Content Delivery Networks Interconnection
Internet-Draft
Intended status: Standards Track
Expires: 2 January 2026

P. Chaudhari
Disney
G. Goldstein
W. Power
Lumen Technologies
A. Warshavsky
Qwilt

1 July 2025

CDNI Client Access Control Metadata
draft-ietf-cdni-client-access-control-metadata-02

Abstract

This specification adds to the basic client access control metadata in RFC8006, providing content providers and upstream content delivery networks (uCDNs) extended capabilities in defining location and time window restrictions. Support is also provided to define required Transport Layer Security (TLS) certificates and encryption levels. The specification also defines configuration metadata for the Common Access Token (CAT), developed jointly by the Streaming Video Technology Alliance (SVTA) and Consumer Technology Association Web Application Video Ecosystem (CTA-WAVE).

Table of Contents

1. Introduction

The [RFC8006] LocationACL and TimeWindowACL objects provide basic capabilities to gate a client's access to content. This specification details alternatives to these objects (using LocationACLExtended and TimeWindowACLExtended), that allow for the configuration of a Hypertext Transfer Protocol (HTTP) response in cases where requests are denied. Additional objects allow for the specification of metadata for required TLS certificates, encryption levels, and authentication tokens leveraging the CTA-WAVE Common Access Token standard [CTA-5007-B] The specification also introduces

standardized names for HTTP2 and HTTP3 protocols.

2. Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. MI.LocationACLExtended

MI.LocationACLExtended is an alternative to the Content Delivery Network Interconnection (CDNI) standard MI.LocationACL object that defines the locations (footprints) a User Agent needs to be in, in order to be able to receive the associated content. MI.LocationACLExtended uses ACL rules of type MI.LocationRuleExtended, providing rules for handling denied requests.

This object conforms to the specification defined for the behavior of MI.LocationACL and the two are mutually exclusive. Note that MI.LocationACLExtended instances that deny access are handled as terminating objects (as defined in [I-D.goldstein-processing-stages-metadata]) in that processing is terminated upon execution.

Property: rules

* Description: List of allow/deny rules per user location.

* Type: Array of MI.LocationRuleExtended objects.

* Mandatory-to-Specify: Yes

The following is an example of MI.LocationACLExtended with "allow/deny" rules:

```
{
  "generic-metadata-type": "MI.LocationACLExtended",
  "generic-metadata-value": {
    "rules": [
      {
        "locations": [
          {
            "footprint-type": "ipv4cidr",
            "footprint-value": [
              "10.1.1.0/24"
            ]
          }
        ],
        "action": "allow",
        "comment": "Support team"
      },
      {
        "locations": [
          {
            "footprint-type": "asn",
            "footprint-value": [
              "as12345"
            ]
          }
        ],
        "action": "deny",
        "comment": "Viewers from Antarctica",
        "deny-response": {
          "response-status": "302",
          "headers": [
            {
              "name": "Location",
              "value": "https: //example.com"
            },
            {
              "name": "Content-Type",
              "value": "text/html"
            }
          ]
        }
      }
    ]
  }
}
```

Figure 1

3.1. MI.LocationRuleExtended

MI.LocationRuleExtended is a subobject of MI.LocationACLExtended that defines pairs of user locations and allow/deny actions.

Property: locations

* Description: An array of client footprints to match against. These footprints, defined by pairs of MI_footprinttype_ex and MI_footprintvalue_ex respectively, extend the CDNI MI_footprinttype and MI_footprintvalue On top of the four footprint types defined by the CDNI in [RFC8006] (ipv4cidr,ipv6cidr,asn,countrycode), three new types are added:(ipv4range, ipv6range, subdivisioncode)

* Type: Array of MI.Footprint objects

* Mandatory-to-Specify: Yes

Property: action

* Description: The action to take place upon a location match.

* Type: String, one of (allow | deny)

* Mandatory-to-Specify: No. The default is "deny".

Property: comment

* Description: An optional text comment for user readability and for incorporating in logging.

* Type: String

* Mandatory-to-Specify: No

Property: deny-response

* Description: The configuration of the entire response to the client in case of a "Deny" action.

