Workgroup: Network Working Group
Internet-Draft:
draft-goldstein-cdni-metadata-model-
extensions-01
Updates: 8006, 8008 (if approved)
Published: 25 October 2021
Intended Status: Standards Track
Expires: 28 April 2022
Authors: G. Goldstein
Lumen Technologies
W. Power
Lumen Technologies
G. Bichot
Broadpeak
A. Siloniz
Telefonica

CDNI Metadata Model Extensions

Abstract

Open Caching architecture is a use case of Content Delivery Networks Interconnection (CDNI) in which the commercial Content Delivery Network (CDN) is the upstream CDN (uCDN) and the ISP caching layer serves as the downstream CDN (dCDN). This document proposed extensions to the RFC8006 Metadata Model by way of a set of GenericMetadata objects that address extend the original CDNI capabilities to meet the more general needs of the CDN and Open Caching industry. Extensions to RFC8008 are also introduced to allow a dCDN to advertise support for these extended metadata capabilities.

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 28 April 2022.

Table of Contents

1. Introduction

The Streaming Video Alliance [SVA] is a global association that works to solve streaming video challenges in an effort to improve end-user experience and adoption. The Open Caching Working Group [OCWG] of the Streaming Video Alliance [SVA] is focused on the delegation of video delivery requests from commerical CDNs to a caching layer at the ISP's network. Open Caching architecture is a specific use case of CDNI where the commercial CDN is the upstream CDN (uCDN) and the ISP caching layer is the downstream CDN (dCDN). The interchange of content delivery configuration metadata between the various entities in the delivery ecosystem is essential for efficient interoperability. The need for an industry-standard API and metadata model becomes increasingly important as content and service providers automate more of their operations, and as technologies, such as open caching, require coordination of content delivery configurations. In order to achieve this, the Open Caching Configuration Interface Specification [OC-CI] defines an interface contemplating a set of use cases.

The following capabilities extend the [RFC8006] Metadata Model:

- *Open Caching Configuration Metadata
- *Enhanced Source definitions, with load balancing, failover, and extended authorization methods Section 2.14
- *A rich set of Cache Control Policies Section 2.3 and computed cache keys Section 2.4
- *Rules for generating Dynamic CORS Headers Section 2.5
- *Possibility to activate compression in the Edge independent of the origin content Section 2.2
- *Traffic Types Section 2.16
- *ServiceID Metadata Section 2.13
- *Processing Stage Rules Section 2.10, enabling metadata to be applied conditionally at various stages in the CDN request/response pipeline.
- *Request URI Rewrites
- *HTTP Header Modifications Section 2.10.8
- *HTTP Status Modifications Section 2.10.5
- *Synthetic HTTP Responses Section 2.10.6
- *An Expression Language for matching rules and synthesis of dynamic values Section 4
- *Private Features Section 2.9

For consistency with other CDNI documents this document follows the CDNI convention of uCDN (upstream CDN) and dCDN (downstream CDN) to represent the commercial CDN and ISP caching layer respectively.

This document defines and registers CDNI GenericMetadata objects (as defined in section 4 of [RFC8006]), registers additional CDNI Payload Types (section 7.1 of [RFC8006]), and adds capability objects (extending section 5 in [RFC8008])

1.1. Terminology

The following terms are used throughout this document:

- *API - Application Programming Interface

*AWS - Amazon Web Services

*CDN - Content Delivery Network

*CDNi - CDN Interconnect

*CORS - Cross-Origin Resource Sharing

*CP - Content Provider

*dCDN - Downstream CDN

*DNS - Domain Name System

*FCI - Footprint and Capabilities Advertising Interface

*HREF - Hypertext Reference (link)

*HTTP - Hypertext Transfer Protocol

*IETF - Internet Engineering Task Force

*ISP - Internet Service Provider

*JSON - JavaScript Object Notation

*MEL - Metadata Expression Language

*Object - A collection of properties.

