the MQTT retain flag set, instructing the broker to retain the most recent message on the topic for delivery to future subscribers. Permitted values are "true" and "false". The default value shall be "false". This option is intended for last value or status topics.	Optional
clean_session	JSON String	Controls the MQTT clean session flag (MQTT 3.1.1) or clean start flag (MQTT 5.0). When "true", the CDMI server shall set this flag, so that the broker discards any previous session state for this client on connection. When "false", the CDMI server shall clear this flag, so that the broker resumes a previous session and re-delivers any unacknowledged messages. Permitted values are "true" and "false". The default value shall be "true".	Optional
keep_alive_interval	JSON String	The MQTT keep-alive interval in seconds. The CDMI server shall send an MQTT PINGREQ control packet to the broker if no other MQTT control packets have been sent within this interval. The value shall be an integer from "0" to "65535" inclusive, corresponding to the 16-bit Keep Alive field defined in the MQTT specification. A value of "0" disables the keep-alive mechanism. The default value shall be “60”.	Optional
dequeue_on_publish	JSON String	When set to "true", each queue value shall be removed from the CDMI queue only after the CDMI server has published the message to each export where dequeue_on_publish field is set to "true" (for QoS 0), or after the CDMI server has received the QoS-appropriate delivery acknowledgment from the broker for each export where dequeue_on_publish field is set to "true" (for QoS 1 and 2). When no exports have the dequeue_on_publish field set to "true", queue values are published but remain in the CDMI queue. Permitted values are "true" and "false". The default value shall be "false".	Optional
username	JSON String	The username credential the CDMI server shall include in the MQTT CONNECT packet when connecting to the broker. If absent, no username is sent.	Optional

continues on next page

Table 1.1 – continued from previous page

Field Name	Type	Description	Requirement
`password`	JSON String	The password or token credential the CDMI server shall include in the MQTT CONNECT packet when connecting to the broker. This field shall be accepted on write, but a CDMI server shall not return the password in any read response. If absent, no password is sent. This field shall not be included if the `password_secret_id` field is provided.	Optional
`password_secret_id`	JSON String	A KMIP Secret Data object identifier allowing the CDMI server to use the identity of the user to retrieve the password or token to be used in the MQTT CONNECT. Only used if the `cdmi_export_mqtt_kmip` system-wide capability is present and set to `true`.	Optional
`password_refresh_interval`	JSON String	The interval, in seconds, at which the CDMI server shall re-fetch the password or token from the KMIP object identified by `password_secret_id`. Used only when `password_secret_id` is present and the `cdmi_export_mqtt_kmip` system-wide capability is `"true"`; ignored otherwise. The value shall be a non-negative integer, and should be lower than the lifetime of the token. A value of `"0"` disables periodic re-fetching. When a non-zero re-fetch returns a password that differs from the one currently in use, the CDMI server shall apply it by performing a graceful MQTT DISCONNECT and reconnecting with the new credential, since MQTT carries credentials only in the CONNECT packet; a re-fetch returning the same value shall not cause a reconnect. If a re-fetch fails, the CDMI server shall retain the current credential and connection, record the failure in `last_problems`, and retry at the next interval. The default value shall be `"0"`.	Optional
`tls`	JSON Object	A JSON object containing TLS configuration sub-fields governing the security of the MQTT broker connection, as described in 1.2. If absent and the `broker_uri` scheme is `"mqtts"`, the CDMI server shall use its system-default trust store without mutual TLS authentication.	Optional
`will`	JSON Object	A JSON object containing Last Will and Testament (LWT) configuration sub-fields, as described in 1.3. The LWT message is published by the broker on behalf of the CDMI server if the connection is terminated unexpectedly. If absent, no LWT is configured.	Optional
`session_expiry_interval`	JSON String	**(MQTT 5.0 only)** The number of seconds the broker retains session state after the CDMI server disconnects, as sent by the CDMI server in the Session Expiry Interval property of the MQTT CONNECT packet. A value of `"0"` causes the session to expire immediately on disconnect. The default value shall be `"0"`.	Optional
`message_expiry_interval`	JSON String	**(MQTT 5.0 only)** The number of seconds after which an undelivered message expires in the broker, as sent by the CDMI server in the Message Expiry Interval property of each MQTT PUBLISH packet. If absent, no Message Expiry Interval property is set.	Optional
`content_type`	JSON String	**(MQTT 5.0 only)** A UTF-8 string describing the MIME content type of published message payloads, carried as the Content Type property in each MQTT PUBLISH packet (e.g., `"application/json"`). If absent and the enqueued CDMI queue value includes a `mimetype` field, that `mimetype` value shall be used instead. If both are absent, no Content Type property is set.	Optional