* Type: MI.SyntheticResponse

* Mandatory-to-Specify: No. The default is { "response-status" : 403 }.

Property: match-all-locations

* Description: The ACL rule match will take place only if all locations In the rule are matched, e.g., both asn and subdivisioncode.

* Type: Boolean

* Mandatory-to-Specify: No. The default is "False".

4. MI.TimeWindowACLExtended

MI.TimeWindowACLExtended is an alternative to the CDNI standard MI.TimeWindow object for implementing time-based access restrictions. It uses ACL rules of type MI.TimeWindowRuleExtended to provide rules for handling denied requests.

This object conforms to the specification defined for the behavior of MI.TimeWindowACL and the two are mutually exclusive. Note that MI.TimeWindowACLExtended instances that deny access are handled as terminating objects [I-D.goldstein-processing-stages-metadata] in that processing is terminated upon execution.

Property: rules

* Description: List of time window allow deny rules.

* Type: An array of MI.TimeWindowRuleExtended objects.

* Mandatory-to-Specify: Yes

The following is an example of MI.TimeWindowACLExtended with "allow" rules:

```
{
  "generic-metadata-type": "MI.TimeWindowACLExtended",
  "generic-metadata-value": {
    "rules": [
      {
        "windows": [
          {
            "start": 1670976000,
            "end": 4294967295
          }
        ],
        "action": "allow",
        "comment": "episode 1 launch"
      }
    ]
  }
}
```

Figure 2

4.1. MI.TimeWindowRuleExtended

Property: windows

* Description: Array of time windows to which the rule applies.

* Type: Array of MI.TimeWindow objects, as defined in RFC8006], using time values expressed in seconds since the UNIX epoch (i.e., zero hours, zero minutes, zero seconds, on January 1, 1970) Coordinated Universal Time (UTC).

* Mandatory-to-Specify: Yes

Property: action

* Description: The action to take place upon a time window match.

* Type: String, one of (allow | deny)

* Mandatory-to-Specify: No. The default is "deny".

Property: comment

* Description: An OPTIONAL text comment for user readability and for incorporating in logging.

* Type: String

* Mandatory-to-Specify: No

Property: deny-response

* Description: The configuration of the entire response to the client in case of a "Deny" action.

* Type: MI.SyntheticResponse

* Mandatory-to-Specify: No. The default is { "response-status" : 403 }.

5. MI.CertificateMetadata

To allow for secure delivery of content, a downstream CDN (dCDN) MUST be configured to support Hypertext Transfer Protocol Secure (HTTPs). The MI.CertificateMetadata object is used to configure the dCDN’s HTTPs attributes, such as TLS certificate credentials, encryption levels, protocols, and ciphers.

Property: encryption-level

* Description: A reference to an MI.EncryptionLevelMetadata object that specifies the TLS protocols and ciphers to use.

* Type: MI.EncryptionLevelMetadata

* Mandatory-to-Specify: Yes

Property: delegated-credentials

* Description: A reference to the certificate’s delegated credentials to use when establishing a TLS session with the client.

* Type: MI.CertificateCredentialsMetadata

* Mandatory-to-Specify: Yes

Property: ocsp-enabled

* Description: When ocsp-enabled is set to "True", the dCDN will check the revocation status of the configured certificate and include that information with the response to the client. See [RFC6066], section 8

* Type: Boolean

* Mandatory-to-Specify: No. The default is "False".

Property: prefer-server-ciphers

* Description: When prefer-server-ciphers is set to "False" (the default), the dCDN will prefer to use cipher suites in the order presented by the client when negotiating the TLS handshake. When prefer-server-ciphers is set to "True", cipher suites will be selected in the order preferred by the dCDN server.

* Type: Boolean

* Mandatory-to-Specify: No. The default is "False".

