Content Delivery Networks Interconnection
Internet-Draft
Intended status: Standards Track
Expires: 19 October 2026

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

17 April 2026

CDNI Source Access Control Metadata
draft-ietf-cdni-source-access-control-metadata-01

Abstract

This specification provides an alternative to the MI.SourceMetadata objects defined in RFC8006, providing greatly extended capabilities with regards to defining multiple sources, load balancing, and failover rules across those sources, as well as a mechanism for a content delivery network (CDN) to monitor source health and pull unhealthy sources out of rotation. Additionally, new methods are defined for authentication access to an upstream source/origin.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 19 October 2026.

Copyright Notice

Table of Contents

1. Introduction

The [RFC8006] MI.SourceMetadata and Source objects provide the dCDN with information about content acquisition, i.e., how to contact an upstream content delivery network (uCDN) or an origin server to obtain the content to be served. This specification details alternatives to these objects (using MI.SourceMetadataExtended and MI.SourceExtended), that provide a rich set of additional capabilities for managing the connection and access to upstream sources, such as:

* Designation of the required protocol for source access..
* Specification of the source’s Transmission Control Protocol (TCP) port number.
* Web root path specification for the source.
* Indication as to whether redirects should be followed.
* Support for additional forms of origin authentication.
* Multi-origin failover - The ability to specify a list of origins that can act as fallbacks to the primary origin in the event of failures connecting to an origin Failure rules can specify types of errors and timeout values that trigger failover.
* Multi-origin load balancing - The ability to specify a list of origins that can be selected by one of several balancing rules (random, content hash, or IP hash).
* Specification of connection control parameters for origin access, with detailed options for specifying error handling, timeouts, and retries.
* Mechanisms to track the health of upstream sources and place unhealthy sources in detention (an unusable state) until they return to good health.

2. Requirements

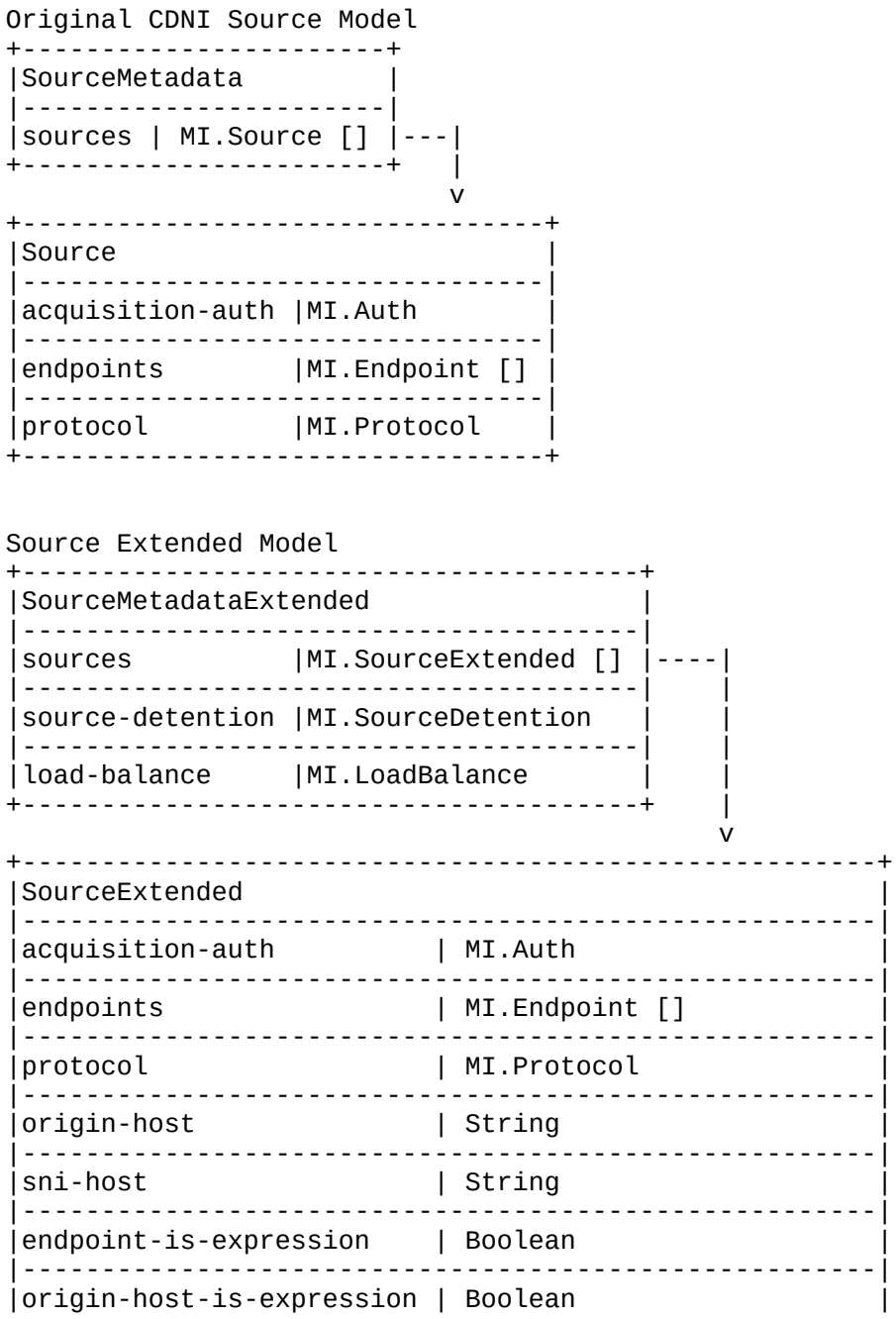
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. MI.SourceMetadataExtended

MI.SourceMetadataExtended is an alternative to the CDN Interconnection (CDNI) standard MI.SourceMetadata object, which adds a property to specify load balancing across multiple sources, a MI.SourceExtended subobject with additional attributes to the CDNI standard Source object and a MI.SourceDetention subobject to manage unavailable sources.

While both MI.SourceMetadataExtended and MI.SourceMetadata can be provided for backward compatibility, a dCDN that advertises capability for MI.SourceMetadataExtended MUST ignore MI.SourceMetadata if both are provided for a given host or path match.