continues on next page

Table 1.1 – continued from previous page

Field Name	Type	Description	Requirement
`response_topic`	JSON String	**(MQTT 5.0 only)** An MQTT topic name included as the Response Topic property in each MQTT PUBLISH packet, indicating to message recipients the topic for response messages. If absent, no Response Topic property is set.	Optional
`user_properties`	JSON Array of JSON Arrays	**(MQTT 5.0 only)** A list of user-defined key-value pairs included as User Property entries in each MQTT PUBLISH packet. Each element of the outer array shall be a two-element JSON array of strings of the form `["key", "value"]`.	Optional
`disabled`	JSON String	Controls whether the MQTT export is administratively disabled. Permitted values are `"true"` and `"false"`. When set to `"true"`, the cloud storage server shall immediately close the connection to the broker. The export configuration is fully retained and the export shall resume operation without reconfiguration when `disabled` is subsequently set to `"false"`. The default value shall be `"false"`.	Optional
`connected`	JSON String	CDMI server populated and ignored if specified in a create or update. Indicates whether the CDMI server currently has an active MQTT connection to the broker. Permitted values are `"true"` and `"false"`.	Read-Only
`messages_published`	JSON String	CDMI server populated and ignored if specified in a create or update. The cumulative count of MQTT messages successfully published to the broker since the export entry was created.	Read-Only
`messages_pending`	JSON String	CDMI server populated and ignored if specified in a create or update. The current count of enqueued CDMI values not yet published to the broker plus the current count of MQTT messages not yet acknowledged by the broker (for QoS level 1 and 2).	Read-Only
`messages_dropped`	JSON String	CDMI server populated and ignored if specified in a create or update. The current count of CDMI values that were unable to be delivered to the MQTT broker and are not retried.	Read-Only
`last_connected`	JSON String	CDMI server populated and ignored if specified in a create or update. The timestamp of the most recent successful broker connection, expressed as an ISO 8601 UTC date-time string (e.g., `"2026-11-01T14:32:00.000000Z"`).	Read-Only
`last_problems`	JSON Array	CDMI server populated and ignored if specified in a create or update. An array of zero or more RFC 9457 problem details JSON objects describing each current connection or protocol error, if any. If no problems are present, this array shall be empty.	Read-Only
`last_problem_time`	JSON String	CDMI server populated and ignored if specified in a create or update. The timestamp of when the `last_problems` was updated, expressed as an ISO 8601 UTC date-time string (e.g., `"2026-11-01T14:32:00.000000Z"`). When `last_problems` array is empty, this time indicates when the last problem was resolved.	Read-Only

Servers shall return an HTTP status code of `400 Bad Request` when an MQTT export entry does not conform to an allowable configuration on the server.

1.2.3 TLS configuration sub-fields

The `tls` JSON object within an MQTT export entry may contain the following sub-fields, which configure the TLS connection the CDMI server uses when connecting to the MQTT broker.