The following is an example of MI.CertificateMetadata:

```
{
  "generic-metadata-type": "MI.CertificateMetadata",
  "generic-metadata-value": {
    "encryption-level": {
      "generic-metadata-type": "MI.EncryptionLevelMetadata",
      "generic-metadata-value": {
        "encryption-level-name": "modern",
        "protocols": [
          "TLSv1.2",
          "TLSv1.3"
        ],
        "ciphers": [
          "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
          "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
          "TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384"
        ]
      }
    },
    "delegated-credentials": {
      "generic-metadata-type": "MI.CertificateCredentialsMetadata",
      "generic-metadata-value": {
        "delegated-credentials-type": "MI.ConfDelegatedCredentials",
        "delegated-credentials-value": {
          "credentials-location-uri":
                              "https://acme.example.com/cert-123"
        }
      }
    },
    "ocsp-enabled": "false",
    "prefer-server-ciphers": "false"
  }
}
```

Figure 3

5.1. MI.EncryptionLevelMetadata

MI.EncryptionLevelMetadata is a subobject of MI.CertificateMetadata to support HTTPS content delivery. MI.EncryptionLevelMetadata specifies the protocols and ciphers to be used by the associated MI.CertificateMetadata object. Externalizing MI.EncryptionLevelMetadata from MI.CertificateMetadata allows security policy (TLS protocols and ciphers) to be defined once and referenced by many configurations.

Property: encryption-level-name

* Description: A descriptive name for the MI.EncryptionLevelMetadata object. This name is expected to be used by operators to reference the MI.EncryptionLevelMetadata configuration.

* Type: String

* Mandatory-to-Specify: Yes

Property: protocols

* Description: An array that lists the allowed protocols for the TLS session.

* Type: Array of enumerated values. Must be one of: "TLSv1.0", "TLSv1.1", "TLSv1.2" , "TLSv1.3", "SSLv3".

* Mandatory-to-Specify: Yes

Property: ciphers

* Description: An array that lists the allowed ciphers for the TLS session, using cipher suite names defined in [RFC5289] For example, TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 or TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384.

* Type: Array of strings

* Mandatory-to-Specify: Yes

The following is an example of MI.EncryptionLevelMetadata:

```
{
  "generic-metadata-type": "MI.EncryptionLevelMetadata",
  "generic-metadata-value": {
    "encryption-level-name": "modern-version-1.2",
    "protocols": [
      "TLSv1.2",
      "TLSv1.3"
    ],
    "ciphers": [
      "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384"
    ]
  }
}
```

Figure 4

5.2. MI.CertificateCredentialsMetadata

MI.CertificateCredentialsMetadata is a subobject of MI.CertificateMetadata and defines the credentials to use when establishing a TLS session between a dCDN and a client.

Note: This document does not define any DelegatedCredentials methods. Individual DelegatedCredentials methods are defined separately, e.g., MI.DelegatedCredentials and Acme-Delegations (see CDNI Metadata for Delegated Credentials [RFC9677] and Delegation Using the Automated Certificate Management Environment [RFC9538]).

Property: delegated-credentials-type

* Description: The DelegatedCredentials type (the CDNI Payload Type [RFC7736] of the GenericMetadata object contained in the delegated-credentials-value property).

* Type: String

* Mandatory-to-Specify: Yes

Property: delegated-credentials-value

* Description: An object conforming to the specification associated with the DelegatedCredentials type.

* Type: GenericMetadata object

* Mandatory-to-Specify: Yes

The following is an example of MI.CertificateCredentialsMetadata:

```
{
  "generic-metadata-type": "MI.CertificateCredentialsMetadata",
  "generic-metadata-value": {
    "delegated-credentials-type":
      <CDNI Payload Type of this DelegatedCredentials object>,
    "delegated-credentials-value": {
      <Properties of this DelegatedCredentials object>
    }
  }
}
```

Figure 5

6. MI.ClientAuthMetadata

The MI.ClientAuthMetadata object defines how a dCDN authenticates client requests.

Property: delivery-auth

* Description: Authentication method to use when granting access to a resource requested by a client.