The following diagram shows MI.SourceMetadataExtended and its subobjects:



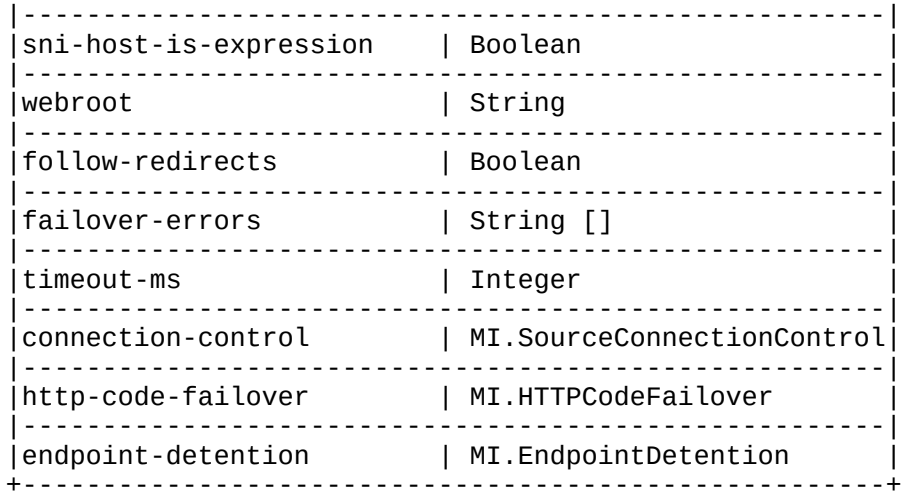


Figure 1: MI.SourceMetadataExtended Object Model

Property: sources

* Description: The sources from which the dCDN can acquire content, listed in order of preference.

* Type: Array of MI.SourceExtended objects

* Mandatory-to-Specify: No. The default is to use a static configuration, out-of-band from the CDNI metadata interface.

Property: source-detention

* Description: Specifies rules for handling detention for a set of sources.

* Type: SourceDetention object

* Mandatory-to-Specify: No. If not specified, the dCDN-specific behavior gets used.

Property: load-balance

* Description: Specifies load balancing rules for the set of sources.

* Type: LoadBalanceMetadata object

* Mandatory-to-Specify: No

The following is an example of an MI.SourceMetadataExtended object with the associated MI.LoadBalanceMetadata configuration object:

```
{
  "generic-metadata-type": "MI.SourceMetadataExtended",
  "generic-metadata-value": {
    "sources": [
      {
        "endpoints": [
          "a.service123.ucdn.example",
          "b.service123.ucdn.example"
        ],
        "protocol": "http/1.1"
      },
      {
        "endpoints": [
          "origin.service123.example"
        ],
        "protocol": "http/1.1"
      }
    ],
    "load-balance": {
      "balance-algorithm": "content-hash",
      "balance-path-pattern": "^/prod/(.*)/.*\\.ts$"
    }
  }
}
```

Figure 2

3.1. MI.SourceExtended

MI.SourceExtended is an alternative to the CDNI standard Source object with additional metadata. It contains all the attributes of the [RFC8006] Source object (acquisition-auth, endpoints, and protocol), with additions specified below.

Property: acquisition-auth

* Description: The authentication method to use when requesting content from this source. Same as [RFC8006]

* Type: Auth (see [RFC8006] Section 4.2.7 and the MI.Auth types in this specification)

* Mandatory-to-Specify: No. The default is no authentication required.

Property: endpoints

* Description: The origins from which the dCDN can acquire content. If multiple endpoints are specified, they are all equal, i.e., the list is not ordered by preference. Same as [RFC8006]

* Type: Array of endpoint objects (see [RFC8006] Section 4.3.3)

* Mandatory-to-Specify: Yes

Property: protocol

* Description: The network retrieval protocol to use when requesting content from this source. Same as [RFC8006]

* Type: Protocol (see [RFC8006] Section 4.3.2)

* Mandatory-to-Specify: Yes

Property: origin-host

* Description: The Hypertext Transfer Protocol (HTTP) host header to pass to the endpoints when retrieving content from a uCDN. The host MUST conform to the Domain Name System (DNS) syntax defined in [RFC1034] and [RFC1123]

* Type: String

* Mandatory-to-Specify: No. The default is to use the host name passed by the dCDN.

Property: sni-host

* The Server Name Indication (SNI) to be used during the TLS handshake when establishing a secure connection with an endpoint.

* Type: String

* Mandatory-to-Specify: No. The default is to use the value set for origin-host, and if this field is not set, use the value of the endpoint.

Property: endpoint-is-expression

* Description: If the endpoint-is-expression property is set to "True", the value of the endpoint will be evaluated as a Metadata Expression Language (MEL) expression. See [I-D.ietf-cdni-metadata-expression-language]

* Type: Boolean

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

Property: origin-host-is-expression

* Description: If the origin-host-is-expression property is set to "True", the value of the origin-host property will be evaluated as a Metadata Expression Language (MEL) expression. See [I-D.ietf-cdni-metadata-expression-language]

* Type: Boolean

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

Property: sni-host-is-expression

* Description: If the sni-host-is-expression property is set to "True", the value of the sni-host property will be evaluated as a Metadata Expression Language (MEL) expression. See [I-D.ietf-cdni-metadata-expression-language]

* Type: Boolean

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

Property: webroot

* Description: The path element that is prepended to a resource’s Uniform Resource Identifier (URI) before retrieving content from a uCDN.

* Type: String that MUST starti with a "/"

* Mandatory-to-Specify: No. The default is to use the original URI.

Property: follow-redirects

* Description: If the follow-redirects property is set to "True", HTTP redirect responses returned from a uCDN will be followed when retrieving content. Otherwise, the HTTP redirect response is returned to the client.

* Type: Boolean

* Mandatory-to-Specify: No. The default is "True" (i.e., follow redirect responses from the uCDN).

Property: failover-errors

* Description: An array of HTTP response error status codes (see Sections 15.5 and 15.6 of [RFC9110]), that if returned from the uCDN, will trigger a connection attempt to the next source in the MI.SourceMetadataExtended source array. If the uCDN returns an HTTP error code that is not in the failover-errors array, that error code is returned to the client of the dCDN. Note that use of the http-code-failover property overrides this setting.

* Type: Array of HTTP response status codes encoded as strings. Any HTTP status code from 100 to 599, or any of the special values, "2xx", "3xx", "4xx" or "5xx", where "xx" implies everything from 00 to 99. While repeated and redundant values in the array are allowed, they SHOULD be avoided for efficiency (no reason to specify both 5xx and 503, for example).

* Mandatory-to-Specify: No. The default is to revert to [RFC8006] behavior.

Property: timeout-ms

* Description: A timeout (in milliseconds), to apply when connecting to a uCDN. If the connection is not established within timeout-ms, this source is abandoned and the next source in the MI.SourceMetadataExtended source array is tried. Once a connection is established, timeout-ms is used on subsequent reads of data from the uCDN. Note that use of the connection-control property overrides this setting.

* Type: Integer

* Mandatory-to-Specify: No. The default is to revert to [RFC8006] behavior if neither this property or connection-control are specified.

Property: connection-control

* Description: Specifies how connection-related errors and timeouts are handled. When specified, it overrides the timeout-ms property.

* Type: SourceConnectionControl object

* Mandatory-to-Specify: No. If not specified, the timeout-ms property is used to configure simple connection control timeouts.