*OC - Open Caching

*OCN - Open Caching Node

*PatternMatch - An object which matches a string pattern

*UA - User Agent

*uCDN - Upstream CDN

*URI - Uniform Resource Identifier

*URN - Uniform Resource Name

*VOD - Video-on-Demand

*W3C - World Wide Web Consortium

Additionally, this document reuses the terminology defined in [RFC6707], [RFC7336], [RFC8006], [RFC8007], [RFC8008], and [RFC8804]. Specifically, we use the following CDNI acronyms:

*uCDN, dCDN - Upstream CDN and Downstream CDN respectively (see [RFC7336])

1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. CDNI Additonal GenericMetadata Objects

Section 4 of [RFC8006] defines a set of GenericMetadata Object Types. Below additional GenericMetadata Objects are defined to meet the more general needs of the CDNs and Open Caching industry.

Note: In the following sections, the term "mandatory-to-specify" is used to convey which properties MUST be included when serializing a given capability object. When mandatory-to-specify is defined as "Yes" for an individual property, it means that if the object containing that property is included in a message, then the mandatory-to-specify property MUST also be included.

2.1. Auth

To meet the CDN industry requirements for origin authentication, two new authentication types are proposed to be registered via the CDNi Media Type Registration process, as described in [RFC7736].

auth-type: headerauth

-Description: Header based authentication is used to pass an HTTP header secret-name and value secret-value to a uCDN when requesting content. The header name and value are agreed upon between parties out of band.

-auth-value: HeaderAuth object as defined in Section 2.1.1

auth-type: awsv4auth

-Description: Allows for the specification of a set of headers to be added to requests that are forwarded to an origin to enable Amazon Web Services AWS authentication as documented by AWS See Specifications Standards References.

-auth-value: AWSv4Auth object specifying the access parameters as defined in Section 2.1.2

2.1.1. HeaderAuth

The HeaderAuth metadata object is used in the auth-value property of the Auth object, as defined in section 4.2.7 of [RFC8006], 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 origin. Note that if a dynamically generated header value is required, the RequestTransform capabilities within StageProcessing (See Section 2.10) can be used.

Property: header-name

-Description: Name of the authentication header.

-Type: String

-Mandatory-to-Specify: Yes

Property: header-value

-Description: Value of the authentication header typically a pre-shared key. Note that this value should not be disclosed it should be protected by some mechanism such as a secret-sharing API which is outside the scope of this specification.

-Type: String

-Mandatory-to-Specify: Yes

2.1.2. AWSv4Auth

The AWSv4Auth metadata object is used in the auth-value property of the Auth object as defined in RFC-8006 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 [AWSv4Method]. A hash-based signature is calculated over the request URI and specified headers, and 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. This is a secret key that should not be disclosed it should be protected by some mechanism such as a secret-sharing API which is outside the scope of this specification.

-Type: String

-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. It defaults to 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. It defaults to using the value of the Host: header of the upstream request. This property is

available in case the application needs to override that behavior.

2.2. AllowCompress

Downstream CDNs often have the ability to compress HTTP response bodies in cases where the client has declared that it can accept compressed responses (via an Accept-Encoding header), but the source/origin has returned an uncompressed response

The specific compression algorithm used by the dCDN is negotiated by the client's Accept-Encoding header according to [RFC7694] (including q= preferences) and the compression capabilities available on the dCDN.

MI.AllowCompress is a new GenericMetadata object that allows the uCDN to control the activation of response compression in the dCDN directly, and allows content providers to disable compression in cases where compressed responses are not handled properly by certain streaming devices. This can be achieved using a match expression on the user agent.

In addition, HeaderTransform allows the uCDN to normalize, or modify, the Accept-Encoding header to allow for fine-grain control over the selection of the compression algorithm (e.g., gzip, compress, deflate, br, etc.).

Properties of AllowCompress object are:

Property: allow-compress

-Description: If set to True then the dCDN will try to compress the response to the client based on the Accept-Encoding request header.

-Type: Boolean

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

2.3. CachePolicy

MI.CachePolicy is a new GenericMetadata object that allows for the uCDN to specify internal caching policies for the dCDN and external caching policies advertised to clients of the dCDN. overriding any cache control policy set in the response from the uCDN

Property: internal

-Description: Specifies the internal cache control policy to be used by the dCDN.

-Type: Number in seconds encoded as string (e.g. 5 is a five second cache) and/or a list of Enumeration [as-is|no-cache| no-store]

-Mandatory-to-Specify: No. The default is to use the cache control policy specified in the response from the uCDN.

Property: external

-Description: Specifies the external cache control policy to be used by clients of this dCDN.