* Type: MI.Auth object, as defined in [RFC8006] section 4.2.7

* Mandatory-to-Specify: Yes

Following is a simple example:

```
{
  "generic-metadata-type": "MI.ClientAuthMetadata",
  "generic-metadata-value": {
    "delivery-auth": {
      "generic-metadata-type": "MI.Auth",
      "generic-metadata-value": {
        "auth-type": <CDNI Payload Type of this Auth object>,
        "auth-value": {}
      }
    }
  }
}
```

Figure 6

7. MI.CATAuth

The MI.CATAuth object defines the configuration for a dCDN to authenticate client requests using the Common Access Token standard as documented in the CTA-WAVE Standard [CTA-5007-B] The MI.CATAuth metadata object is used in the auth-value property of the MI.Auth object, as defined in [RFC8006] section 4.2.7, and MAY be applied to to client requests by including it under the MI.ClientAuthMetadata.delivery-auth property.

Property: version

* Description: Specifies the version of the CAT token. dCDN must reject a token for an unsupported version number. A rejected token results in the client response being defined by the ‘verification-action’ property.

* Type: Unsigned Integer. A value greater than 0

* Mandatory-to-Specify: No. The default value is 1

Property: token-object-name

* Description: Specifies the name of the token object of type MI.CATTokenObject that captures the results of the token verification. The verification result is available for inspection in all processing stages [I-D.goldstein-processing-stages-metadata]

* Type: string

* Mandatory-to-Specify: No

Property: tokens

* Description: Specifies the location in the client request where one or more tokens are present and read for evaluation. When multiple tokens are present, the tokens are read and processed per section 4.4 of [CTA-5007-B]

* Type: Array of MI.CATTokenLocator objects

* Mandatory-to-Specify: Yes

Property: configuration

* Description: Specifies the configuration parameters needed to verify a token.

* Type: MI.CATTokenConfiguration object

* Mandatory-to-Specify: Yes

Property: verification-action

* Description: Specifies how an invalid token OR a valid but rejected token defines the response returned to the client.

* Type: MI.CATTokenVerificationAction object

* Mandatory-to-Specify: Yes

7.1. MI.CATTokenLocator

The MI.CATTokenLocator object defines the location to read one or more tokens from the client request.

Property: location

* Description: Specifies the location name to find one or more tokens in the client request. Valid location names are:

  - "authorization-header" for the Authorization request header, in a recognised and configured scheme format.

  - "http-header" for any other HTTP Header locations.

    o "path-style-parameter" per [RFC6570], section 3.2.7.

    o "form-style-parameter" per [RFC6570], section 3.2.8 and [RFC6570], section 3.2.9

    o "cookie" for HTTP Cookies

  - Type: String

  - Mandatory-to-Specify: Yes

Property: scheme

* Description: The scheme for the Authorization request header that contains the CAT token

* Type: String

* Mandatory-to-Specify: Yes, only if the location property is ‘Authorization-Header’. Otherwise, his property is ignored.

Property: name

* Description: The names of the url-path-style parameter, query parameter, HTTP Header Field or HTTP cookies that contain one or more CAT tokens.

* Type: String

* Mandatory-to-Specify: Yes, only if the location property is either ‘path-style-parameter’, ‘form-style-parameter’, ‘cookie’ or ‘http-header’ and not using the default "CTA-Common-Access-Token" header field name. Otherwise, his property is ignored.

7.2. MI.CATTokenConfiguration

The MI.TokenConfiguration object defines the configuration parameters for verifying a CAT token.

Property: integrity-algo

* Description: The hashing / signing algorithm used for integrity protection of the CAT token. The minimum set of values to support are "hs256" (HMAC 256/256) (kty number 5, [RFC 9053], Section 3.1), "ps256" (kty number -37, [RFC9053], section 3.1) and "es256" (kty number -7, [RFC9053], section 2.1)

* Type: String

* Mandatory-to-Specify: Yes

Property: integrity-algo-key

* Description: The value of the key used for integrity protection of the CAT token.

* Type: MI.SecretValue object that can either contain the hex version of the key in the clear (not recommended) or as a reference in an external key store. See the Protected Secrets Metadata standard [I-D.ietf-cdni-protected-secrets-metadata] for more details.

* Mandatory-to-Specify: Yes.

Property: encryption-algo

* Description: The algorithm used for encryption via the COSE standard (Ref. [RFC9052]) of the entire CAT token. The minimum value to support is "ecdh-ss+a128kw" (see [RFC9053], section 6.4.1).