Property: http-code-failover

* Description: Specifies how HTTP response error codes are handled. When specified, it overrides the failover-errors property.

* Type: HTTPCodeFailover object

* Mandatory-to-Specify: No. If not specified, the failover-errors property is used to configure error handling behavior for HTTP response error codes.

Property: endpoint-detention

* Description: Defines error-based triggers for which an endpoint is put in detention (excluded from future use) for a given duration of time.

* Type: EndpointDetention object

* Mandatory-to-Specify: No. If not specified, no health check and detention is done on an unhealthy endpoint.

The following is an example of an MI.SourceMetadataExtended object describing a pair of sources with a fail-over relationship.. The 2nd source dynamically constructs the origin endpoint based on the client request host header:

```
   {
     "generic-metadata-type": "MI.SourceMetadataExtended",
     "generic-metadata-value": {
       "sources": [
         {
           "endpoints": [
             "a.service123.ucdn.example",
             "b.service123.ucdn.example:8443"
           ],
           "protocol": "https/1.1",
           "origin-host": "internal.example.com",
           "webroot": "/prod",
           "follow-redirects": false,
           "timeout-ms": 4000,
           "failover-errors": [ "502", "503", "504" ]
         },
         {
           "endpoints": [ "match_replace(req.h.host,' (^[^\.]+).*' ,' $1' ) . ' .example.
com' " ],
           "endpoint-is-expression": true,
           "protocol": "http/1.1",
           "sni-host": "acme-host.com",
           "origin-host": "origin.service123.example",
           "webroot": "/prod",
           "follow-redirects": true,
           "timeout-ms": 8000
         }
       ]
     }
   }
```

Figure 3

3.1.1. MI.SourceConnectionControl

MI.SourceConnectionControl is a subobject of MI.SourceExtended and allows for the dCDN to specify how it will handle connection timeouts and errors against an origin/uCDN. The use of this object overrides the timeout-ms property of MI.SourceExtended, even if any timeout properties of this object are not configured.

The following diagram illustrates the model for a CDN source load balancer failing over between multiple sources based on different types of timeout errors, and calls out differences from the Configuration Interface version 1.1 behavior.

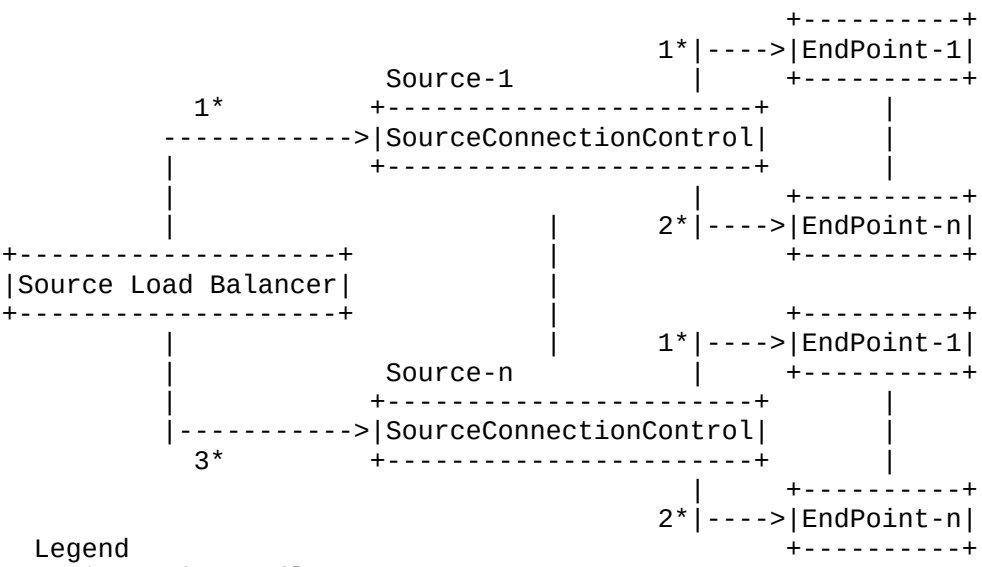


Figure 4: Failover Model for timeouts

Property: connection-setup-timeout-ms

* Description: The time (in milliseconds) within which a connection MUST be established against an endpoint of an MI.SourceExtended object.

* Type: Integer. A value greater than 0 to define a timeout (in milliseconds) for establishing a new network connection.

* Mandatory-to-Specify: No. If not specified, internal dCDN values are used.

Property: connection-setup-timeout-ms-actions

* Description: Specifies error handling actions to take when the connection could not be established within a given timeout, as specified by the connection-setup-timeout-ms property. It is an error to specify this property without the corresponding connection-setup-timeout-ms property.

* Type: SourceTimeoutActions object

* Mandatory-to-Specify: No. If not specified, and a timeout occurs when establishing a new network connection, the next MI.SourceExtended will be used as a failover action.

Property: first-byte-read-timeout-ms

* Description: The time, in milliseconds, within which the first byte MUST be received/read on a new network connection.

* Type: Integer. A value greater than 0 to define a timeout, in milliseconds, for reading the first byte on a new network connection.

* Mandatory-to-Specify: No. If not specified, internal dCDN values are used.

Property: first-byte-read-timeout-ms-actions

* Description: Specifies the error handling actions to take when the first byte on a new connection could not be read within a given timeout, as specified by the first-byte-read-timeout-ms property. It is an error to specify this property without the corresponding first-byte-read-timeout-ms property.

* Type: SourceTimeoutActions object

* Mandatory-to-Specify: No. If not specified, and a timeout occurs when reading the first byte on a new network connection, the next MI.SourceExtended will be used as a failover action.

Property: byte-read-timeout-ms

* Description: The time (in milliseconds) within which the next (inter) byte MUST be received/read on a connection.

* Type: Integer. A value greater than 0 to define a timeout, in milliseconds, for reading the next byte on an established network connection.

* Mandatory-to-Specify: No. If not specified, internal dCDN values are used.

Property: byte-read-timeout-ms-actions

* Description: Specifies error handling actions to take when the next byte on a connection could not be read within a given timeout, as specified by the byte-read-timeout-ms property. It is an error to specify this property without the corresponding byte-read-timeout-ms property.

* Type: SourceByteReadTimeoutActions object

* Mandatory-to-Specify: No. If not specified, and a timeout occurs when reading bytes from an existing network connection, the next MI.SourceExtended will be used as a failover action.

Property: connection-keep-alive-time-ms

* Description: Specifies the time, in milliseconds, to keep an idle connection open.

* Type: Integer. A value greater than 0 defines the amount of time for which a connection SHOULD be kept alive.

* Mandatory-to-Specify: No. If unspecified, dCDN defined internal values are used.

Property: max-connection-retries-per-source

* Description: Specifies the upper bound of retries allowed across all endpoints of an MI.SourceExtended object for all timeout errors.