-Type: Number in seconds encoded as string (e.g. 5 is a five second cache) and/or a list of Enumeration [as-is|no-cache| no-store]

-Mandatory-to-Specify: No. The default is to use the cache control policy specified in the response from the uCDN.

Property: force

-Description: If set to True the metadata interface cache policy defined in the MI.CachePolicy will override any cache control policy set in the response from the uCDN. If set to False the MI.CachePolicy is only used if there is no cache control policy provided in the response from the uCDN.

-Type: Boolean

-Mandatory-to-Specify: No. The default is False which will apply the MI.CachePolicy only if no policy is provided in the response from the uCDN.

2.4. ComputedCacheKey

While the properties provided by the standard CDNi metadata Cache object (See Section 4.2.6 [RFC8006]) provide some simple control over the construction of the cache key, it is typical in advanced CDN configurations to generate cache keys that are dynamically constructed via lightweight processing of various properties of the HTTP request and/or response. As an example, an origin may specify a cache key as a value returned in a specific HTTP response header.

MI.ComputedCacheKey is a new GenericMetadata object that allows for the specification of a cache key using the metadata expression language. See Section 4. Typical use cases would involve the construction of a cache key from one or more elements of the HTTP

request. In cases where both the ComputedCacheKey and the Cache object are applied, the ComputedCacheKey will take precedence.

Property: expression

-Description: The expression that specifies how the cache key shall be constructed.

-Type: String. An expression using CDNI-MEL (Section 4) to dynamically construct the cache key from elements of the HTTP request and/or response.

-Mandatory-to-Specify: Yes

2.5. CrossoriginPolicy

Delegation of traffic between an uCDN over a dCDN based on HTTP redirection does change the domain name in the client requests. This represents a cross-origin request that must be managed appropriately using Cross-Origin Resource Sharing (CORS) headers in the responses in the dCDN.

The dynamic generation of CORS headers is typical in modern HTTP request processing and avoids CORS validation forwarded to the uCDN origin servers, particularly with the preflight OPTIONS requests. The CDNI metadata model requires extensions to specify how a dCDN should generate and evaluate these headers.

Simple CORS requests are those where both HTTP method and headers in the request are included in the safe list defined by the World Wide Web Consortium [W3C]. The user agent request can include an origin header set to the URL domain of the webpage where a player runs. Depending on the metadata configuration, the logic to apply by the dCDN is:

1. Validation of the origin header

2. Wildcard usage

3. Set a default value for CORS response headers

When a UA makes a request that includes a method or headers that are not included in the safe-list, the client will make a CORS preflight request using the OPTIONS method to the resource including the origin header. If CORS is enabled and the requests passes the origin validation, the OCN should respond with the set of headers that indicate what is permitted for that resource, including one or more of the following:

1. Allowed methods

2. Allowed credentials

3. Allowed request headers

4. max-age that the OPTIONS request is valid

5. Headers that can be exposed to the client

CrossoriginPolicy allows for the specification of dynamically generated CORS headers.

Property: allow-origin

-Description: Validation of simple CORS requests.

-Type: Object

-Values: One element for each of the following properties.

-Mandatory-to-Specify: Yes

2.5.1. AllowOrigin

The AllowOrigin object has the following properties:

Property: allow-list

-Description: List of valid URLs only scheme and host name that will be used to match the request origin header.

-Type: Array of PatternMatch objects (Section 4.1.5 of [RFC8006])

-Mandatory-to-Specify: Yes

Property: wildcard-return

-Description: If True the dCDN will include a wildcard in the Access-Control-Allow-Origin response header. If False the dCDN will reflect the request origin header in the Access-Control-Allow-Origin response header.

-Type: Boolean

-Mandatory-to-Specify: Yes

Property: expose-headers

-Description: A list of values the dCDN will include in the Access-Control-Expose-Headers response header to a preflight request.

-Type: Array of strings

-Mandatory-to-Specify: No

Property: allow-methods

-Description: A list of values the dCDN will include in the Access-Control-Allow-Methods response header to a preflight request.

-Type: Array of strings

-Mandatory-to-Specify: No

Property: allow-headers:

-Description: A list of values the dCDN will include in the Access-Control-Allow-Headers response header to a preflight request.