* Type: String

* Mandatory-to-Specify: Yes - IF using COSE Encrypted CAT tokens.

Property: encryption-algo-key

* Description: The value of the key used for encryption of the CAT Token via the COSE standard.

* Type: MI.SecretValue object that can either contain the hex version of the key in the clear (not recommended) or as a reference in an external key store. See the Protected Secrets Metadata standard [I-D.ietf-cdni-protected-secrets-metadata] for more details.

* Mandatory-to-Specify: Yes - IF using COSE Encrypted CAT tokens.

7.3. MI.CATTokenVerificationAction

The MI.CATTokenVerificationAction object defines how a client response is created for an invalid OR missing OR valid but rejected token, potentially overriding any actions that may be defined by claims within the token.

Property: rejected-token-action

* Description: An enumeration of ways in which a client response is created. Valid values are:

  - "fail" for returning a standard HTTP 403 response.

  - "allow" for ignoring the result of token validation, as if the token was not present. This typically would result in serving a HTTP 200 for the requested resource.

  - "token-defined" for constructing the client response per the parameters defined within the token. For a valid but rejected CAT token, the ‘catif’ claim is used to construct the client response.

* Type: String

* Mandatory-to-Specify: Yes

Property: token-defined-response

* Description: Specifies how the token values are used to create the client response for an invalid OR valid but rejected token. This property gets used when the value of ‘rejected-token-action’ property is ‘token-defined’. Use of this property for any other value of ‘rejected-token-action’ is an invalid configuration.

* Type: MI.CATTokenDefinedResponse object

* Mandatory-to-Specify: Yes, if ‘rejected-token-action’ property is ‘token-defined’

7.4. MI.CATTokenDefinedResponse

The MI.CATTokenDefinedResponse object defines how a client response is constructed using the token’s available ‘catif’ claim or some fallback.

Property: fallback-response

* Description: An enumeration defining a fallback response when the token defined values are inapplicable. Valid values are:

  - "fail" for returning a HTTP 403 response.

  - "allow" for ignoring the result of token validation, as if it was not present.

    o A missing token.

    o An invalid CAT token.

    o A valid but rejected CAT token with:

For CAT, this fallback-response will get used under the following conditions:

* Type: String

* Mandatory-to-Specify: No. Default value is ‘fail’.

Property: catif

* Description: Specifies the configuration for processing the ‘catif’ claim to construct the client response for a valid but rejected CAT token.

* Type: MI.CATIF object

* Mandatory-to-Specify: Yes, when the rejected-token-action property for the MI.CATTokenVerificationAction object is set to ‘token-defined’

7.5. MI.CATIF

The MI.CATIF object specifies the processing of the ‘catif’ claim to construct the client response for a valid but rejected CAT token. See Section 4.9.1 of [CTA-5007-B]

Property: response-header-names-force-add

* Description: For specific named headers of a rejected claim within the ‘catif’ claim, these will be added to the client response, even if headers with same names were already going to be added to the response via other means.

* Type: Array of Strings

* Mandatory-to-Specify: No. If not specified, header names of a rejected claim within ‘catif’ claim will be skipped in the client response if similarly named headers were to be added via other means.

7.6. MI.CATTokenObject

The MI.CATTokenObject object defines a read-only object that captures the results of the token verification. The verification result is available for inspection in all processing stages [I-D.goldstein-processing-stages-metadata]

Property: status

* Description: Captures the verification status of the CAT token via the following string values:

  - ‘success’

  - ‘failure’

* Type: String

Property: token

* Description: The string literal of the token in the request.

* Type: String

Property: status_details

* Description: Specifies details about the token verification result when the token verification ‘status’ is ‘failure’, else it is an empty string. This string contains one of the following values:

  - Empty string for a missing token.