* Type: Integer. A value greater than or equal to 0 indicates the maximum number of retries against all endpoints of the current MI.SourceExtended object for connection establishment, first-byte-read, and byte-read timeout errors. A value of 0 indicates that no retries are to be performed against this source.

* Mandatory-to-Specify: No. If not specified, any retry values defined in the properties of connection-setup-timeout-ms-actions, first-byte-read-timeout-ms-actions, and byte-read-timeout-ms-actions for their respective error types are used.

Property: resume-from-last-byte-of-previous-source

* Description: Upon establishing a new connection to an origin/uCDN, this specifies whether the next byte to read will start from the last byte read from a previous MI.SourceExtended object.

* Type: Boolean. If "True", the next byte read will resume from the last successful byte downloaded from a previous MI.SourceExtended object. The default is "False".

* Mandatory-to-Specify: No.

Property: resume-from-last-byte-of-previous-endpoint

* Description: Upon establishing a new connection to an origin/uCDN, this specifies whether the next byte read will start from the last byte read from a previous endpoint of the current MI.SourceExtended.

* Type: Boolean. If "True", the next byte read will resume from the last successful byte downloaded from a previous endpoint of the current MI.SourceExtended object. The default is "False".

* Mandatory-to-Specify: No.

3.1.1.1. MI.SourceByteReadTimeoutActions

MI.SourceByteReadTimeoutActions is a subobject of MI.SourceConnectionControl and allows for the specification of actions to be taken when a timeout occurs when reading the next byte on an existing network connection against an endpoint of an origin/uCDN.

Property: retries

* Description: Specifies the number of retries that MUST occur against a given endpoint when an error occurs.

* Type: SourceConnectionRetries object

* Mandatory-to-Specify: No. If not specified, no retries are done.

Property: error-state

* Description: Specifies whether to capture any error message for the current error condition.

* Type: SetVariable object specifying a user-defined variable. See the Metadata Expression Language Specification [I-D.ietf-cdni-metadata-expression-language]

* Mandatory-to-Specify: No. If not specified, a dCDN will not capture the error condition for which this object is used.

Property: resume-from-last-byte

* Description: Specifies whether to resume reading from the last byte of the current endpoint only if the content length of the object being downloaded is known.

* Type: Boolean. If set to "True", when a byte read timeout occurs against an endpoint and if a retry action is performed against the same endpoint, the byte read will resume from the last byte downloaded. The default value is "False".

* Mandatory-to-Specify: No. If not specified, a retry against the current endpoint will download the requested object from the first byte.

3.1.1.2. MI.SourceTimeoutActions

MI.SourceTimeoutActions is a subobject of MI.SourceConnectionControl and allows for the specification of actions to be taken when a timeout occurs against an endpoint of an origin/uCDN.

Property: retries

* Description: Specifies the number of retries that MUST occur against a given endpoint when an error occurs.

* Type: SourceConnectionRetries object

* Mandatory-to-Specify: No. If not specified, no retries are made.

Property: error-state

* Description: Specifies whether to capture any error message for the current error condition.

  - Type: SetVariable object specifying a user-defined variable. See the Metadata Expression Language Specification [I-D.ietf-cdni-metadata-expression-language]

* Mandatory-to-Specify: No. If not specified, a dCDN will not capture the error condition for which this object is used.

3.1.1.3. MI.SourceConnectionRetries

MI.SourceConnectionRetries is a subobject that allows for the specification of retries to be performed against endpoints of an MI.SourceExtended object.

Property: max-retries-per-source

* Description: Specifies the maximum number of retries against all endpoints of the MI.SourceExtended object for a given class of error.

* Type: Integer. A value of greater than or equal to 0 indicates the number of retries. Example: If the value of max-retries-per-source of the retries property of connection-setup-timeout-ms-actions is set to 5, a maximum of 5 retries are allowed for connection setup timeout errors across all endpoints of the current MI.SourceExtended object. If this value is set to 0, no retries will be done.

* Mandatory-to-Specify: No. If not specified, the individual endpoint-specific retries are honored.

Property: retries-per-endpoint

* Description: Specifies the number of retries against the current endpoint of the MI.SourceExtended object for a given class of error.

* Type: Integer. A value of greater than or equal to 0 indicates the number of retries. Example: If the value of retries-per-endpoint of the retries property of connection-setup-timeout-ms-actions is set to 5, 5 retries will be done for connection-setup timeout errors on the current endpoint of the current MI.SourceExtended object. If this value is set to 0, no retries will be done.

* Mandatory-to-Specify: No. If not specified, the value of 0 is assumed and no retries will be done for the given class of error.

Following is an example illustrating how different connection errors against an origin/uCDN can be handled with a custom response sent to the player/client. This example illustrates use of the Processing Stages Model [I-D.ietf-cdni-processing-stages-metadata] to apply metadata at the clientResponse Stage:

```
[
  {
    "generic-metadata-type": "MI.SourceConnectionControl",
    "generic-metadata-value": {
      "connection-setup-timeout-ms": 10,
      "connection-setup-timeout-ms-actions": {
        "retries": {
          "max-retries-per-source": 3,
```

```
            "retries-per-endpoint": 1
          },
          "error-state": {
            "variable-name": "connection-setup-timeout",
            "variable-value": "true"
          }
        },
        "first-byte-read-timeout-ms": 1,
        "first-byte-read-timeout-ms-actions": {
          "retries": {
            "max-retries-per-source": 3,
            "retries-per-endpoint": 1
          },
          "error-state": {
            "variable-name": "first-byte-read-timeout",
            "variable-value": "true"
          }
        },
        "byte-read-timeout-ms": 1,
        "byte-read-timeout-ms-actions": {
          "resume-from-last-byte": true,
          "retries": {
            "max-retries-per-source": 3,
            "retries-per-endpoint": 1
          },
          "error-state": {}
        },
        "connection-keep-alive-time-ms": 3,
        "max-connection-retries-per-source": 3,
        "resume-from-last-byte-of-previous-source": true,
        "resume-from-last-byte-of-previous-endpoint": true
      }
    },
    {
      "generic-metadata-type": "MI.ProcessingStages",
      "generic-metadata-value": {
        "client-response": [
          {
            "match": "var.connection-setup-timeout-ms == 'true'",
            "stage-metadata": {
              "response-transform": {
                "header-transform": {
                  "add": [
                    {
                      "name": "x-failure-handling",
                      "value": "connection setup timeout",
                      "value-is-expression": false
                    }
```

```
                  ]
                },
                "response-status": "404",
                "status-is-expression": false
              }
            }
          }
        ]
      }
    }
  ]
```

Figure 5

3.1.2. MI.HTTPCodeFailover

MI.HTTPCodeFailover is a GenericMetadata subobject of MI.SourceExtended that allows the dCDN to specify how it will handle HTTP error codes against an origin/uCDN.