-Type: Array of strings

-Mandatory-to-Specify: No

Property: allow-credentials

-Description: The value the dCDN will include in the Access-Control-Allow-Credentials response header to a preflight request.

-Type: Boolean

-Mandatory-to-Specify: No

Property: max-age

-Description: The value the dCDN will include in the Access-Control-Max-Age response header to a preflight request.

-Type: Integer

-Mandatory-to-Specify: No

2.6. NegativeCachePolicy

MI.NegativeCachePolicy is a new GenericMetadata object that allows for the specification of caching policies based on error response codes received from the origin, allowing for fine-grained control of the downstream caching of error responses. For example, it may be desirable to cache error responses at the uCDN for a short period of

time to prevent an overwhelmed origin service from being flooded with requests

Property: error-codes

-Description: Array of HTTP response error status codes (See Sextions 6.5 and 6.6 of [RFC7231]) that if returned from the uCDN will be cached using the cache policy defined by the cache-policy property.

-Type: Array of HTTP response error status codes

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

Property: cache-policy

-Description: MI.CachePolicy to apply to the HTTP response error status codes returned by the uCDN.

-Mandatory-to-Specify: Yes

2.7. CacheBypassPolicy

MI.CacheBypassPolicy is a new GenericMetadata object that allows a client request to be set as non cacheable. It is expected that this feature will be used to allow clients to bypass cache when testing the uCDN fill path. Note, CacheBypassPolicy only applies to the current request. In addition, any content previously cached (by client requests that do not set CacheBypassPolicy) is not evicted.

Property: error-codes

-Description: Array of HTTP response error status codes (See Sextions 6.5 and 6.6 of [RFC7231]) that if returned from the uCDN will be cached using the cache policy defined by the cache-policy property.

-Type: Array of HTTP response error status codes

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

Property: cache-policy

-Description: MI.CachePolicy to apply to the HTTP response error status codes returned by the uCDN.

-Mandatory-to-Specify: Yes

2.8. OcnSelection

MI.OcnSelection is a new GenericMetadata object that allows the uCDN to indicate to the dCDN a preference in terms of OCN selection.

Property: ocn-delivery

-Description: Instructs the dCDN to perform delegation operating a particular medium and/or a transport arrangement.

-Type: An OcnDeliver yObject as defined in Section 2.8.1

2.8.1. OcnDelivery

An OcnDelivery object contains the following properties:

-Property: ocn-medium

oDescription: Instructs the dCDN to perform delegation operating a particular medium. The following values are specified: SATELLITE.

oType: String

oMandatory-to-Specify: No. Either the ocn-medium property or the ocn-transport property must be present.

-Property: ocn-transport

oDescription: Instructs the dCDN to perform delegation operating a particular transport arrangement. The following values are specified: MABR.

oType: String

oMandatory-to-Specify: No. Either the ocn-medium property or the ocn-transport property must be present.

oMandatory-to-Specify: No. At least one of the two properties ocn-type or ocn-delivery must be present.

-Property: ocn-type

oDescription: Instructs the dCDN to perform delegation operating the type of open caching nodes.

oType: A string corresponding to one of the open caching node types announced by the dCDN through the FCI interface.

oMandatory-to-Specify: No. At least one of the two properties ocn-type or ocn-delivery must be present.

-Property: ocn-selection

oDescription: This property enforces the selection of OCNs considering the ocn-type and/or the ocn-delivery properties. False means best-effort.

oType: string. attempt-or-failed and attempt-or-besteffort mean that the delegation must be attempted considering the ocn-type and/or the ocn-delivery properties. If not possible it is considered as an error and either fails configuration failure or the dCDN continues with a best-effort procedure. Last best effort means the dCDN tries its best to fulfill the requested ocn-selection policy.

oMandatory-to-Specify: No. Best-effort is the default OCN selection policy.

2.9. PrivateFeatureList

MI.PrivateFeatureList is a new GenericMetadata configuration object as a base generic object that permits the control of private features. Note that the private features exposed by the dCDN can be advertised through a dedicated FCI object.

Property: features

-Description: The list of feature configuration objects.

-Type: List array of MI.PrivateFeature objects .