  - "Invalid CAT" when a token cannot be parsed, per [RFC8392], Section 7.2

  - "Invalid Claim: <name of the first claim that caused the CAT token to be rejected>"

* Type: String

The following example shows use of the CAT scheme as the authentication mechanism for client requests at the dCDN. Some notable points are:

```
[
  {
    "generic-metadata-type": "MI.CATAuth",
    "generic-metadata-value": {
      "version": "1",
      "token-object-name" : "CATAuthResult",
      "tokens": [
        {
          "location": "form-style-parameter",
          "name": "catv1"
        },
        {
          "location": "authorization-header",
          "scheme": "bearer"
        },
        {
          "location": "path-style-parameter",
          "name": "catv1"
        },
        {
           "location": "cookie",
           "name": "catv1"
        },
      ],
      "configuration": {
        "integrity-algo": "hs256",
```

```
          "integrity-algo-key": {
            "secret-store-id": "cat-key-vault",
            "secret-path": "/client-auth/cat/keys/integrity/examplevideoprovider"
          },
          "encryption-algo": "ecdh-ss+a128kw",
          "encryption-algo-key": {
            "secret-store-id": "cat-key-vault",
            "secret-path": "/client-auth/cat/keys/enc/examplevideoprovider"
          }
        },
        "verification-action": {
          "rejected-token-action": "token-defined",
          "token-defined-response": {
            "fallback-response": "fail",
            "catif": {
              "response-header-names-force-add": [
                "x-cat-reason",
                "x-uCDN-name"
              ]
            }
          }
        },
      },
    }

    {
      "generic-metadata-type": "MI.ProcessingStages",
      "generic-metadata-value": {
        "client-response": [
          {
            "match": "var.CATAuthResult and var.CATAuthResult.status == 'failure'",
            "stage-metadata": {
              "response-transform": {
                "header-transform": {
                  "add": [
                    {
                      "name": "x-cat-auth-debug",
                      "value": "'Invalid token' . var.CATAuthResult.token . 'failed veri
fication. Error:' . var.CATAuthResult.status_details",
                      "value-is-expression": true
                    }
                  ]
                }
              }
            }
          }
        ]
      }
    }
```

]

Figure 7

8. Security Considerations

The FCI and MI objects defined in the this document are transferred via the interfaces defined in CDNI [RFC8006] which describes how to secure these interfaces by protecting integrity and confidentiality while ensuring the authenticity of the dCDN and uCDN.

9. Iana Considerations

9.1. CDNI Payload Types

This document requests the registration of the following entries under the "CDNI Payload Types" registry hosted by IANA:

Payload Type	Specification
MI.LocationACLExtended	RFCthis
MI.LocationRuleExtended	RFCthis
MI.TimeWindowACLExtended	RFCthis
MI.TimeWindowRuleExtended	RFCthis
MI.CertificateMetadata	RFCthis
MI.EncryptionLevelMetadata	RFCthis
MI.CertificateCredentialsMetadata	RFCthis
MI.ClientAuthMetadata	RFCthis
MI.CATAuth	RFCthis
MI.CATTokenLocator	RFCthis
MI.CATTokenConfiguration	RFCthis
MI.CATTokenVerificationAction	RFCthis
MI.CATTokenDefinedResponse	RFCthis
MI.CATIF	RFCthis
MI.CATTokenObject	RFCthis

Table 1

9.2. "CDNI Metadata Protocol Types" Registry

The Internet Assigned Numbers Authority (IANA) "CDNI Metadata Protocol Types" registry in the "Content Delivery Network Interconnection Parameters" registry group defines the valid Protocol object values used by the ProtocolACL object defined in [RFC8006]

The following table defines the new protocol values needed for the ProtocolACL object defined in [RFC8006] such that CDN delivery restrictions can be configured for these protocols.

Protocol Type	Description	Type Specification	Protocol Specification
http/2	Hypertext Transfer Protocol Version 2 (unencrypted)	RFCthis	[RFC9113]
https/2	Hypertext Transfer Protocol Version 2 (encrypted)	RFCthis	[RFC9113]
h2	Hypertext Transfer Protocol Version 2, alternate name	RFCthis	[RFC9113]
https/3	Hypertext Transfer Protocol Version 3	RFCthis	[RFC9114]
h3	Hypertext Transfer Protocol Version 3, alternate name	RFCthis	[RFC9114]

Table 2

10. Acknowledgements

The authors would like to express their gratitude to the members of the Streaming Video Technology Alliance [SVTA] Open Caching Working Group for their contributions and guidance.