Table 1.2: MQTT Export TLS Sub-fields

Field Name	Type	Description	Requirement
`verify_broker`	JSON String	Controls whether the CDMI server validates the broker's TLS certificate against the configured CA certificate(s). Setting this value to `"false"` disables certificate verification and should not be used in production environments. Permitted values are `"true"` and `"false"`. The default value shall be `"true"`.	Optional
`ca_cert`	JSON String	A PEM-encoded X.509 CA certificate or certificate chain used to verify the broker's TLS certificate. If absent, the CDMI server's system-default trust store shall be used. This field shall not be included if the `ca_cert_id` field is provided.	Optional
`ca_cert_id`	JSON String	A KMIP Certificate object identifier allowing the CDMI server to use the identity of the user to retrieve a CA certificate or certificate chain used to verify the broker's TLS certificate. Only used if the `cdmi_export_mqtt_kmip` system-wide capability is present and set to `true`.	Optional
`client_cert`	JSON String	A PEM-encoded X.509 client certificate to be presented to the broker for mutual TLS authentication. If absent, mutual TLS authentication is not performed. A CDMI server shall not return this value in any read response. This field shall not be included if the `client_cert_id` field is provided.	Optional
`client_cert_id`	JSON String	A KMIP Certificate object identifier allowing the CDMI server to use the identity of the user to retrieve a client certificate to be presented to the broker for mutual TLS authentication. Only used if the `cdmi_export_mqtt_kmip` system-wide capability is present and set to `true`.	Optional
`client_key`	JSON String	The PEM-encoded private key corresponding to `client_cert`. A CDMI server shall not return this value in any read response. If absent, mutual TLS authentication is not performed. This field shall not be included if the `client_key_id` field is provided.	Optional
`client_key_id`	JSON String	A KMIP Private Key object identifier allowing the CDMI server to use the identity of the user to retrieve the private key corresponding to `client_cert_id`. Only used if the `cdmi_export_mqtt_kmip` system-wide capability is present and set to `true`.	Optional
`sni`	JSON String	The server name the CDMI server shall send in the TLS Server Name Indication (SNI) extension of the ClientHello, as defined in RFC 6066. If absent, the CDMI server shall use the host component of `broker_uri`, unless that host is an IP address literal, in which case no SNI extension shall be sent. When present and `verify_broker` is `"true"`, this value shall also be used as the reference identity for broker certificate hostname verification.	Optional

continues on next page

Table 1.2 – continued from previous page

Field Name	Type	Description	Requirement
`alpn_protocols`	JSON Array	An ordered list of Application-Layer Protocol Negotiation (ALPN) protocol identifier strings, as defined in RFC 7301. When present, the CDMI server shall offer the specified ALPN protocol identifier strings in the TLS ClientHello in decreasing order of preference. The IANA-registered identifier for MQTT is `"mqtt"`; broker-specific identifiers may also be used (e.g., `"x-amzn-mqtt-ca"`). If absent, the CDMI server shall not send the ALPN extension. If the broker requires ALPN and selects none of the offered identifiers, the TLS handshake shall fail and the error shall be recorded in `last_problems`.	Optional

1.2.4 Last Will and Testament sub-fields

The `will` JSON object within an MQTT export entry may contain the following sub-fields, which configure the MQTT Last Will and Testament message that the broker publishes if the CDMI server disconnects unexpectedly.

Table 1.3: MQTT Export Last Will and Testament Sub-fields

Field Name	Type	Description	Requirement
`topic`	JSON String	The MQTT topic on which the broker publishes the LWT message on unexpected disconnection. Wildcard characters shall not be used.	Mandatory
`payload`	JSON String	The UTF-8 encoded payload of the LWT message.	Mandatory
`qos`	JSON String	The QoS level at which the broker publishes the LWT message. Permitted values are `"0"`, `"1"`, and `"2"`. The default value shall be `"0"`.	Optional
`retain`	JSON String	Indicates if the broker retains the LWT message on the LWT topic. Permitted values are `"true"` and `"false"`. The default value shall be `"false"`.	Optional

1.2.5 Queue value to MQTT message mapping

Each CDMI queue value is mapped to a single MQTT PUBLISH message as follows:

Table 1.4: CDMI Queue to MQTT Mapping

MQTT 3.1.1	MQTT 5.0	MQTT Use	CDMI Source
Yes	Yes	MQTT message payload.	Raw binary value of enqueued CDMI queue item. If the size of the value exceeds the MQTT maximum packet size, the MQTT message shall be discarded without being published, and the `messages_dropped` counter shall be incremented.
Yes	Yes	MQTT retain.	Value of export `retain` field.
Yes	Yes	MQTT QoS.	Value of export `qos` field.
No	Yes	MQTT Content Type.	If specified in export, value of export `content_type` field. If not specified in export, enqueued CDMI queue item `mimetype` field.
No	Yes	MQTT User Properties.	If specified in export, value of export `user_properties` object.
No	Yes	MQTT Payload Format Indicator.	If enqueued CDMI queue item `mimetype` field includes `"charset=utf-8"`, set to “1”. Otherwise, set to “0”.