-Mandatory-to-Specify: Yes

2.9.1. PrivateFeature

A MI.PrivateFeature object contains the following properties:

-Property: feature-oid

oDescription: The owner organization that has specified that feature.

oType: String

oMandatory-to-Specify: Yes

-Property: feature-type

oDescription: Indicates the typename of the private feature configuration object.

oType: String

oMandatory-to-Specify: Yes

-Property: feature-value

oDescription: Feature configuration object.

oType: Format type is defined by the value of the feature-type property above.

oMandatory-to-Specify: Yes

2.10. ProcessingStages

A ProcessingStages object is a type of GenericMetadata that describes the matching rules, metadata, and transformations to be applied at specific stages in the request processing pipeline.

It is typical in CDN configurations to define matching rules and metadata that are to be applied at specific stages in the request processing pipeline. For example, it may be required to append a host header prior to forwarding a request to an origin, or modify the response returned from an origin prior to storing in the cache. The following four processing stages are defined:

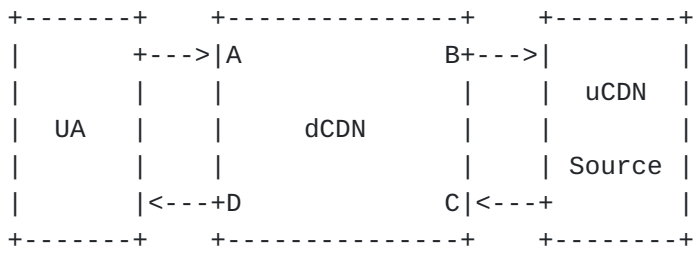

Figure 1: Rule stages

a. clientRequest - Rules run on the client request prior to further processing.

b. originRequest - Rules run prior to making a request to the origin.

c. originResponse - Rules run after a response is received from the origin and before being placed in the cache.

d. clientResponse - Rules run prior to sending the response to the client. If the response is from the cache, rules are applied to the response retrieved from the cache prior to sending to the client.

Each of the four processing stages is represented by an array of StageRules objects, with each StageRules object defining match criteria along with metadata that should be applied if the match applies to true. It should be noted that all of the StageRules objects in the array are evaluated and processed in order. A possible future extension to this processing model could allow for an if-else structure, where processing for a stage is halted upon matching of a StageRule match expression.