The following diagram illustrates the model for a CDN source load balancer failing over between multiple sources based on HTTP error code status.

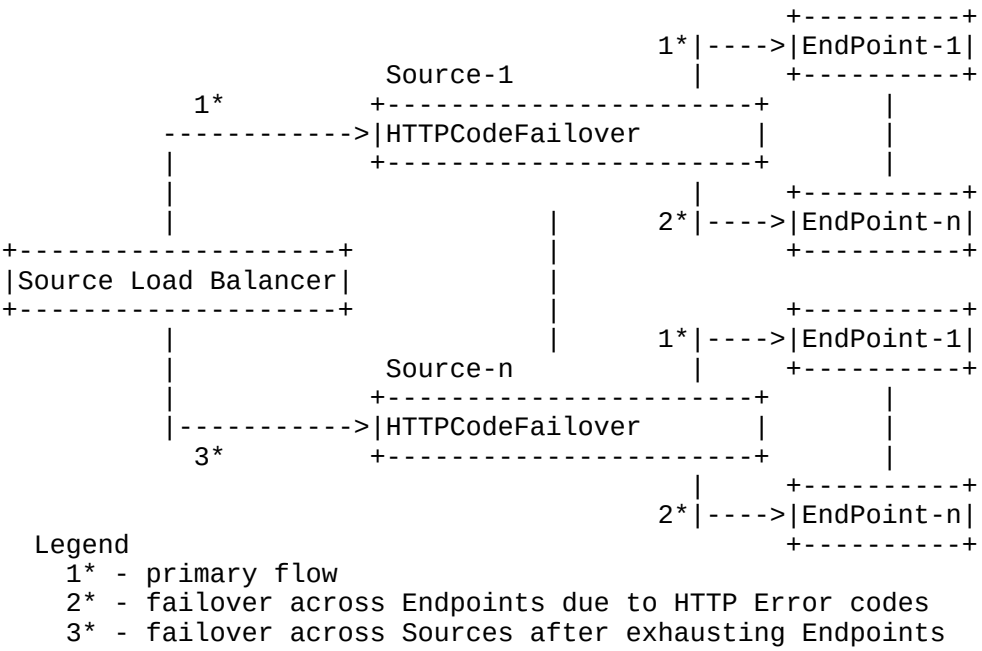


Figure 6: Failover Model for HTTP status codes

Property: max-reforwards-per-source

* Description: Specifies the upper bound of reforwards allowed across all endpoints of an MI.SourceExtended object for all configured HTTP error codes, where the term reforward refers to the act of selecting an alternate endpoint

* Type: Integer. A value greater than or equal to 0 indicates the maximum number of retries against all endpoints of the current MI.SourceExtended object for all configured HTTP error codes.

* Mandatory-to-Specify: No. If not specified, any reforward values defined within the http-code-failover-actions property are used.

Property: http-code-failover-actions

* Description: Specifies the different HTTP codes for which failover actions against an origin/uCDN MUST be done.

* Type: Array of MI.HTTPCodeFailoverActions object

* Mandatory-to-Specify: Yes.

Following is an example of MI.HTTPCodeFailover illustrating how different HTTP error codes from an origin/uCDN can be handled with a custom response sent to the player/client. In this example, a MEL metadata expression user-defined variable named "forward-http-code-failover" is set with the string value "404-failure".

```
{
  "generic-metadata-type": "MI.HTTPCodeFailover",
  "generic-metadata-value": {
    "max-reforwards-per-source": 2,
    "http-code-failover-actions": [
      {
        "http-codes": [
          "404"
        ],
        "reforwards": {
          "max-reforwards-per-source": 1,
          "reforwards-per-endpoint": 2
        },
        "error-state": {
          "variable-name": "forward-http-code-failover",
          "variable-value": "404-failure"
        }
      },
      {
        "http-codes": [
          "512"
        ],
        "reforwards": {
          "max-reforwards-per-source": 1,
          "reforwards-per-endpoint": 2
        },
        "error-state": {
          "variable-name": "forward-http-code-failover",
          "variable-value": "404-failure"
        }
      }
    ]
  }
}
```

Figure 7

Continuing with the example, the "forward-http-code-failover" user-defined variable is tested in a subsequent match expression in the context of the Processing Stages Model [I-D.ietf-cdni-processing-stages-metadata] to trigger an HTTP response transformation (in this case, the addition of a custom header).

```
{
  "generic-metadata-type": "MI.ProcessingStages",
  "generic-metadata-value": {
    "client-response": [
      {
        "match": "var.forward-http-code-failover == '404-failure'",
        "stage-metadata": {
          "response-transform": {
            "header-transform": {
              "add": [
                {
                  "name": "x-failure-handling",
                  "value": "forward side 404",
                  "value-is-expression": false
                }
              ]
            },
            "response-status": "404",
            "status-is-expression": false
          }
        }
      }
    ]
  }
}
```

Figure 8

3.1.2.1. MI.HTTPCodeFailoverActions

MI.HTTPCodeFailoverActions is a subobject of MI.HTTPCodeFailover. It defines one or more HTTP error codes against which some failover action MUST be performed.

Property: http-codes

* Description: An array of HTTP response error status codes (see Sections 15.5 and 15.6 of [RFC9110]) received from an origin/uCDN for which a failover MUST be performed.

* Type: Array of HTTP response status codes encoded as strings. Any HTTP status code from 100 to 599, or any of the special values, "2xx", "3xx", "4xx" or "5xx", where "xx" implies everything from 00 to 99. While repeated and redundant values in the array are allowed, they SHOULD be avoided for efficiency (no reason to specify both 5xx and 503, for example).

* Mandatory-to-Specify: Yes

Property: reforwards

* Description: Specifies how the reforwards MUST be done.

* Type: MI.HTTPCodeReforwards object

* Mandatory-to-Specify: Yes

Property: error-state

* Description: Specifies whether to capture any error message for the current error condition.

* Type: SetVariable object specifying a user-defined variable. See the Metadata Expression Language (MEL) specification [I-D.ietf-cdni-metadata-expression-language]

* Mandatory-to-Specify: No. If not specified, a dCDN will not capture any information about the specific HTTP error code.

3.1.2.2. MI.HTTPCodeReforwards

MI.HTTPCodeReforwards is a subobject of MI.HTTPCodeFailoverActions and allows for the specification of reforwards that MUST occur against endpoints of an MI.SourceExtended object.

Property: max-reforwards-per-source

* Description: Specifies the maximum number of reforwards that can be performed across all endpoints of the MI.SourceExtended object for a given set of HTTP error codes.

* Type: Integer. A value greater than or equal to 0 indicates the maximum number of reforwards against an MI.SourceExtended object. A value of 0 indicates that no reforwards will be done against this source for the given set of HTTP error codes.

