Delegated Access Control Extension

Version 1.1f

"Publication of this Working Draft for review and comment has been approved by the Cloud Storage Technical Working Group. This draft represents a "best effort" attempt by the Cloud Storage Technical Working Group to reach preliminary consensus, and it may be updated, replaced, or made obsolete at any time. This document should not be used as reference material or cited as other than a 'work in progress.' Suggestion for revision should be directed to http:/snia.org/feedback."

Working Draft

Revision History

Date	Version	By	Comments
2015-07-20	1.1a	CDMI TWG	Initial draft from Portland Face-to-face for TWG review
2015-11-19	1.1b	CDMI TWG	Updates at the Colorado Springs TWG meeting
2016-01-19	1.1c	CDMI TWG	Updates at the San Jose TWG meeting
2016-03-14	1.1d	CDMI TWG	Updates at the Tuscon TWG meeting
2016-05-10	1.1e	CDMI TWG	Updates at the Colorado Springs TWG meeting
2016-06-15	1.1f	CDMI TWG	Final updates in preparation for public review

Delegated Access Control (DAC) Extension

Overview

The Cloud Data Management Interface (CDMI) specifies metadata in Clause 16.1 that is used to determine if a given operation requested by a given user can be performed against a given object. This metadata is based on the industry-standard NFSv4 ACL system.

This ACL-based model assumes that the cloud server handling the client request has the required authority and knowledge to allow or deny operations. In many use cases, this authority may be resident on a different system, and at this time, no standardized methods are available to delegate the access control decision.

To provide a standardized way that a cloud service can delegate access control decisions to such a third-party system, this extension defines a set of metadata and a series of interactions to indicate when delegated access control (DAC) is to be performed and how the DAC request and response messages are formatted. This extension builds on top of the Encrypted Object Extension but also has several uses for non-encrypted objects.

In the typical data flow, a client sends a request to a CDMI server in order to access or manipulate an object. The CDMI server delegates the access control decision to a DAC provider. Based on the response from the DAC provider, the CDMI server allows or denies the client request and optionally passes through additional private information received from the DAC provider.

To provide interoperable delegated access control, the following areas must be specified:

- How CDMI servers and clients determine when DAC is required for operations against an object
- The way that CDMI servers determine how to contact the DAC provider
- How the DAC provider is sent information about:
    - The operation the client is requesting
    - The object the operation is being requested against
    - The client that is requesting the operation
    - The CDMI server performing the operation
- How the DAC provider:
    - Returns the access control decision to the CDMI server
    - Returns the decryption key to the CDMI server
    - Passes through to the client additional information that may be relevant for the access control decision
- How additional information from the client, which is relevant for the access control decision, can be passed through to the DAC provider
- How the messages exchanged between the CDMI server and the DAC provider are protected

Modifications to the CDMI 1.1 spec:

1) In Clause 2, add reference to the following RFCs:

- RFC 6068, The 'mailto' URI Scheme – https://www.ietf.org/rfc/rfc6068.txt
- RFC 7515, JSON Web Signature (JWS) - https://www.ietf.org/rfc/rfc7515.txt
- RFC 7516, JSON Web Encryption (JWE) - https://www.ietf.org/rfc/rfc7516.txt
- RFC 7517, JSON Web Key (JWK) - https://www.ietf.org/rfc/rfc7517.txt

2) In Clause 3, add the following terms:

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

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

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

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

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

JOSE
Javascript Object Signing and Encryption

JWE
JSON Web Encryption

JWK
JSON Web Key

JWS
JSON Web Signing

3) In Clause 12.1.1, add new rows at end of table "Table 100 - System-Wide Capabilities".

Capability Name	Type	Description
cdmi_dac	JSON String	If present and "true", this capability indicates that the cloud storage system supports delegated access control.

4) In Clause 16.3, add new row at end of table "Table 118 – Storage System Metadata".

Metadata Name	Type	Description	Requirement
cdmi_dac_uri	JSON String	Contains the URI that is used to submit a DAC request for the data object. URI schemes supported shall include "https" and may include "mailto". Both cdmi_dac_certificate and cdmi_dac_uri shall be included for delegated access control to be enabled for a given object.	Optional
cdmi_dac_certificate	JSON Object	A JSON object, containing a JWK which contains a X.509 certificate or certificate chain belonging to the DAC provider, that is used to submit a DAC request for the data object. Both cdmi_dac_certificate and cdmi_dac_uri shall be included for delegated access control to be enabled for a given object.	Optional

5) Add new clause 24, "Access Control"

24.1 Overview

CDMI access control is based around Access Control Lists (ACLs) that are stored as object metadata. When a client requests to perform an operation against a CDMI object, the CDMI server validates the client's identity and credentials against the object ACL to determine if the operation is allowed. This request assumes that the CDMI server is trusted and capable of making these access control decisions.

Figure 1 illustrates an ACL-based access control request:

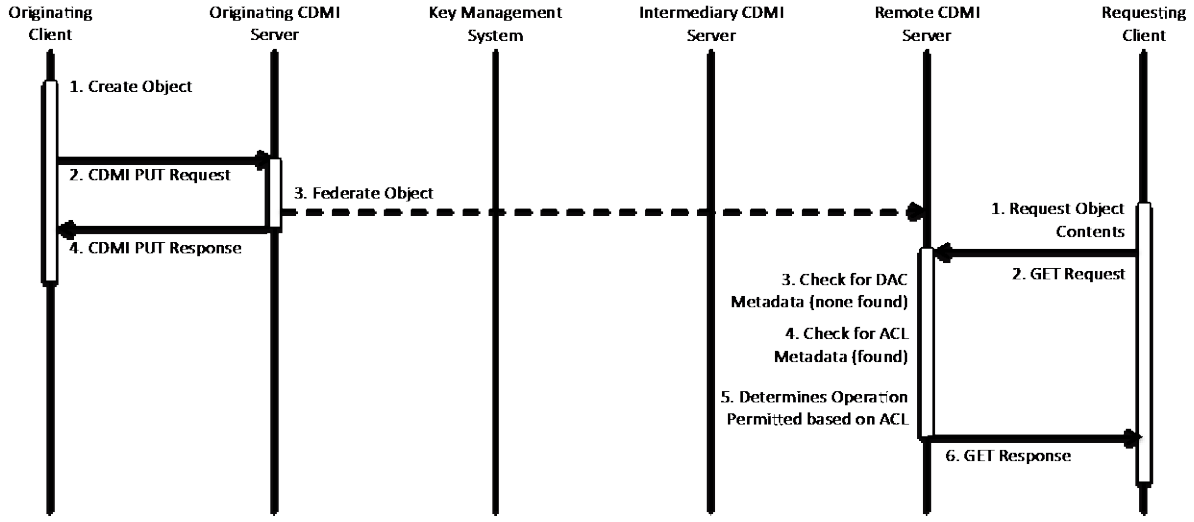

Figure 1 – Non-delegated (ACL-based) access control data flow

When an access control decision needs to be made by a third party (such as by the originating CDMI server in Figure 1), access control is delegated. When cdmi_dac_uri and

cdmi_dac_certificate object metadata is present, as specified in 16.3, Delegated Access Control (DAC) shall be used.

The process by which objects are federated between systems is outside the scope of access control delegation and involves how objects are replicated, synchronized, mirrored, or migrated between CDMI servers. These processes are typically under the control of policies or external policy management systems. Federation is typically performed by third-party systems that use CDMI features including notification, serialization, and the preservation of globally unique object identifiers, which forms the basis for client-transparent interoperability.

24.2 Delegated Access Control (DAC)

A cloud storage system may implement support for DAC, which is indicated by the presence of the cdmi_dac system-wide capability.

DAC enables requests for operations against an object to be allowed or denied by a third-party DAC provider, in addition to ACL access control. When an encrypted object is accessed, the DAC provider may provide the decryption key. The decryption key enables access to encrypted objects, even if the CDMI server cannot access the keys directly.

Clients often have different degrees to which they trust the CDMI server with which they are interacting. Table 1 describes the four ways that DAC shall interact with stored objects.