1.2.6 Examples

EXAMPLE 1: Add an MQTT export to an existing CDMI queue object:

HTTP request:

```
PATCH /queues/telemetry?exports HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-queue

{
  "exports": {
    "mqtt_primary": {
      "type": "MQTT",
      "broker_uri": "mqtt://broker.example.com:1883",
      "topic": "telemetry/data",
      "client_id": "cdmi-test",
      "protocol_version": "3.1.1",
      "qos": "1",
      "dequeue_on_publish": "true"
    }
  }
}
```

HTTP response:

```
HTTP/1.1 204 NO CONTENT
```

EXAMPLE 2: Retrieve an MQTT export entry, showing CDMI server populated fields:

HTTP request:

```
GET /queues/telemetry?exports HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-queue
```

HTTP response:

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

{
  "exports": {
    "mqtt_primary": {
      "type": "MQTT",
      "broker_uri": "mqtt://broker.example.com:1883",
      "topic": "telemetry/data",
      "client_id": "cdmi-test",
      "reconnect_strategy": "linear",
      "qos": "1",
      "dequeue_on_publish": "true",
      "connected": "true",
      "messages_published": "41892",
      "messages_pending": "0",
      "messages_dropped": "0",
      "last_connected": "2026-11-01T08:15:00.000000Z",
      "last_problems": [ ],
      "last_problem_time": "2026-11-01T08:15:00.000000Z"
    }
  }
}
```

EXAMPLE 3: Modify a value in an existing MQTT export:

HTTP request:

```
PATCH /queues/telemetry?exports HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-queue

{
  "exports": {
    "mqtt_primary": {
      "reconnect_strategy": "exponential"
    }
  }
}
```

HTTP response:

```
HTTP/1.1 204 NO CONTENT
```

EXAMPLE 4: Create an MQTT export with username and password authentication:

HTTP request:

```
PATCH /queues/alerts?exports HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-queue

{
  "exports": {
    "mqtt_auth": {
      "type": "MQTT",
      "broker_uri": "mqtt://broker.example.com:1883",
      "topic": "telemetry/alerts",
      "client_id": "cdmi-test",
      "qos": "2",
      "username": "cdmiserver",
      "password": "..."
    }
  }
}
```

HTTP response:

```
HTTP/1.1 204 NO CONTENT
```

EXAMPLE 5: Create an MQTT export with TLS, mutual certificate authentication, and a Last Will and Testament message:

HTTP request:

```
PATCH /queues/secure-telemetry?exports HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-queue

{
  "exports": {
    "mqtt_tls": {
      "type": "MQTT",
      "broker_uri": "mqtts://broker.example.com:8883",
      "topic": "telemetry/data",
      "client_id": "cdmi-test",
      "qos": "1",
      "username": "cdmiserver",
      "password": "...",
      "tls": {
        "verify_broker": "true",
        "ca_cert": "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----\n",
        "client_cert": "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----\n",
        "client_key": "-----BEGIN RSA PRIVATE KEY-----\n...\n-----END RSA PRIVATE KEY-----\n"
      },
      "will": {
        "topic": "telemetry/status",
        "payload": "{\"status\": \"offline\"}",
        "qos": "1",
        "retain": "true"
      }
    }
  }
}
```

HTTP response:

```
HTTP/1.1 204 NO CONTENT
```

EXAMPLE 6: Retrieve an MQTT export entry, showing a TLS error in the problem detail:

HTTP request:

```
GET /queues/telemetry?exports HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-queue
```

HTTP response:

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

{
  "exports": {
    "mqtt_primary": {
      "type": "MQTT",
      "broker_uri": "mqtts://broker.example.com:8883",
      "topic": "telemetry/data",
      "client_id": "cdmi-test",
      "qos": "1",
      "username": "cdmiserver",
      "tls": {
        "verify_broker": "true",
        "ca_cert": "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----\n"
      },
      "will": {
        "topic": "telemetry/status",
        "payload": "{\"status\": \"offline\"}",
        "qos": "1",
        "retain": "true"
      },
      "connected": "false",
      "messages_published": "0",
      "messages_pending": "4378",
      "messages_dropped": "0",
      "last_problems": [
        {
          "type": "https://snia.org/cdmi/problems/exports/mqtt/tls-handshake-failed",
          "title": "MQTT TLS handshake failed",
          "detail": "The CDMI server could not verify the TLS certificate presented by the
↪broker. The issuer is not trusted by the specified CA. Check the ca_cert or ca_cert_id field.
↪",
          "tls_error_code": "CERTIFICATE_VERIFY_FAILED"
        }
      ],
      "last_problem_time": "2026-11-01T08:15:00.000000Z"
    }
  }
}
```