* Mandatory-to-Specify: No. If not specified, any reforward values defined in the reforwards-per-endpoint property are honored without any caps.

Property: reforwards-per-endpoint

* Description: Specifies the number of reforwards to be performed across the current endpoint of the MI.SourceExtended object for a given set of HTTP error codes.

* Type: Integer. A value of greater than or equal to 0 indicates the number of reforwards against the current endpoint. A value of 0 indicates that no reforwards will be performed against the current endpoint for the given set of HTTP error codes.

* Mandatory-to-Specify: No. If not specified, the default value is 0.

3.1.3. MI.EndpointDetention

MI.EndpointDetention is a subobject of MI.SourceExtended and is used to track the health of an endpoint (see [RFC8006] section 4.3.3) and place it in detention (i.e., take it out of use) if deemed unhealthy. Unhealthy endpoints are not used for future HTTP requests until a set cool-off time is met. Error-based triggers are defined for which an endpoint can be put in detention for a given duration of time.

Property: connection-setup-fail-trigger

* Description: A trigger for the connection setup error, including timeouts for connection establishment, that is fired for a given threshold of such errors.

* Type: EndpointDetentionTrigger object

* Mandatory-to-Specify: No. If not specified, no health check and detention is done for connection errors or connection timeouts.

Property: read-timeout-trigger

* Description: A trigger for the byte read timeout error that is fired for a given threshold of such errors.

* Type: EndpointDetentionTrigger object

* Mandatory-to-Specify: No. If not specified, no health check and detention is done for both first byte and byte read timeout errors.

Property: http-error-code-trigger

* Description: A list of HTTP error codes for which a trigger is fired for a given threshold of such error codes received from an endpoint.

* Type: HTTPErrorCodeTrigger object

* Mandatory-to-Specify: No. If not specified, no health check and detention is done upon receiving HTTP error codes.

Property: detention-seconds

* Description: The time (in seconds) for which an endpoint is put in detention (not used) after any error-based trigger fires.

* Type: Integer. A value greater than 0 defines a time, in seconds, for which an endpoint is put in detention (not used).

* Mandatory-to-Specify: Yes

3.1.3.1. MI.HTTPErrorCodeTrigger

MI.HTTPErrorCodeTrigger is a subobject of MI.EndpointDetention and defines a trigger for detention based on HTTP error codes. Any HTTP code received from an endpoint that matches against the list of defined HTTP error codes counts against a single EndPointTrigger for detention.

Property: trigger

* Description: A trigger for HTTP error codes that is fired for a given threshold of such error codes received in HTTP responses.

* Type: EndpointDetentionTrigger object

* Mandatory-to-Specify: Yes

Property: error-codes

* Description: An array of HTTP error response codes that, when returned from an endpoint, can cause the endpoint to be triggered into an unhealthy state.

* Type: Array of HTTP response codes. The error codes are from 400 to 599, or one of the special values "4xx" or "5xx", where "xx" implies everything from 00 to 99.

* Mandatory-to-Specify: Yes

3.1.3.2. MI.EndpointDetentionTrigger

MI.EndpointDetentionTrigger is a subobject of MI.EndpointDetention and defines a trigger for a specific kind of error against an endpoint.

Property: trigger-value

* Description: The threshold at which an endpoint is triggered to unhealthy state and put in detention.

* Type: EndpointRepeatingFailures object

* Mandatory-to-Specify: Yes

3.1.3.3. MI.EndpointRepeatingFailures

MI.EndpointRepeatingFailures is a subobject of MI.EndpointDetentionTrigger and triggers an endpoint into a failure state if "event-count" errors take place during "time-window-millisec", while accounting for at least "fail-event-percent-threshold"% of the transactions to this endpoint during this period.

Property: event-count

* Description: The number of errors against an endpoint within a duration of time-window-millsec.

* Type: Integer. A value greater than 0.

* Mandatory-to-Specify: Yes

Property: time-window-millsec

* Description: The rolling duration, in milliseconds, for which error statistics of event-count and fail-event-percent-threshold are tracked against an endpoint.

* Type: Integer. A value greater than 0.

* Mandatory-to-Specify: Yes

Property: fail-event-percent-threshold

* Description: The minimum percentage of errors against an endpoint within a duration of time-window-millsec.

* Type: Integer. A value greater than or equal to 0.

* Mandatory-to-Specify: No. If not specified, an endpoint is triggered to an unhealthy state after encountering event-count errors without regards to the overall percentage of such errors.

The following example shows the first MI.SourceExtended object (sources property) with an EndpointDetention object (endpoint-detention property) for the three error conditions of connection setup, read timeout, and HTTP error codes.

The trigger for connection setup errors will put the endpoint in an unhealthy state when there are three such errors within 1000 milliseconds, as long as 10% of attempts to establish a connection (as opposed to attempted HTTP requests) against the endpoint have failed within that time duration.

The trigger for read timeout errors is configured to fire for three such errors within 1000 milliseconds without any condition for error percentage.

The trigger for HTTP error codes is configured to fire for 404 and 5xx response codes with at least one such HTTP code received from the endpoint within 1000 milliseconds.

```
{
  "generic-metadata-type": "MI.SourceMetadataExtended",
  "generic-metadata-value": {
    "sources": [
      {
        "endpoints": [
          "a.origin-tier1.com",
          "b.origin-tier1.com",
          "c.origin-tier1.com"
        ],
        "protocol": "http/1.1",
        "timeout-ms": "4000",
        "failover-errors": [
          "502",
          "503",
          "504",
          "500"
        ],
        "endpoint-detention": {
          "connection-setup-fail-trigger": {
            "trigger-type": "MI.EndpointRepeatingFailures",
            "trigger-value": {
              "event-count": 3,
              "time-window-millisec": 1000,
              "fail-event-percent-threshold": 10
            }
          },
          "read-timeout-trigger": {
            "trigger-type": "MI.EndpointRepeatingFailures",
```

```
              "trigger-value": {
                "event-count": 3,
                "time-window-millisec": 1000
              }
            },
            "http-error-code-trigger": {
              "error-codes": [
                "404, 5xx"
              ],
              "trigger": {
                "trigger-type": "MI.EndpointRepeatingFailures",
                "trigger-value": {
                  "event-count": 1,
                  "time-window-millisec": 1000
                }
              }
            },
            "detention-seconds": 60
          }
        },
        {
          "endpoints": [
            "origin-tier2.com"
          ],
          "protocol": "http/1.1"
        }
      ]
    }
  }
```

Figure 9

3.2. MI.SourceDetention

MI.SourceDetention is a subobject of MI.SourceMetadataExtended and defines detention rules for a set of sources.

Property: detention-full-behavior

* Description: Defines how a response is constructed when every endpoint of each MI.SourceMetadata has been put in detention.

* Type: DetentionFullBehavior object

* Mandatory-to-Specify: No. If not specified, the dCDN-specific behavior is used.