Table 1. Access Modes for DAC

Mode of Access	Degree of Trust
Client-side decryption	CDMI server is not trusted with keys or to make delegated access control decisions. 1. Client requests object. 2. Client is responsible for custom implementation for getting decryption keys out of band. 3. Client receives ciphertext. 4. Client verifies signatures (if present). 5. Client verifies correct object. 6. Client decrypts object. This mode of access does not use any functionality indicated by the cdmi_dac capability and is supported by all CDMI servers.
Client-side decryption with DAC	CDMI server is not trusted with keys and is used to establish an opaque channel of communication between the client and the DAC provider for key delivery. 1. Client requests object. 2. Client is responsible for custom implementation that gets decryption keys through secure exchange between CDMI server and DAC provider. 3. Client receives ciphertext. 4. Client verifies signatures (if present). 5. Client verifies correct object. 6. Client decrypts object. This mode of access requires the cdm_dac capability but does not require encrypted object support.

	In this mode, data is exchanged between the client and the DAC provider using one or more "CDMI-DAC-" headers, as described in 24.4.
Server-side decryption with DAC	CDMI server is trusted with keys and to delegate access control decisions. DAC message exchange is used to get the decryption keys to decrypt the contents of the object, and keys are not revealed to the client. 1. Client requests object. 2. CDMI server contacts the DAC provider to determine access control decision and gets decryption keys, where the keys are not revealed to the client. 3. CDMI server verifies signatures (if present). 4. CDMI server verifies correct object. 5. CDMI server decrypts object. 6. Client receives plaintext. This mode of access requires DAC and encrypted object support.
Plaintext objects with DAC	CDMI server is trusted with plaintext and to not bypass delegated access control decisions. 1. Client requests object. 2. CDMI server contacts DAC provider to determine access control decision. 3. CDMI server verifies signatures (if present). 4. CDMI server verifies correct object. 5. Client receives plaintext. This mode of access requires DAC support.

The cdmi_dac_uri metadata item indicates where delegated access control requests are submitted, and the cdmi_dac_certificate metadata item indicates how to securely communicate with the delegated access control provider. Both of these metadata items shall be present for DAC to be enabled for a given object.

DAC requests are submitted to a DAC provider using two typical methods:

Direct The DAC request is submitted directly to the absolute URI specified in the cdmi_dac_uri metadata item. This approach requires the host specified in the URI to be accessible from the CDMI server, and for the CDMI server making the request to have sufficient permissions to send the DAC request to that location (for example, HTTP PUT).

Indirect The DAC request is sent to the DAC provider using an indirect route. Indirect routing is useful when the cdmi_dac_uri does not specify a host. An example of indirect routing is when the cdmi_dac_uri contains a mailto URI; the Internet mail system is then responsible for delivering the DAC request.

In other cases, the certificate included with the DAC request (taken from the cdmi_dac_certificate metadata) may be used by intermediary CDMI servers to determine the further routing of the DAC request. For example, DAC requests using a E.U.-issued certificate can be forwarded to a different intermediary CDMI server to

those requests using a U.S.-issued certificate. How certificate fields are used to determine routing is not defined in this International Standard.

Both direct and indirect routing may be synchronous or asynchronous. If a DAC response is not received within the CDMI server or client timeout windows, the client request may time out; however a subsequent request may be processed locally if the DAC response allows response caching. When the CDMI server times out while waiting for a DAC response, it shall return an HTTP status code of 504 Gateway Timeout.

24.3 Delegated Access Control Message Exchange

When a client requests to access or modify an object containing DAC metadata on a CDMI server that supports DAC, the CDMI server shall create and send a DAC request as specified in 24.5. Upon receiving a DAC response as specified in 24.6, the CDMI server shall allow or deny the operation based on the contents of the response.

Figure 2 provides an example of access control delegation for a non-encrypted object. The black solid lines show indirect routing, and gray dashed lines show direct routing.

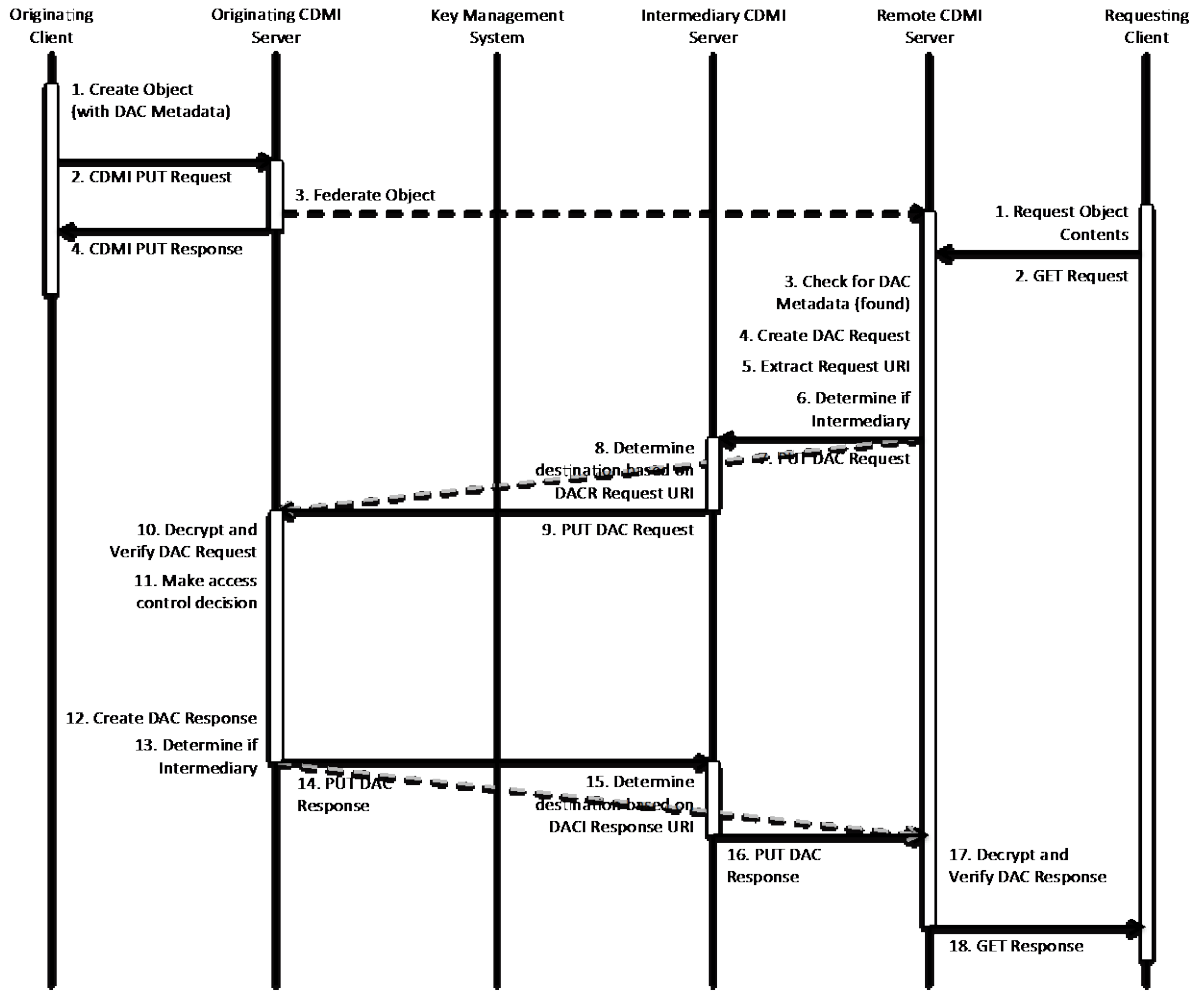

Figure 2 – Delegated access control data flow example for non-encrypted object

For non-encrypted objects, an originating client indicates that DAC is requested by including the DAC metadata items. It is important to emphasize that for non-encrypted objects, DAC cannot

be guaranteed to be enforced, as when an object with DAC metadata is accessed from a CDMI server that does not support DAC; only ACL-based access control shall be evaluated.

Figure 3 provides a second example of access control delegation for an encrypted object. The black solid lines show indirect routing, and gray dashed lines show direct routing.