Particulary the following people contribute in one or other way to the content of this draft:

* Guillaume Bichot - Broadpeak

* Christoph Neumann - Broadpeak

* Chris Lemmons - Comcast

* Rajeev RK - picoNETS

* Shmuel Asafi - Qwilt

* Yoav Gressel - Qwilt

* Nir Sopher - Qwilt

* Alfonso Siloniz - Telefonica

* Ben Rosenblum - Vecima

11. Normative References

[CTA-5007-B]
CTA, "Web Application Video Ecosystem - Common Access Token (CTA-5007-B)", <https://shop.cta.tech/collections/standards/products/cta-5007-B>.

[I-D.ietf-cdni-protected-secrets-metadata]
Rosenblum, B. and G. Goldstein, "CDNI Protected Secrets Metadata", Work in Progress, Internet-Draft, draft-ietf-cdni-protected-secrets-metadata-04, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-cdni-protected-secrets-metadata-04>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5289] Rescorla, E., "TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)", RFC 5289, DOI 10.17487/RFC5289, August 2008, <https://www.rfc-editor.org/info/rfc5289>.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <https://www.rfc-editor.org/info/rfc6066>.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, DOI 10.17487/RFC6570, March 2012, <https://www.rfc-editor.org/info/rfc6570>.

[RFC7736] Ma, K., "Content Delivery Network Interconnection (CDNI) Media Type Registration", RFC 7736, DOI 10.17487/RFC7736, December 2015, <https://www.rfc-editor.org/info/rfc7736>.

[RFC8006] Niven-Jenkins, B., Murray, R., Caulfield, M., and K. Ma, "Content Delivery Network Interconnection (CDNI) Metadata", RFC 8006, DOI 10.17487/RFC8006, December 2016, <https://www.rfc-editor.org/info/rfc8006>.

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <https://www.rfc-editor.org/info/rfc8392>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/info/rfc9052>.

[RFC9053] Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", RFC 9053, DOI 10.17487/RFC9053, August 2022, <https://www.rfc-editor.org/info/rfc9053>.

[RFC9113] Thomson, M., Ed. and C. Benfield, Ed., "HTTP/2", RFC 9113, DOI 10.17487/RFC9113, June 2022, <https://www.rfc-editor.org/info/rfc9113>.

[RFC9114] Bishop, M., Ed., "HTTP/3", RFC 9114, DOI 10.17487/RFC9114, June 2022, <https://www.rfc-editor.org/info/rfc9114>.

12. Informative References

[I-D.goldstein-processing-stages-metadata] Goldstein, G., Power, W., and A. Warshavsky, "CDNI Processing Stages Metadata", Work in Progress, Internet-Draft, draft-goldstein-processing-stages-metadata-03, 25 February 2025, <https://datatracker.ietf.org/doc/html/draft-goldstein-processing-stages-metadata-03>.

[RFC9538] Fieau, F., Ed., Stephan, E., and S. Mishra, "Content Delivery Network Interconnection (CDNI) Delegation Using the Automated Certificate Management Environment", RFC 9538, DOI 10.17487/RFC9538, February 2024, <https://www.rfc-editor.org/info/rfc9538>.

[RFC9677] Fieau, F., Stephan, E., Bichot, G., and C. Neumann, "Content Delivery Network Interconnection (CDNI) Metadata for Delegated Credentials", RFC 9677, DOI 10.17487/RFC9677, October 2024, <https://www.rfc-editor.org/info/rfc9677>.

[SVTA] SVTA, "Streaming Video Technology Alliance Home Page", <https://www.svta.org>.

Authors' Addresses

Pankaj Chaudhari
Disney
United States of America
Email: pankaj.chaudhari.pub@gmail.com

Glenn Goldstein
Lumen Technologies
United States of America
Email: glenng1215@gmail.com

Will Power
Lumen Technologies
United States of America
Email: wrpower@gmail.com

Arnon Warshavsky
Qwilt
Israel
Email: arnon@qwilt.com