Property: detention-reset-behavior

* Description: After every endpoint of each MI.SourceMetadata is put in detention, it defines early removal of an endpoint from detention without having served the full duration of detention.

* Type: DetentionResetBehavior object

* Mandatory-to-Specify: No. If not specified, an endpoint is removed from detention only upon the completion of the detention duration.

3.2.1. MI.DetentionFullBehavior

A uCDN can configure how a response is constructed when all endpoints are in detention.

Property: serve-if-stale-available

* Description: A stale version of the requested object is served back to the client. If no stale objects exist, other properties of MI.DetentionFullBehavior apply.

* Type: Boolean

* Mandatory-to-Specify: No. The default value is "False", in which case other properties of MI.DetentionFullBehavior apply.

Property: synthetic-response

* Description: The synthetic response to return back to the client.

* Type: MI.SyntheticResponse object. See the Processing Stages Metadata Specification [I-D.ietf-cdni-processing-stages-metadata]

* Mandatory-to-Specify: No. If not specified, the dCDN-specific error response is used.

3.2.2. MI.DetentionResetBehavior

When all endpoints of an MI.SourceMetadata are put in detention, this object allows a uCDN to define the early removal of an endpoint from detention. All endpoints removed from detention through this behavior are available for immediate use for all subsequent client requests.

Property: reset-endpoints

* Description: The list of endpoints to be removed from detention after no healthy endpoints are available across all MI.SourceExtended objects.

* Type: Array of endpoint objects.

* Mandatory-to-Specify: No. The default value is an empty array, indicating that no endpoints can be removed early from detention.

Property: reset-all-endpoints

* Description: All endpoints are removed from detention after no healthy endpoints are available across all MI.SourceExtended objects.

* Type: Boolean

* Mandatory-to-Specify: No. The default value is "False", indicating that no endpoints can be removed early from detention.

The following example shows two MI.SourceExtended objects, each with two endpoints. A detention trigger of type HTTPErrorCodeTrigger for the first source and connection setup failure for the second source are also defined, with each endpoint set to be put in detention for 60 seconds. For the case when all endpoints are put in detention, the defined MI.SourceDetention object allows for the return of a stale version of the object and fallback to return a 502 if no stale object is found. Additionally, the "a2.origin-tier2.com" endpoint is set for early removal from detention.

```
{
  "generic-metadata-type": "MI.SourceMetadataExtended",
  "generic-metadata-value": {
    "sources": [
      {
        "endpoints": [
          "a1.origin-tier1.com",
          "b1.origin-tier1.com",
          "c.origin-tier1.com"
        ],
        "failover-errors": [
          "502",
          "503",
          "504",
          "500"
        ],
        "endpoint-detention": {
          "http-error-code-trigger": {
```

```
            "error-codes": [
              "404, 5xx"
            ],
            "trigger": {
              "trigger-type": "MI.EndpointRepeatingFailures",
              "trigger-value": {
                "event-count": 1,
                "time-window-millisec": 1000
              }
            }
          },
          "detention-seconds": 60
        }
      },
      {
        "endpoints": [
          "a2.origin-tier2.com",
          "b2.origin-tier2.com"
        ],
        "timeout-ms": 4000,
        "endpoint-detention": {
          "connection-setup-fail-trigger": {
            "trigger-type": "MI.EndpointRepeatingFailures",
            "trigger-value": {
              "event-count": 3,
              "time-window-millisec": 1000,
              "fail-event-percent-threshold": 10
            }
          },
          "detention-seconds": 60
        }
      }
    ],
    "source-detention": {
      "detention-full-behavior": {
        "serve-if-stale-available": true,
        "synthetic-response": {
          "headers": [
            {
              "name": "x-error-reason",
              "value": "all sources in detention"
            }
          ],
          "response-status": 502,
          "response-body":
            "req.uri . 'is unavailable due to origin errors'",
          "body-is-expression": true
        }
```

```
        },
        "detention-reset-behavior": {
          "reset-endpoints": [
            "a2.origin-tier2.com"
          ],
          "reset-all-endpoints": true
        }
      }
    }
  }
```

Figure 10

3.3. MI.LoadBalanceMetadata

The MI.LoadBalanceMetadata object is a subobject of MI.SourceMetadataExtended,defining how content acquisition requests are distributed over the MI.SourceExtended objects listed in the MI.SourceMetadataExtended object.

Property: balance-algorithm

* Description: Specifies the algorithm to be used when distributing content acquisition requests over the sources in an MI.SourceMetadataExtended object. The available algorithms are:

o random: Requests are distributed over the sources in proportion to their associated weights.

o content-hash: Requests are distributed over the sources in a consistent fashion, based on the balance-path-pattern property.

o ip-hash: Requests are distributed over the sources in a consistent fashion based on the IP address of the client requestor.

* Type: String, one of the following: random | content-hash | ip-hash

* Mandatory-to-Specify: No. The default is to use sources in preference order as defined in the MI.SourceMetadataExtended object.

Property: balance-weights

* Description: Specifies the relative frequency that a source is used when distributing requests. When specified, the number of elements MUST match the number of sources. For example, if there

are three MI.SourceExtended objects in a MI.SourceMetadataExtended object with balance-weights [1, 2, 1], source 1 will receive 1/4 of the requests, source 2 will receive 2/4 of the requests, and source 3 will receive 1/4 of the requests.

* Type: Array of integers

* Mandatory-to-Specify: No. The default is to use sources in preference order as defined in the MI.SourceMetadataExtended object.

Property: balance-path-pattern

* Description: This property specifies a PCRE regular expression pattern to apply to the URI when calculating the content hash used by the balance-algorithm. For example, "balance-path-pattern": "^/prod/(.*)/.*\.ts$" would distribute requests based on the URI but exclude the /prod prefix and the .ts segment file.

* Type: String (regular expression)

* Mandatory-to-Specify: No. The default is to use the original URI.

In example 1, the MI.LoadBalanceMetadata object distributes content acquisition requests over sources using a content-hash algorithm:

```
{
  "generic-metadata-type": "MI.LoadBalanceMetadata",
  "generic-metadata-value": {
    "balance-algorithm": "content-hash",
    "balance-path-pattern": "^/prod/(.*)/.*\\.ts$"
  }
}
```

Figure 11

In example 2, the MI.LoadBalanceMetadata object distributes content acquisition requests over sources using the random algorithm:

```
{
  "generic-metadata-type": "MI.LoadBalanceMetadata",
  "generic-metadata-value": {
    "balance-algorithm": "random",
    "balance-weights": [
      1,2, 1
    ]
  }
}
```

Figure 12

4. Authentication Types

To meet the typical industry requirements for authenticating CDNs to external origins, two new authentication types are defined (MI.HeaderAuth and MI.AWSv4Auth) to be used within the MI.Auth object defined in [RFC8006], section 4.2.7 These authentication types are designed to leverage the Protected Secrets Metadata standard [I-D.ietf-cdni-protected-secrets-metadata] , allowing for sensitive values to be protected by referencing them from an external keystore as an alternative to embedding them in the clear in the configuration metadata.