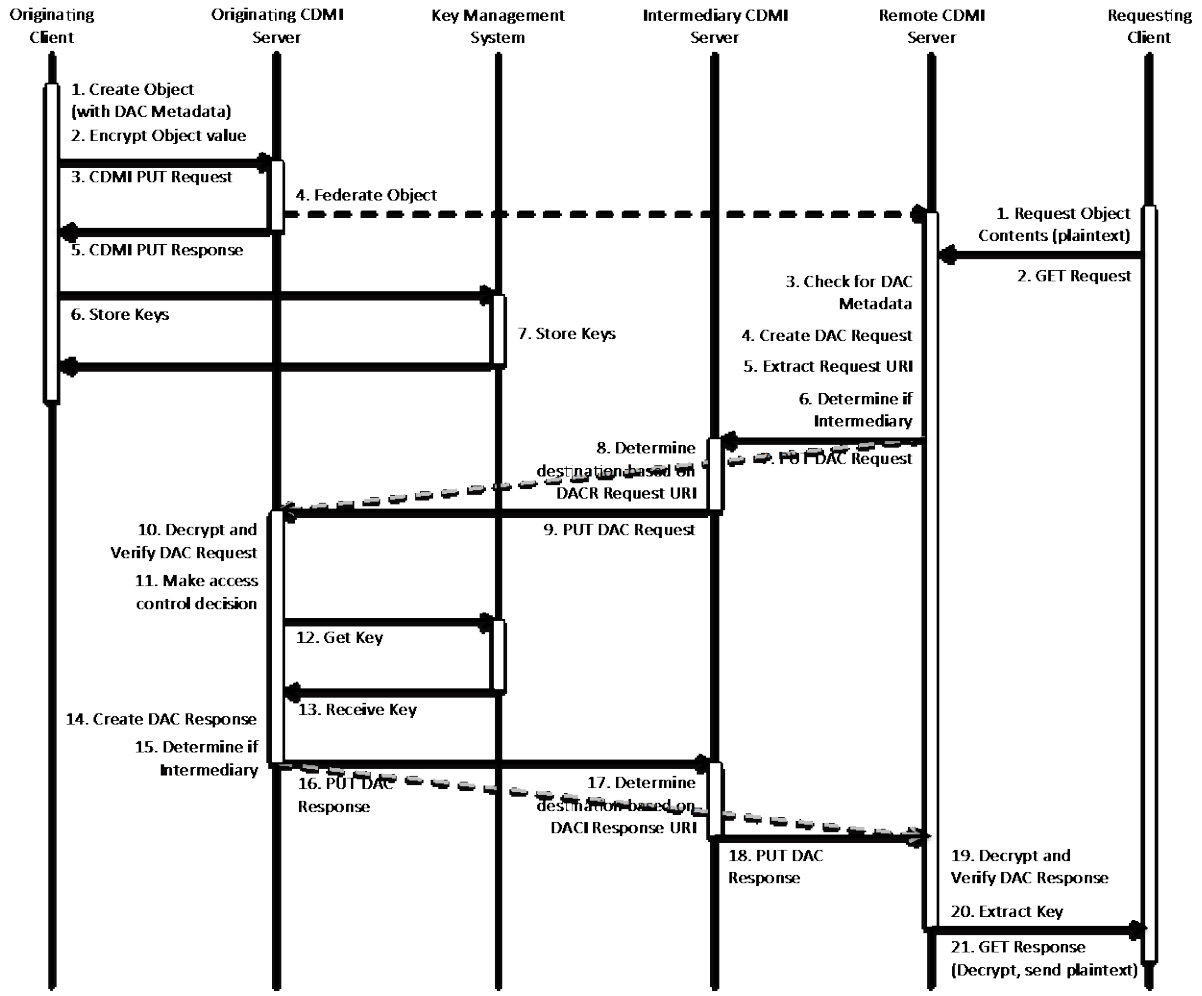

Figure 3 – Delegated access control data flow example for encrypted object

For encrypted objects, as access to the decryption keys are provided in the DAC response, the plaintext is inaccessible unless the CDMI server supports DAC.

When the DAC provider processes the DAC request, if the operation is allowed and the key is requested by the CDMI server, the object key, if present, shall be obtained and sent back as part of the DAC response. Upon receiving the DAC response, the CDMI server shall extract the key to perform the client operation.

24.4 Client Header Passthrough

The Delegated Access Control extension provides facilities to allow client-provided HTTP request headers to be passed through to the DAC provider, and for the DAC provider to pass HTTP response headers back to the client. These headers are identified by the "CDMI-DAC-" prefix.

The contents and full names of these headers are not defined in this International Standard. However, it is anticipated that these headers shall be used to allow the client to provide additional information that may be required for the access control decision-making process, for audit purposes, or for secure key exchange.

For example, when an operation is allowed by a DAC provider, the object key may be encrypted using the public key from a client-provided certificate (verified by the DAC provider), which is included in a "CDMI-DAC-" request header, with the encrypted object key being sent back to the client in a "CDMI-DAC-" response header. In this scenario, the CDMI server cannot decrypt the ciphertext but can securely pass on the encrypted object key to the client. The client can then use its private key to decrypt the response header to get the object key, which can then be used to decrypt the object.

24.5 DAC Request

When a CDMI server that supports DAC needs to contact the DAC provider as specified in the DAC metadata, it constructs a DAC request, as specified below:

Field Name	Type	Description	Requirement
dac_request_version	JSON String	Indicates the version of the DAC request. This field is currently set to the value "1".	Mandatory
dac_request_id	JSON String	Contains a system-specified identifier that is used to match up the corresponding DAC response. This identifier shall be unique within the window that multiple DAC responses may be received.	Mandatory
server_identity	JSON Object	A JSON object containing a JWK, which contains a X.509 certificate or certificate chain belonging to the CDMI server that is generating the DAC request. This certificate is used to ensure that only the CDMI Server that generated the DAC request can read the DAC response.	Mandatory
client_identity	JSON Object	A JSON object containing the following JSON entities: JSON String, "acl_name", containing the ACL name of the client requesting the operation. JSON Array, "acl_group", containing the ACL group(s) of the client requesting the operation.	Mandatory
acl_effective_mask	JSON String	A text or hexadecimal string representation of the ACE mask that shall be used for the operation, as defined in 16.1.5.	Mandatory
client_headers	JSON Object	A JSON object containing a JSON string for each HTTP header in the operation request that starts with "CDMI-DAC-", where the JSON string name is the header name, and the JSON string value is the header value. These headers can be used for tunneling information from the client to the DAC provider.	Mandatory
cdmi_objectID	JSON	Contains the object ID of the object the operation is	Mandatory

	String	performed against.	
cdmi_enc_keyID	JSON String	Contains the encryption key identifier, which is used to indicate that the CDMI server is requesting the encryption key. Contains the unique key identifier (for example, a KMIP identifier) for the symmetric key that is used to encrypt and decrypt the object.	Optional
cdmi_operation	JSON String	Contains a string indicating which operation is being requested to be performed against the object. The following operations are defined: "cdmi_read" "cdmi_modify" "cdmi_delete"	Mandatory
dac_response_uri	JSON String	An optional URI that specifies where to send the DAC response. This URI is required for asynchronous DAC requests, such as when sent via email URIs. If this field is omitted, the DAC response shall be based on the context of the request, for example, as a message body returned for the request PUT when using HTTPS, or an email reply when using a mailto URI.	Optional