EXAMPLE 7: Create an MQTT 5.0 export with extended message properties:

HTTP request:

```
PATCH /queues/extended-telemetry?exports HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-queue

{
  "exports": {
    "mqtt5_primary": {
      "type": "MQTT",
      "broker_uri": "mqtts://broker.example.com:8883",
      "topic": "telemetry/data",
      "client_id": "cdmi-test",
      "protocol_version": "5.0",
      "qos": "1",
      "clean_session": "false",
      "session_expiry_interval": "3600",
      "message_expiry_interval": "300",
      "content_type": "application/json",
      "response_topic": "devices/responses",
      "user_properties": [
        ["source", "cdmi"],
        ["region", "us-west-2"]
      ],
      "tls": {
        "verify_broker": "true"
      },
      "username": "cdmiserver",
      "password": "..."
    }
  }
}
```

HTTP response:

```
HTTP/1.1 204 NO CONTENT
```

EXAMPLE 8: Remove an MQTT export entry by setting it to null:

HTTP request:

```
PATCH /queues/telemetry?exports HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-queue

{
  "exports": {
    "mqtt_primary": null
  }
}
```

HTTP response:

```
HTTP/1.1 204 NO CONTENT
```

1.2.7 MQTT export JSON schema

A CDMI server that supports MQTT queue exports shall validate client-supplied `exports` entries with a `type` of `"MQTT"` against the following schema:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "http://snia.org/cdmi/3.0/cdmi-export-mqtt-schema.json#",
  "definitions": {
    "tls_config": {
      "type": "object",
      "properties": {
        "verify_broker": { "type": "string",
                           "enum": ["true", "false"] },
        "ca_cert":        { "type": "string" },
        "ca_cert_id":     { "type": "string" },
        "client_cert":    { "type": "string" },
        "client_cert_id": { "type": "string" },
        "client_key":     { "type": "string" },
        "client_key_id":  { "type": "string" },
        "sni":            { "type": "string" },
        "alpn_protocols": { "type": "array",
                            "items": { "type": "string" },
                            "minItems": 1 }
      },
      "additionalProperties": false
    },
    "will_config": {
      "type": "object",
      "properties": {
        "topic":   { "type": "string" },
        "payload": { "type": "string" },
        "qos":     { "type": "string",
                     "enum": ["0", "1", "2"] },
        "retain":  { "type": "string",
                     "enum": ["true", "false"] }
      },
      "required": [ "topic", "payload" ],
      "additionalProperties": false
    },
    "user_property_pair": {
      "type": "array",
      "items": { "type": "string" },
      "minItems": 2,
      "maxItems": 2
    },
    "export_mqtt": {
      "type": "object",
      "properties": {
        "type":                    { "type": "string",
                                     "enum": ["MQTT"] },
        "broker_uri":              { "type": "string" },
        "topic":                   { "type": "string" },
        "client_id":               { "type": "string" },
        "protocol_version":        { "type": "string",
                                     "enum": ["3.1.1","5.0"] },
        "reconnect_strategy":      { "type": "string",
                                     "enum": ["fixed", "linear", "exponential"] },
        "qos":                     { "type": "string",
                                     "enum": ["0", "1", "2"] },
        "retain":                  { "type": "string",
                                     "enum": ["true", "false"] },
        "clean_session":           { "type": "string",
                                     "enum": ["true", "false"] },
        "keep_alive_interval":     { "type": "string",
                                     "pattern": "^(0|[1-9]\\d{0,3}|[1-5]\\d{4}|6[0-4]\\d{3}
↪|65[0-4]\\d{2}|655[0-2]\\d|6553[0-5])$" },
        "dequeue_on_publish":      { "type": "string",
                                     "enum": ["true", "false"] },
```

(continues on next page)

(continued from previous page)