4.1. MI.HeaderAuth

The MI.HeaderAuth 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 any

source by including or referencing it under its authentication property. This method of authentication provides a simple capability for a mutually agreed upon header to be added by the CDN to all requests sent to a specific source/origin. Note that if a dynamically generated header value is required, the RequestTransform capabilities within StageProcessing can be used as an alternative to synthesize a value.

Property: header-name

* Description: The name of the authentication header.

* Type: String

* Mandatory-to-Specify: Yes

Property: header-value

* Description: The value of the authentication header (typically a pre-shared key).

* Type: MI.SecretValue object that can either contain the header value in the clear or as a reference to the header value in an external key store. See the Protected Secrets Metadata standard [I-D.ietf-cdni-protected-secrets-metadata] for more details. Clear text values SHOULD NOT be used outside of testing environments.

* Mandatory-to-Specify: Yes

In the following example the Auth object is used for header authentication, referencing an external key vault for the protected header value:

```
{
  "generic-metadata-type": "MI.SourceMetadataExtended",
  "generic-metadata-value": {
    "sources": [
      {
        "endpoints": [
          "origin.example.com"
        ],
        "protocol": "http/1.1",
        "acquisition-auth": {
          "generic-metadata-type": "MI.Auth",
          "generic-metadata-value": {
            "auth-type": "MI.HeaderAuth",
            "auth-value": {
              "header-name": "X-Origin-Auth",
              "header-value": {
                "secret-store-id": "my-key-vault",
                "secret-path":
                  "/source/keys/origin.example.com_header"
              }
            }
          }
        }
      }
    ]
  }
}
```

Figure 13

4.2. MI.AWSv4Auth

The MI.AWSv4Auth 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 any source by including or referencing it under its authentication property.

AWSv4 authentication causes upstream requests to have a signature applied, following the method described in [AWSv4]. A hash-based signature is calculated over the request URI and specified headers, and is provided in an Authorization: header on the upstream request. The signature is tied to a pre-shared secret key specific to an AWS service, region, and key ID.

Property: access-key-id

* Description: The preconfigured ID of the pre-shared authorization secret.

* Type: String

* Mandatory-to-Specify: Yes

Property: secret-access-key

* Description: The pre-shared authorization secret, which is the basis of building the signature.

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

* Mandatory-to-Specify: Yes

Property: aws-region

* Description: The AWS region name that is hosting the service and shares the key ID and corresponding pre-shared secret.

* Type: String

* Mandatory-to-Specify: Yes

Property: aws-service

* Description: The AWS service name that is serving the upstream requests.

* Type: String

* Mandatory-to-Specify: No. The default is "s3" if not specified.

Property: host-name

* Description: The host name to use as part of the signature calculation.

* Type: String

* Mandatory-to-Specify: No. The default is to use the value of the Host: header of the upstream request. This property is available in case the application needs to override the default behavior.

In the following example, the Auth object is used for AWSv4 authentication, referencing an external key vault for the protected access key:

```
{
  "generic-metadata-type": "MI.SourceMetadataExtended",
  "generic-metadata-value": {
    "sources": [
      {
        "endpoints": [
          "origin.example.com"
        ],
        "protocol": "http/1.1",
        "acquisition-auth": {
          "generic-metadata-type": "MI.Auth",
          "generic-metadata-value": {
            "auth-type": "MI.AWSv4Auth",
            "auth-value": {
              "access-key-id": "MYACCESSKEYID",
              "secret-access-key": {
                "secret-store-id": "my-key-vault",
                "secret-path": "/source/keys/aws/s3-key1"
              },
              "aws-region": "us-west-1",
              "aws-service": "s3"
            }
          }
        }
      }
    ]
  }
}
```

Figure 14

5. Security Considerations

The FCI and MI objects defined in 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.

Protection of Authentication Material and Secrets: Some metadata objects describe authentication methods or reference credentials. Metadata SHOULD avoid the use of plaintext secrets in production environments. When metadata must reference secrets, they SHOULD reference items stored in a protected keystore or secret-management service rather than carrying them inline.

Failover, Retry, and Abuse Resistance: It is possible for Metadata directives controlling failover, retry policies, and reforwarding logic to be abused in ways that may cause problematic traffic

amplification. Implementations SHOULD apply conservative limits for retries and failover frequency, and are advised to monitor failover event rates to detect abuse.

6. IANA Considerations

6.1. 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.SourceMetadataExtended	RFCthis
MI.SourceExtended	RFCthis
MI.SourceConnectionControl	RFCthis
MI.SourceByteReadTimeoutActions	RFCthis
MI.SourceTimeoutActions	RFCthis
MI.SourceConnectionRetries	RFCthis
MI.HTTPCodeFailover	RFCthis
MI.HTTPCodeFailoverActions	RFCthis
MI.HTTPCodeReforwards	RFCthis
MI.EndpointDetention	RFCthis
MI.HTTPErrorCodeTrigger	RFCthis
MI.EndpointDetentionTrigger	RFCthis
MI.EndpointRepeatingFailures	RFCthis
MI.SourceDetention	RFCthis
MI.DetentionFullBehavior	RFCthis
MI.DetentionResetBehavior	RFCthis
MI.LoadBalanceMetadata	RFCthis
MI.HeaderAuth	RFCthis
MI.AWSv4Auth	RFCthis

Table 1: CDNI Payload Types

7. 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
* Eric Klein - Sirius XM
* Andrew Ryan - Sirius XM
* Alfonso Siloniz - Telefonica
* Ben Rosenblum - Vecima
* Sanjay Mishra - Verizon

8. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <https://www.rfc-editor.org/info/rfc1123>.

[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>.

[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>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

9. Informative References

[AWSv4] AWS, "Authenticating Requests (AWS Signature Version 4)", <https://docs.aws.amazon.com/AmazonS3/latest/API/sig-v4-authenticating-requests.html>.

[I-D.ietf-cdni-metadata-expression-language] Power, W., Goldstein, G., and A. Warshavsky, "CDNI Metadata Expression Language", Work in Progress, Internet-Draft, draft-ietf-cdni-metadata-expression-language-01, 17 April 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-cdni-metadata-expression-language-01>.

[I-D.ietf-cdni-processing-stages-metadata] Goldstein, G., Power, W., and A. Warshavsky, "CDNI Processing Stages Metadata", Work in Progress, Internet-Draft, draft-ietf-cdni-processing-stages-metadata-00, 17 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-cdni-processing-stages-metadata-00>.

[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-06, 22 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-cdni-protected-secrets-metadata-06>.

[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