An example of a DAC request is shown below:

```
{
   "dac_request_version" : "1",
   "dac_request_id" : "F55AA0B6-8F54-4A03-AC21-87052D58485A",
   "server_identity" : {
      "kty":"RSA",
      "use":"sig",
      "kid":"1b94c",
      "n":"vrjOfz9Ccdgx5nQudyhdoR17V-IubWMeOZCwX_jj0hgA
           sz2J_pqYW08PLbK_PdiVGKPrqzmDIsLI7sA25VEnHU1u
           CLNwBuUiCO11_-7dYbsr4iJmG0Qu2j8DsVyT1azpJC_N
           G84Ty5KKthuCaPod7iI7w0LK9orSMhBEwwZDCxTWq4aY
           WAchc8t-emd9qOvWtVMDC2BXksRngh6X5bUYLy6AyHKv
           j-nUy1wgzjYQDwHMTplCoLtU-o-8SNnZ1tmRoGE9uJkB
           Ldh5gFENabWnU5m1ZqZPdwS-qo-meMvVfJb6jJVWRpl2
           SUtCnYG2C32qvbWbjZ_jBPD5eunqsIo1vQ",
      "e":"AQAB",
      "x5c": [
          "MIIDQjCCAiqgAwIBAgIGATz/FuLiMA0GCSqGSIb3DQEB
           BQUAMGIxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJDTzEP
           MA0GA1UEBxMGRGVudmVyMRwwGgYDVQQKExNQaW5nIElk
           ZW50aXR5IENvcnAuMRcwFQYDVQQDEw5CcmlhbiBDYW1w
           YmVsbDAeFw0xMzAyMjEyMzI5MTVaFw0xODA4MTQyMjI5
           MTVaMGIxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJDTzEP
           MA0GA1UEBxMGRGVudmVyMRwwGgYDVQQKExNQaW5nIElk
           ZW50aXR5IENvcnAuMRcwFQYDVQQDEw5CcmlhbiBDYW1w
           YmVsbDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC
           ggEBAL64zn8/QnHYMeZ0LncoXaEde1fiLm1jHjmQsF/4
```

```
            49IYALM9if6amFtPDy2yvz3YlRij66s5gyLCyO7ANuVR
            Jx1NbgizcAblIgjtdf/u3WG7K+IiZhtELto/A7Fck9Ws
            6SQvzRvOE8uSirYbgmj6He4iO8NCyvaK0jIQRMMGQwsU
            1quGmFgHIXPLfnpnfajr1rVTAwtgV5LEZ4Iel+W1GC8u
            gMhyr4/p1MtcIM42EA8BzE6ZQqC7VPqPvEjZ2dbZkaBh
            PbiZAS3YeYBRDWm1p1OZtWamT3cEvqqPpnjL1XyW+oyV
            VkaZdklLQp2Btgt9qr21m42f4wTw+Xrp6rCKNb0CAwEA
            ATANBgkqhkiG9w0BAQUFAAOCAQEAh8zGlfSlcI0o3rYD
            PBB07aXNswb4ECNIKG0CETTUxmXl9KUL+9gGlqCz5iWL
            OgWsnrcKcY0vXPG9J1r9AqBNTqNgHq2G03X09266X5Cp
            Oe1zFo+Owb1zxtp3PehFdfQJ610CDLEaS9V9Rqp17hCy
            ybEpOGVwe8fnk+fbEL2Bo3UPGrpsHzUoaGpDftmWssZk
            hpBJKVMJyf/RuP2SmmaIzmnw9JiSlYhzo4tpzd5rFXhj
            Rbg4zW9C+2qok+2+qDM1iJ684gPHMIY8aLWrdgQTxkum
            GmTqgawR+N5MDtdPTEQ0XfIBc2cJEUyMTY5MPvACWpkA
            6SdS4xSvdXK3IVfOWA=="
         ]
      }
   }
   "client_identity" : {
      "acl_name" : "jdoe",
      "acl_group" : ["users"]
   },
   "acl_effective_mask" : "READ_ALL"
   "client_headers" : {
      "CDMI-DAC-TEST" : "Testing"
   },
   "cdmi_objectid" : "00007ED90010D891022876A8DE0BC0FD",
   "cdmi_enc_keyID" : "testkey",
   "cdmi_event_type" : "cdmi_read",
   "dac_response_uri" : "https://cloud.example.com/dacr"
}
```

The above JSON (DAC request) is encrypted in JWE format, where the recipient is the public key of the DAC provider certificate (as specified in the DAC object metadata), and is JWS signed using the private key of the CDMI server that corresponds to the server identity certificate included in the DAC request. The certificate of the DAC provider from the object is then attached:

Field Name	Type	Description	Requirement
dac_request	JSON Object	JOSE encrypted and signed request	Mandatory
dac_request_dest_certificate	JSON Object	The cdmi_dac_certificate metadata value, (a JSON object containing a JWK, which contains a X.509 certificate or certificate chain belonging to the DAC provider) indicating where the DAC request is to be sent.	Mandatory
dac_request_dest_uri	JSON String	The cdmi_dac_uri metadata value indicating where the DAC request is to be sent.	Mandatory

Once created, the DAC request is submitted using the DAC request URI specified in the DAC object metadata, for example, as an HTTP PUT operation or via an SMTP email. The

dac_request_dest_certificate and dac_request_dest_uri can also be used to route the request through intermediary hops if needed.

24.6 DAC Response

When a DAC provider receives a DAC request, it decrypts the request using its private key, verifies the signature of the CDMI server, and evaluates the request. Based on the information provided, the DAC provider shall allow or deny operations by modifying or replacing the ACL mask that was initially determined by the CDMI server.

To indicate the result of the DAC request to the requesting CDMI server, the DAC provider constructs a DAC response, as specified below:

Field Name	Type	Description	Requirement
dac_response_version	JSON String	Indicates the version of the DAC response. Currently set to the value "1".	Mandatory
dac_response_id	JSON String	Contains the system-specified identifier specified in the corresponding dac_request_id.	Mandatory
dac_identity	JSON Object	A JSON object containing a JWK, which contains a X.509 certificate or certificate chain belonging to the DAC provider that is generating the DAC response. This certificate is used to ensure that the CDMI server can validate the identity of the DAC provider.	Mandatory
dac_applied_mask	JSON String	A text or hexadecimal string representation of the ACE mask to be used, as defined in 16.1.5.	Mandatory
dac_object_key	JSON Object	The key for the object in JWK format (See RFC 7517). This key is only disclosed when cdmi_enc_keyID is included in the DAC request and the DAC provider allows access.	Optional
dac_response_headers	JSON Object	A series of headers that start with "CDMI-DAC-" to be returned to the client. These headers can be used to pass information from the DAC provider back to the client.	Optional
dac_key_cache_expiry	JSON String	The complete date/time when the object key is no longer to be cached, specified in ISO 8601 date/time format. If this field is not included, the key shall not be cached.	Optional

dac_response_cache_expiry	JSON String	The complete date/time when the DAC response is no longer to be cached, specified in ISO 8601 date/time format. If this field is not included, the response shall not be cached.	Optional
dac_redirect_objectID	JSON String	Indicates an alternate CDMI Object ID used to access the requested object. If present, the CDMI server shall send an HTTP Redirect to the client.	Optional
dac_audit_uri	JSON String	Indicates a URI to a CDMI queue where audit logging messages associated with the operations shall be submitted. When present, audit logging messages shall be generated for receiving the response, performing the operation, and determining when to purge the key (see 20.2).	Optional