```
        "username":                    { "type": "string" },
        "password":                    { "type": "string" },
        "password_secret_id":          { "type": "string" },
        "password_refresh_interval":   { "type": "string",
                                         "pattern": "^\\d+$" },
        "tls":                         { "$ref":
                                         "#/definitions/tls_config" },
        "will":                        { "$ref":
                                         "#/definitions/will_config" },
        "session_expiry_interval":     { "type": "string",
                                         "pattern": "^\\d+$" },
        "message_expiry_interval":     { "type": "string",
                                         "pattern": "^\\d+$" },
        "content_type":                { "type": "string" },
        "response_topic":              { "type": "string" },
        "user_properties":             {
                                         "type": "array",
                                         "items": { "$ref": "#/definitions/user_property_pair" }
↪},
        "disabled":                    { "type": "string",
                                         "enum": ["true", "false"] },
        "connected":                   { "type": "string",
                                         "enum": ["true", "false"] },
        "messages_published":          { "type": "string",
                                         "pattern": "^\\d+$" },
        "messages_pending":            { "type": "string",
                                         "pattern": "^\\d+$" },
        "messages_dropped":            { "type": "string",
                                         "pattern": "^\\d+$" },
        "last_connected":              { "type": "string" },
        "last_problems":               { "type": "array",
                                         "items": { "$ref": "http://snia.org/cdmi/3.0/cdmi-
↪problem-details-schema.json#" } },
        "last_problem_time":           { "type": "string" }
      },
      "required": [ "type", "broker_uri", "topic", "client_id" ],
      "additionalProperties": false
    }
  }
}
```

Clause 2

Instructions to the editor

2.1 Overview

In addition to adding the preceding clause, incorporating this extension into the CDMI 2.0 specification requires the following changes to be made to the specification document.

2.2 Cloud storage system-wide capabilities

Add the following entries to the end of the table starting on line 135 of `cdmi_advanced/cdmi_capability_object.txt`, as follows:

Table 2.5: System-wide capabilities

Capability name	Type	Definition
`cdmi_export_mqtt`	JSON String	If present and `"true"`, indicates that the CDMI server supports MQTT queue exports.
`cdmi_export_mqtt_tls`	JSON String	If present and `"true"`, indicates that the CDMI server supports TLS-secured MQTT broker connections for queue exports.
`cdmi_export_mqtt_kmip`	JSON String	If present and `"true"`, indicates that the CDMI server supports connecting to a KMIP server to retrieve passwords and TLS certificates and keys.
`cdmi_export_mqtt_3_1_1`	JSON String	If present and `"true"`, indicates that the CDMI server supports MQTT protocol version 3.1.1 for queue exports.
`cdmi_export_mqtt_5_0`	JSON String	If present and `"true"`, indicates that the CDMI server supports MQTT protocol version 5.0 for queue exports, including all MQTT 5.0 extension fields defined in this clause.

2.3 Queue object capabilities

Add the following entries to the end of the table starting on line 860 of `cdmi_advanced/cdmi_capability_object.txt`, as follows:

Table 2.6: Capabilities for queue objects

Capability name	Type	Definition
`cdmi_export_mqtt`	JSON String	If present and `"true"`, indicates that this queue object may be configured with an MQTT export entry.

2.4 Queue object fields

Add `"exports"` to the list of optional fields in the queue object field table. The semantics of the `exports` field on queue objects shall be as described in this clause. For queue objects, only export entries with `"type": "MQTT"` are defined by this extension; the export protocol types defined in the exported protocols clause of the CDMI 2.0 specification apply only to container objects and shall not be used with queue objects.

2.5 Normative references

Add the following normative references:

IETF RFC 6066, Transport Layer Security (TLS) Extensions: Extension Definitions, `https://www.rfc-editor.org/rfc/rfc6066`

IETF RFC 7301, Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension, `https://www.rfc-editor.org/rfc/rfc7301`

OASIS MQTT Version 3.1.1, `https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html`

OASIS MQTT Version 5.0, `https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html`