An example of a DAC response is shown below:

```
{
   "dac_response_version" : "1",
   "dac_response_id" : "<identifier of request>",
   "dac_identity" : {
      "kty":"RSA",
      "use":"sig",
      "kid":"1b94c",
      "n":"vrjOfz9Ccdgx5nQudyhdoR17V-IubWMeOZCwX_jj0hgA
           sz2J_pqYW08PLbK_PdiVGKPrqzmDIsLI7sA25VEnHU1u
           CLNwBuUiCO11_-7dYbsr4iJmG0Qu2j8DsVyT1azpJC_N
           G84Ty5KKthuCaPod7iI7w0LK9orSMhBEwwZDCxTWq4aY
           WAchc8t-emd9qOvWtVMDC2BXksRngh6X5bUYLy6AyHKv
           j-nUy1wgzjYQDwHMTplCoLtU-o-8SNnZ1tmRoGE9uJkB
           Ldh5gFENabWnU5m1ZqZPdwS-qo-meMvVfJb6jJVWRpl2
           SUtCnYG2C32qvbWbjZ_jBPD5eunqsIo1vQ",
      "e":"AQAB",
      "x5c": [
          "MIIDQjCCAiqgAwIBAgIGATz/FuLiMA0GCSqGSIb3DQEB
           BQUAMGIxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJDTzEP
           MA0GA1UEBxMGRGVudmVyMRwwGgYDVQQKExNQaW5nIElk
           ZW50aXR5IENvcnAuMRcwFQYDVQQDEw5CcmlhbiBDYW1w
           YmVsbDAeFw0xMzAyMjEyMzI5MTVaFw0xODA4MTQyMjI5
           MTVaMGIxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJDTzEP
           MA0GA1UEBxMGRGVudmVyMRwwGgYDVQQKExNQaW5nIElk
           ZW50aXR5IENvcnAuMRcwFQYDVQQDEw5CcmlhbiBDYW1w
           YmVsbDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC
           ggEBAL64zn8/QnHYMeZ0LncoXaEde1fiLm1jHjmQsF/4
           49IYALM9if6amFtPDy2yvz3YlRij66s5gyLCyO7ANuVR
           Jx1NbgizcAblIgjtdf/u3WG7K+IiZhtELto/A7Fck9Ws
           6SQvzRvOE8uSirYbgmj6He4iO8NCyvaK0jIQRMMGQwsU
           1quGmFgHIXPLfnpnfajr1rVTAwtgV5LEZ4Iel+W1GC8u
           gMhyr4/p1MtcIM42EA8BzE6ZQqC7VPqPvEjZ2dbZkaBh
           PbiZAS3YeYBRDWm1p1OZtWamT3cEvqqPpnjL1XyW+oyV
           VkaZdklLQp2Btgt9qr21m42f4wTw+Xrp6rCKNb0CAwEA
```

```
            ATANBgkqhkiG9w0BAQUFAAOCAQEAh8zGlfSlcI0o3rYD
            PBB07aXNswb4ECNIKG0CETTUxmXl9KUL+9gGlqCz5iWL
            OgWsnrcKcY0vXPG9J1r9AqBNTqNgHq2G03X09266X5Cp
            Oe1zFo+Owb1zxtp3PehFdfQJ610CDLEaS9V9Rqp17hCy
            ybEpOGVwe8fnk+fbEL2Bo3UPGrpsHzUoaGpDftmWssZk
            hpBJKVMJyf/RuP2SmmaIzmnw9JiSlYhzo4tpzd5rFXhj
            Rbg4zW9C+2qok+2+qDM1iJ684gPHMIY8aLWrdgQTxkum
            GmTqgawR+N5MDtdPTEQ0XfIBc2cJEUyMTY5MPvACWpkA
            6SdS4xSvdXK3IVfOWA=="
         ]
      }
   }
   "dac_effective_mask" : "ALL_PERMS",
   "dac_object_key" {
      "kty" : "oct",
      "alg" : "A128KW",
      "k" : "GawgguFyGrWKav7AX4VKUg"}
   },
   "dac_key_cache_expiry" : "2015-07-20T14:12:44.835294Z",
   "dac_response_cache_expiry" : "2015-07-20T14:12:44.835294Z",
   "dac_audit_uri" : "<URI to Audit Queue>"
}
```

The above JSON (DAC response) is encrypted in JWE format where the recipient is the public key of the CDMI server certificate (as specified in the DAC request), and is JWS-signed using the private key of the DAC provider that corresponds to the DAC identity certificate that is included in the DAC response.

Once created, the DAC response is returned to the CDMI server or is submitted to the dac_response_uri specified in the DAC request. The certificate of the server included with the DAC request is then attached:

Field Name	Type	Description	Requirement
dac_response	JSON Object	JOSE encrypted and signed response	Mandatory
dac_response_dest_certificate	JSON Object	A JSON object containing a JWK, which contains a X.509 certificate or certificate chain belonging to the server that initiated the DAC requester (taken from the DAC request)	Mandatory
dac_response_dest_uri	JSON String	A URI indicating where the DAC response is to be sent (taken from the DAC request)	Mandatory

Once created, the DAC response is submitted using the DAC response URI specified in the DAC request, for example, as an HTTP PUT operation or via an SMTP email. The dac_response_dest_certificate and dac_response_dest_uri can also be used to route the request through intermediary hops if needed.

When the CDMI server receives a DAC response message, it shall decrypt it using its private key and verify the signature using the public key from the object's DAC metadata. If the decryption and signature verification are successful, the CDMI server shall use the provided dac_applied_mask in place of the ACL computed mask.

If the CDMI server supports key or DAC response caching, cache expiry values shall be honored. Cached responses and keys can only be used for identical client operations, where the client identity, objectID, operation, and "CDMI-DAC-" request headers are identical. Otherwise, the cached response shall be expired. If an audit URI is present in the cached response, audit messages shall also be generated for all operations allowed using the cached response.

The CDMI server shall also implement audit logging as specified in the DAC response.

If a dac_redirect_objectID field is returned in the DAC response, the CDMI server shall return an HTTP redirect to the specified Object ID. This redirect allows a DAC provider to create a client operation-specific instance of the object that is encrypted with a single-use key so that the object key is not disclosed.

24.7 Error Handling

In the following scenarios, the following HTTP response codes shall be returned to a client:

- When a DAC response denies the requested operation, an HTTP status code of 403 Forbidden shall be returned to the client along with any dac_response_headers included in the response.
- When a DAC response includes a dac_redirect_objectID, an HTTP status code of 302 Found shall be returned to the client along with any dac_response_headers included in the response.
- When a DAC request to access or modify an encrypted object is allowed, but the key is not included in the DAC response, an HTTP status code of 401 Unauthorized shall be returned to the client along with any dac_response_headers included in the response.
- When a DAC request to access or modify an encrypted object is allowed, but cannot be performed due to lack of support for an encryption algorithm, signing algorithm, or key type, an HTTP status code of 501 Not Implemented shall be returned along with any dac_response_headers included in the response.
- When a DAC request times out, an HTTP status code of 500 Internal Server Error shall be returned to the client.
- When a DAC request cannot be sent or routed because the DAC metadata is not supported or valid, an HTTP status code of 501 Not Implemented shall be returned to the client.
- When a DAC request cannot be sent or routed because an upstream system is unavailable, an HTTP status code of 500 Internal Server Error shall be returned to the client.

24.8 Examples

The following examples illustrate the primary ways that DAC requests are performed.

24.8.1 GET ciphertext of encrypted object with delegated access control

The following CDMI operation is performed against an encrypted CDMI object with delegated access control metadata:

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

The CDMI server verifies local access controls and determines that the request can proceed. The following DAC request is generated:

```
{
   "dac_request_version" : "1",
   "dac_request_id" : "F55AA0B6-8F54-4A03-AC21-87052D58485A",
   "server_identity" : {
      "kty":"RSA",
      "use":"sig",
      "kid":"1b94c",
      "n":"vrjOfz9Ccdgx5nQudyhdoR17V-IubWMeOZCwX_jj0hgA
           sz2J_pqYW08PLbK_PdiVGKPrqzmDIsLI7sA25VEnHU1u
           CLNwBuUiCO11_-7dYbsr4iJmG0Qu2j8DsVyT1azpJC_N
           G84Ty5KKthuCaPod7iI7w0LK9orSMhBEwwZDCxTWq4aY
           WAchc8t-emd9qOvWtVMDC2BXksRngh6X5bUYLy6AyHKv
           j-nUy1wgzjYQDwHMTplCoLtU-o-8SNnZ1tmRoGE9uJkB
           Ldh5gFENabWnU5m1ZqZPdwS-qo-meMvVfJb6jJVWRpl2
           SUtCnYG2C32qvbWbjZ_jBPD5eunqsIo1vQ",
      "e":"AQAB",
      "x5c": [
          "MIIDQjCCAiqgAwIBAgIGATz/FuLiMA0GCSqGSIb3DQEB
           BQUAMGIxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJDTzEP
           MA0GA1UEBxMGRGVudmVyMRwwGgYDVQQKExNQaW5nIElk
           ZW50aXR5IENvcnAuMRcwFQYDVQQDEw5CcmlhbiBDYW1w
           YmVsbDAeFw0xMzAyMjEyMzI5MTVaFw0xODA4MTQyMjI5
           MTVaMGIxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJDTzEP
           MA0GA1UEBxMGRGVudmVyMRwwGgYDVQQKExNQaW5nIElk
           ZW50aXR5IENvcnAuMRcwFQYDVQQDEw5CcmlhbiBDYW1w
           YmVsbDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC
           ggEBAL64zn8/QnHYMeZ0LncoXaEde1fiLm1jHjmQsF/4
           49IYALM9if6amFtPDy2yvz3YlRij66s5gyLCyO7ANuVR
           Jx1NbgizcAblIgjtdf/u3WG7K+IiZhtELto/A7Fck9Ws
           6SQvzRvOE8uSirYbgmj6He4iO8NCyvaK0jIQRMMGQwsU
           1quGmFgHIXPLfnpnfajr1rVTAwtgV5LEZ4Iel+W1GC8u
           gMhyr4/p1MtcIM42EA8BzE6ZQqC7VPqPvEjZ2dbZkaBh
           PbiZAS3YeYBRDWm1p1OZtWamT3cEvqqPpnjL1XyW+oyV
           VkaZdklLQp2Btgt9qr21m42f4wTw+Xrp6rCKNb0CAwEA
           ATANBgkqhkiG9w0BAQUFAAOCAQEAh8zGlfSlcI0o3rYD
           PBB07aXNswb4ECNIKG0CETTUxmXl9KUL+9gGlqCz5iWL
           OgWsnrcKcY0vXPG9J1r9AqBNTqNgHq2G03X09266X5Cp
           Oe1zFo+Owb1zxtp3PehFdfQJ610CDLEaS9V9Rqp17hCy
           ybEpOGVwe8fnk+fbEL2Bo3UPGrpsHzUoaGpDftmWssZk
           hpBJKVMJyf/RuP2SmmaIzmnw9JiSlYhzo4tpzd5rFXhj
           Rbg4zW9C+2qok+2+qDM1iJ684gPHMIY8aLWrdgQTxkum
           GmTqgawR+N5MDtdPTEQ0XfIBc2cJEUyMTY5MPvACWpkA
           6SdS4xSvdXK3IVfOWA=="
        ]
     }
   }
   "client_identity" : {
      "acl_name" : <acl name of client>
      "acl_group" : [<acl groups of clients>]
   },
```

```
   "acl_effective_mask" : <acl mask of client>,
   "cdmi_objectid" : <CDMI Object ID of object being accessed>,
   "cdmi_event_type" : "cdmi_read",
   "dac_response_uri" : "https://cloud.example.com/dacr"
}
```

The DAC provider processes the request and returns the following DAC response:

```
{
   "dac_response_version" : "1",
   "dac_response_id" : "F55AA0B6-8F54-4A03-AC21-87052D58485A",
   "dac_identity" : {
      "kty":"RSA",
      "use":"sig",
      "kid":"1b94c",
      "n":"vrjOfz9Ccdgx5nQudyhdoR17V-IubWMeOZCwX_jj0hgA
           sz2J_pqYW08PLbK_PdiVGKPrqzmDIsLI7sA25VEnHU1u
           CLNwBuUiCO11_-7dYbsr4iJmG0Qu2j8DsVyT1azpJC_N
           G84Ty5KKthuCaPod7iI7w0LK9orSMhBEwwZDCxTWq4aY
           WAchc8t-emd9qOvWtVMDC2BXksRngh6X5bUYLy6AyHKv
           j-nUy1wgzjYQDwHMTplCoLtU-o-8SNnZ1tmRoGE9uJkB
           Ldh5gFENabWnU5m1ZqZPdwS-qo-meMvVfJb6jJVWRpl2
           SUtCnYG2C32qvbWbjZ_jBPD5eunqsIo1vQ",
      "e":"AQAB",
      "x5c": [
          "MIIDQjCCAiqgAwIBAgIGATz/FuLiMA0GCSqGSIb3DQEB
           BQUAMGIxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJDTzEP
           MA0GA1UEBxMGRGVudmVyMRwwGgYDVQQKExNQaW5nIElk
           ZW50aXR5IENvcnAuMRcwFQYDVQQDEw5CcmlhbiBDYW1w
           YmVsbDAeFw0xMzAyMjEyMzI5MTVaFw0xODA4MTQyMjI5
           MTVaMGIxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJDTzEP
           MA0GA1UEBxMGRGVudmVyMRwwGgYDVQQKExNQaW5nIElk
           ZW50aXR5IENvcnAuMRcwFQYDVQQDEw5CcmlhbiBDYW1w
           YmVsbDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC
           ggEBAL64zn8/QnHYMeZ0LncoXaEde1fiLm1jHjmQsF/4
           49IYALM9if6amFtPDy2yvz3YlRij66s5gyLCyO7ANuVR
           Jx1NbgizcAblIgjtdf/u3WG7K+IiZhtELto/A7Fck9Ws
           6SQvzRvOE8uSirYbgmj6He4iO8NCyvaK0jIQRMMGQwsU
           1quGmFgHIXPLfnpnfajr1rVTAwtgV5LEZ4Iel+W1GC8u
           gMhyr4/p1MtcIM42EA8BzE6ZQqC7VPqPvEjZ2dbZkaBh
           PbiZAS3YeYBRDWm1p1OZtWamT3cEvqqPpnjL1XyW+oyV
           VkaZdklLQp2Btgt9qr21m42f4wTw+Xrp6rCKNb0CAwEA
           ATANBgkqhkiG9w0BAQUFAAOCAQEAh8zGlfSlcI0o3rYD
           PBB07aXNswb4ECNIKG0CETTUxmXl9KUL+9gGlqCz5iWL
           OgWsnrcKcY0vXPG9J1r9AqBNTqNgHq2G03X09266X5Cp
           Oe1zFo+Owb1zxtp3PehFdfQJ610CDLEaS9V9Rqp17hCy
           ybEpOGVwe8fnk+fbEL2Bo3UPGrpsHzUoaGpDftmWssZk
           hpBJKVMJyf/RuP2SmmaIzmnw9JiSlYhzo4tpzd5rFXhj
           Rbg4zW9C+2qok+2+qDM1iJ684gPHMIY8aLWrdgQTxkum
           GmTqgawR+N5MDtdPTEQ0XfIBc2cJEUyMTY5MPvACWpkA
           6SdS4xSvdXK3IVfOWA=="
        ]
     }
   }
   "dac_effective_mask" : "ALL_PERMS"
}
```

Since the operation is allowed by the DAC provider, the following response is sent:

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

<CMS Encrypted Object>
```

24.8.2 GET ciphertext of encrypted object with passthrough key access

The following CDMI operation is performed against an encrypted CDMI object with delegated access control metadata:

```
GET /MyContainer/MyEncryptedObject.txt HTTP/1.1
Host: cloud.example.com
Accept: application/cms, application/jose+json
CDMI-DAC-N: <client-specific headers requesting decryption key>
```

The CDMI server verifies local access controls and determines that the request can proceed. The following DAC request is generated:

```
{
   "dac_request_version" : "1",
   "dac_request_id" : "F55AA0B6-8F54-4A03-AC21-87052D58485A",
   "server_identity" : {
      "kty":"RSA",
      "use":"sig",
      "kid":"1b94c",
      "n":"vrjOfz9Ccdgx5nQudyhdoR17V-IubWMeOZCwX_jj0hgA
           sz2J_pqYW08PLbK_PdiVGKPrqzmDIsLI7sA25VEnHU1u
           CLNwBuUiCO11_-7dYbsr4iJmG0Qu2j8DsVyT1azpJC_N
           G84Ty5KKthuCaPod7iI7w0LK9orSMhBEwwZDCxTWq4aY
           WAchc8t-emd9qOvWtVMDC2BXksRngh6X5bUYLy6AyHKv
           j-nUy1wgzjYQDwHMTplCoLtU-o-8SNnZ1tmRoGE9uJkB
           Ldh5gFENabWnU5m1ZqZPdwS-qo-meMvVfJb6jJVWRpl2
           SUtCnYG2C32qvbWbjZ_jBPD5eunqsIo1vQ",
      "e":"AQAB",
      "x5c": [
          "MIIDQjCCAiqgAwIBAgIGATz/FuLiMA0GCSqGSIb3DQEB
           BQUAMGIxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJDTzEP
           MA0GA1UEBxMGRGVudmVyMRwwGgYDVQQKExNQaW5nIElk
           ZW50aXR5IENvcnAuMRcwFQYDVQQDEw5CcmlhbiBDYW1w
           YmVsbDAeFw0xMzAyMjEyMzI5MTVaFw0xODA4MTQyMjI5
           MTVaMGIxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJDTzEP
           MA0GA1UEBxMGRGVudmVyMRwwGgYDVQQKExNQaW5nIElk
           ZW50aXR5IENvcnAuMRcwFQYDVQQDEw5CcmlhbiBDYW1w
           YmVsbDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC
           ggEBAL64zn8/QnHYMeZ0LncoXaEde1fiLm1jHjmQsF/4
           49IYALM9if6amFtPDy2yvz3YlRij66s5gyLCyO7ANuVR
           Jx1NbgizcAblIgjtdf/u3WG7K+IiZhtELto/A7Fck9Ws
           6SQvzRvOE8uSirYbgmj6He4iO8NCyvaK0jIQRMMGQwsU
           1quGmFgHIXPLfnpnfajr1rVTAwtgV5LEZ4Iel+W1GC8u
           gMhyr4/p1MtcIM42EA8BzE6ZQqC7VPqPvEjZ2dbZkaBh
           PbiZAS3YeYBRDWm1p1OZtWamT3cEvqqPpnjL1XyW+oyV
           VkaZdklLQp2Btgt9qr21m42f4wTw+Xrp6rCKNb0CAwEA
           ATANBgkqhkiG9w0BAQUFAAOCAQEAh8zGlfSlcI0o3rYD
           PBB07aXNswb4ECNIKG0CETTUxmXl9KUL+9gGlqCz5iWL
```

```
            OgWsnrcKcY0vXPG9J1r9AqBNTqNgHq2G03X09266X5Cp
            Oe1zFo+Owb1zxtp3PehFdfQJ610CDLEaS9V9Rqp17hCy
            ybEpOGVwe8fnk+fbEL2Bo3UPGrpsHzUoaGpDftmWssZk
            hpBJKVMJyf/RuP2SmmaIzmnw9JiSlYhzo4tpzd5rFXhj
            Rbg4zW9C+2qok+2+qDM1iJ684gPHMIY8aLWrdgQTxkum
            GmTqgawR+N5MDtdPTEQ0XfIBc2cJEUyMTY5MPvACWpkA
            6SdS4xSvdXK3IVfOWA=="
         ]
      }
   }
   "client_identity" : {
      "acl_name" : <acl name of client>
      "acl_group" : [<acl groups of clients>]
   },
   "acl_effective_mask" : <acl mask of client>
   "client_headers" : {
      <CDMI-DAC- headers from client>
   },
   "cdmi_objectid" : <CDMI Object ID of object being accessed>,
   "cdmi_event_type" : "cdmi_read",
   "dac_response_uri" : "https://cloud.example.com/dacr"
}
```

The DAC provider processes the request and returns the following DAC response:

```
{
   "dac_response_version" : "1",
   "dac_response_id" : "F55AA0B6-8F54-4A03-AC21-87052D58485A",
   "dac_identity" : {
      "kty":"RSA",
      "use":"sig",
      "kid":"1b94c",
      "n":"vrjOfz9Ccdgx5nQudyhdoR17V-IubWMeOZCwX_jj0hgA
           sz2J_pqYW08PLbK_PdiVGKPrqzmDIsLI7sA25VEnHU1u
           CLNwBuUiCO11_-7dYbsr4iJmG0Qu2j8DsVyT1azpJC_N
           G84Ty5KKthuCaPod7iI7w0LK9orSMhBEwwZDCxTWq4aY
           WAchc8t-emd9qOvWtVMDC2BXksRngh6X5bUYLy6AyHKv
           j-nUy1wgzjYQDwHMTplCoLtU-o-8SNnZ1tmRoGE9uJkB
           Ldh5gFENabWnU5m1ZqZPdwS-qo-meMvVfJb6jJVWRpl2
           SUtCnYG2C32qvbWbjZ_jBPD5eunqsIo1vQ",
      "e":"AQAB",
      "x5c": [
          "MIIDQjCCAiqgAwIBAgIGATz/FuLiMA0GCSqGSIb3DQEB
           BQUAMGIxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJDTzEP
           MA0GA1UEBxMGRGVudmVyMRwwGgYDVQQKExNQaW5nIElk
           ZW50aXR5IENvcnAuMRcwFQYDVQQDEw5CcmlhbiBDYW1w
           YmVsbDAeFw0xMzAyMjEyMzI5MTVaFw0xODA4MTQyMjI5
           MTVaMGIxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJDTzEP
           MA0GA1UEBxMGRGVudmVyMRwwGgYDVQQKExNQaW5nIElk
           ZW50aXR5IENvcnAuMRcwFQYDVQQDEw5CcmlhbiBDYW1w
           YmVsbDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC
           ggEBAL64zn8/QnHYMeZ0LncoXaEde1fiLm1jHjmQsF/4
           49IYALM9if6amFtPDy2yvz3YlRij66s5gyLCyO7ANuVR
           Jx1NbgizcAblIgjtdf/u3WG7K+IiZhtELto/A7Fck9Ws
           6SQvzRvOE8uSirYbgmj6He4iO8NCyvaK0jIQRMMGQwsU
           1quGmFgHIXPLfnpnfajr1rVTAwtgV5LEZ4Iel+W1GC8u
           gMhyr4/p1MtcIM42EA8BzE6ZQqC7VPqPvEjZ2dbZkaBh
```

```
            PbiZAS3YeYBRDWm1p1OZtWamT3cEvqqPpnjL1XyW+oyV
            VkaZdklLQp2Btgt9qr21m42f4wTw+Xrp6rCKNb0CAwEA
            ATANBgkqhkiG9w0BAQUFAAOCAQEAh8zGlfSlcI0o3rYD
            PBB07aXNswb4ECNIKG0CETTUxmXl9KUL+9gGlqCz5iWL
            OgWsnrcKcY0vXPG9J1r9AqBNTqNgHq2G03X09266X5Cp
            Oe1zFo+Owb1zxtp3PehFdfQJ610CDLEaS9V9Rqp17hCy
            ybEpOGVwe8fnk+fbEL2Bo3UPGrpsHzUoaGpDftmWssZk
            hpBJKVMJyf/RuP2SmmaIzmnw9JiSlYhzo4tpzd5rFXhj
            Rbg4zW9C+2qok+2+qDM1iJ684gPHMIY8aLWrdgQTxkum
            GmTqgawR+N5MDtdPTEQ0XfIBc2cJEUyMTY5MPvACWpkA
            6SdS4xSvdXK3IVfOWA=="
         ]
      }
   }
   "dac_effective_mask" : "ALL_PERMS"
   "client_headers" : {
      <CDMI-DAC- headers to client>
   },
}
```

Since the operation is allowed by the DAC provider, the following response is sent:

```
HTTP/1.1 200 OK
Content-Type: application/cms
Content-Length: 1425
CDMI-DAC-N: <client-specific headers including decryption key>

<CMS Encrypted Object>
```

24.8.3 GET plaintext of encrypted object with delegated access control

The following CDMI operation is performed against an encrypted CDMI object with DAC metadata:

```
GET /MyContainer/MyEncryptedObject.txt HTTP/1.1
Host: cloud.example.com
Accept: */*
```

The CDMI server verifies local access controls and determines that the request can proceed. The following DAC request is generated:

```
{
   "dac_request_version" : "1",
   "dac_request_id" : "F55AA0B6-8F54-4A03-AC21-87052D58485A",
   "server_identity" : {
      "kty":"RSA",
      "use":"sig",
      "kid":"1b94c",
      "n":"vrjOfz9Ccdgx5nQudyhdoR17V-IubWMeOZCwX_jj0hgA
           sz2J_pqYW08PLbK_PdiVGKPrqzmDIsLI7sA25VEnHU1u
           CLNwBuUiCO11_-7dYbsr4iJmG0Qu2j8DsVyT1azpJC_N
           G84Ty5KKthuCaPod7iI7w0LK9orSMhBEwwZDCxTWq4aY
           WAchc8t-emd9qOvWtVMDC2BXksRngh6X5bUYLy6AyHKv
           j-nUy1wgzjYQDwHMTplCoLtU-o-8SNnZ1tmRoGE9uJkB
           Ldh5gFENabWnU5m1ZqZPdwS-qo-meMvVfJb6jJVWRpl2
           SUtCnYG2C32qvbWbjZ_jBPD5eunqsIo1vQ",
      "e":"AQAB",
```

```
      "x5c": [
          "MIIDQjCCAiqgAwIBAgIGATz/FuLiMA0GCSqGSIb3DQEB
           BQUAMGIxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJDTzEP
           MA0GA1UEBxMGRGVudmVyMRwwGgYDVQQKExNQaW5nIElk
           ZW50aXR5IENvcnAuMRcwFQYDVQQDEw5CcmlhbiBDYW1w
           YmVsbDAeFw0xMzAyMjEyMzI5MTVaFw0xODA4MTQyMjI5
           MTVaMGIxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJDTzEP
           MA0GA1UEBxMGRGVudmVyMRwwGgYDVQQKExNQaW5nIElk
           ZW50aXR5IENvcnAuMRcwFQYDVQQDEw5CcmlhbiBDYW1w
           YmVsbDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC
           ggEBAL64zn8/QnHYMeZ0LncoXaEde1fiLm1jHjmQsF/4
           49IYALM9if6amFtPDy2yvz3YlRij66s5gyLCyO7ANuVR
           Jx1NbgizcAblIgjtdf/u3WG7K+IiZhtELto/A7Fck9Ws
           6SQvzRvOE8uSirYbgmj6He4iO8NCyvaK0jIQRMMGQwsU
           1quGmFgHIXPLfnpnfajr1rVTAwtgV5LEZ4Iel+W1GC8u
           gMhyr4/p1MtcIM42EA8BzE6ZQqC7VPqPvEjZ2dbZkaBh
           PbiZAS3YeYBRDWm1p1OZtWamT3cEvqqPpnjL1XyW+oyV
           VkaZdklLQp2Btgt9qr21m42f4wTw+Xrp6rCKNb0CAwEA
           ATANBgkqhkiG9w0BAQUFAAOCAQEAh8zGlfSlcI0o3rYD
           PBB07aXNswb4ECNIKG0CETTUxmXl9KUL+9gGlqCz5iWL
           OgWsnrcKcY0vXPG9J1r9AqBNTqNgHq2G03X09266X5Cp
           Oe1zFo+Owb1zxtp3PehFdfQJ610CDLEaS9V9Rqp17hCy
           ybEpOGVwe8fnk+fbEL2Bo3UPGrpsHzUoaGpDftmWssZk
           hpBJKVMJyf/RuP2SmmaIzmnw9JiSlYhzo4tpzd5rFXhj
           Rbg4zW9C+2qok+2+qDM1iJ684gPHMIY8aLWrdgQTxkum
           GmTqgawR+N5MDtdPTEQ0XfIBc2cJEUyMTY5MPvACWpkA
           6SdS4xSvdXK3IVfOWA=="
        ]
      }
   }
   "client_identity" : {
      "acl_name" : <acl name of client>
      "acl_group" : [<acl groups of clients>]
   },
   "acl_effective_mask" : <acl mask of client>
   "cdmi_objectid" : <CDMI Object ID of object being accessed>,
   "cdmi_enc_keyID" : "testkey",
   "cdmi_event_type" : "cdmi_read",
   "dac_response_uri" : "https://cloud.example.com/dacr"
}
```

The DAC provider processes the request and returns the following DAC response:

```
{
   "dac_response_version" : "1",
   "dac_response_id" : "F55AA0B6-8F54-4A03-AC21-87052D58485A",
   "dac_identity" : {
      "kty":"RSA",
      "use":"sig",
      "kid":"1b94c",
      "n":"vrjOfz9Ccdgx5nQudyhdoR17V-IubWMeOZCwX_jj0hgA
           sz2J_pqYW08PLbK_PdiVGKPrqzmDIsLI7sA25VEnHU1u
           CLNwBuUiCO11_-7dYbsr4iJmG0Qu2j8DsVyT1azpJC_N
           G84Ty5KKthuCaPod7iI7w0LK9orSMhBEwwZDCxTWq4aY
           WAchc8t-emd9qOvWtVMDC2BXksRngh6X5bUYLy6AyHKv
           j-nUy1wgzjYQDwHMTplCoLtU-o-8SNnZ1tmRoGE9uJkB
           Ldh5gFENabWnU5m1ZqZPdwS-qo-meMvVfJb6jJVWRpl2
```

```
            SUtCnYG2C32qvbWbjZ_jBPD5eunqsIo1vQ",
        "e":"AQAB",
        "x5c": [
            "MIIDQjCCAiqgAwIBAgIGATz/FuLiMA0GCSqGSIb3DQEB
             BQUAMGIxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJDTzEP
             MA0GA1UEBxMGRGVudmVyMRwwGgYDVQQKExNQaW5nIElk
             ZW50aXR5IENvcnAuMRcwFQYDVQQDEw5CcmlhbiBDYW1w
             YmVsbDAeFw0xMzAyMjEyMzI5MTVaFw0xODA4MTQyMjI5
             MTVaMGIxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJDTzEP
             MA0GA1UEBxMGRGVudmVyMRwwGgYDVQQKExNQaW5nIElk
             ZW50aXR5IENvcnAuMRcwFQYDVQQDEw5CcmlhbiBDYW1w
             YmVsbDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC
             ggEBAL64zn8/QnHYMeZ0LncoXaEde1fiLm1jHjmQsF/4
             49IYALM9if6amFtPDy2yvz3YlRij66s5gyLCyO7ANuVR
             Jx1NbgizcAblIgjtdf/u3WG7K+IiZhtELto/A7Fck9Ws
             6SQvzRvOE8uSirYbgmj6He4iO8NCyvaK0jIQRMMGQwsU
             1quGmFgHIXPLfnpnfajr1rVTAwtgV5LEZ4Iel+W1GC8u
             gMhyr4/p1MtcIM42EA8BzE6ZQqC7VPqPvEjZ2dbZkaBh
             PbiZAS3YeYBRDWm1p1OZtWamT3cEvqqPpnjL1XyW+oyV
             VkaZdklLQp2Btgt9qr21m42f4wTw+Xrp6rCKNb0CAwEA
             ATANBgkqhkiG9w0BAQUFAAOCAQEAh8zGlfSlcI0o3rYD
             PBB07aXNswb4ECNIKG0CETTUxmXl9KUL+9gGlqCz5iWL
             OgWsnrcKcY0vXPG9J1r9AqBNTqNgHq2G03X09266X5Cp
             Oe1zFo+Owb1zxtp3PehFdfQJ610CDLEaS9V9Rqp17hCy
             ybEpOGVwe8fnk+fbEL2Bo3UPGrpsHzUoaGpDftmWssZk
             hpBJKVMJyf/RuP2SmmaIzmnw9JiSlYhzo4tpzd5rFXhj
             Rbg4zW9C+2qok+2+qDM1iJ684gPHMIY8aLWrdgQTxkum
             GmTqgawR+N5MDtdPTEQ0XfIBc2cJEUyMTY5MPvACWpkA
             6SdS4xSvdXK3IVfOWA=="
          ]
      }
   }
   "dac_effective_mask" : "ALL_PERMS",
   "dac_object_key" {
      "kty" : "oct",
      "alg" : "A128KW",
      "k" : "GawgguFyGrWKav7AX4VKUg"}
   },
   "dac_key_cache_expiry" : "2015-07-20T14:12:44.835294Z",
   "dac_response_cache_expiry" : "2015-07-20T14:12:44.835294Z"
}
```

Since the operation is allowed by the DAC provider and the key is provided, the object is decrypted by the CDMI server and the following response is sent:

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

<Decrypted contents of Encrypted Value>