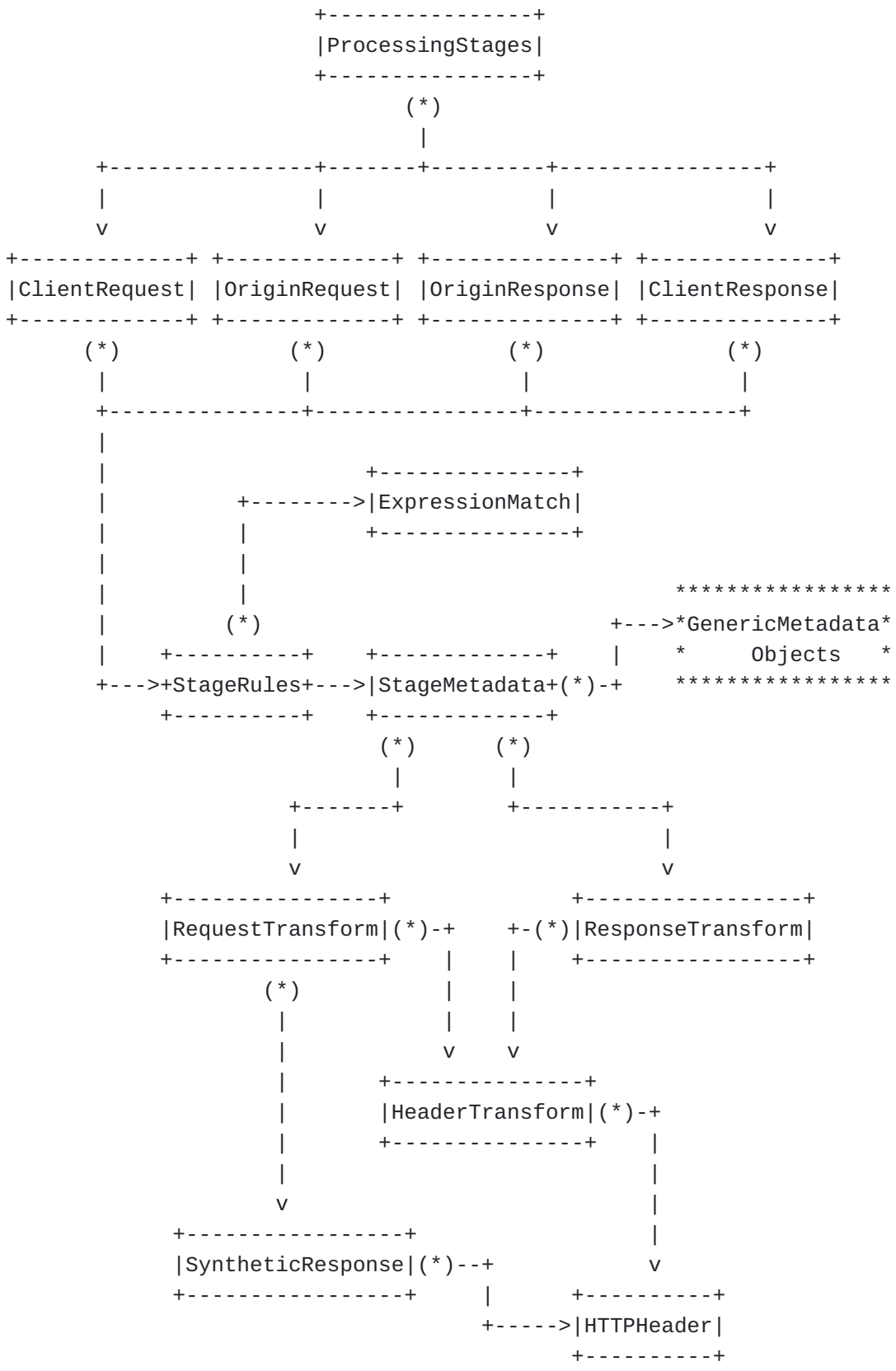

Figure 2: CDNi ProcessingStages metadata model with contained objects

Each of the four processing stages is represented by an array of StageRules objects, with each StageRules object defining criteria along with metadata that should be applied if the match applies to True. It should be noted that the StageRules objects in the array are evaluated and processed in order. A possible future extension to this processing model could allow for an if-else structure, where processing for a stage is halted upon the first match of a StageRules expression.

Property: client-request

-Description: Allows for the specification of conditional metadata to be applied at the client request processing stages as defined in the Rule Processing Stages section. The StageRules in the array are evaluated in order.

-Type: Array of StageRules objects

-Mandatory-to-Specify: No

Property: origin-request

-Description: Allows for the specification of conditional metadata to be applied at origin request processing stages as defined in the Rule Processing Stages section. The StageRules in the array are evaluated in order.

-Type: Array of StageRules objects

-Mandatory-to-Specify: No

Property: origin-response

-Description: Allows for the specification of conditional metadata to be applied at origin response processing stages as defined in the Rule Processing Stages section. The StageRules in the array are evaluated in order.

-Type: Array of StageRules objects

-Mandatory-to-Specify: No

Property: client-response

-Description: Allows for the specification of conditional metadata to be applied at client response processing stages as defined in the Rule Processing Stages section. The StageRules in the array are evaluated in order.

-Type: Array of StageRules objects

-Mandatory-to-Specify: No

2.10.1. StageRules

A StageRules object is used within the context of ProcessingStages to define elements in a list of match rules and stage-specific metadata and transformations that should be applied conditionally on a rich expression match.

Property: match

-Description: An ExpressionMatch object encapsulating a rich expression using the CDNi Metadata Expression Language CDNI-MEL to evaluate aspects of the HTTP request and/or response. The stage-metadata rules are only applied if the match evaluates to True or if no match expression is provided

-Type: ExpressionMatch object

-Mandatory-to-Specify: No. The stage-metadata rules are always applied if no match expression is provided. This would be the case when stage-metadata should be applied unconditionally within the context of the higher-level host and path matches.

Property: stage-metadata

-Description: Specifies the set of StageMetadata to be applied at the processing stage if the match expression evaluates to True or is not present.

-Type: Array of StageMetadata objects applied in order.

-Mandatory-to-Specify: Yes

2.10.2. ExpressionMatch

CDN and open caching systems often require a rich set of matching rules, with full regular expressions and Boolean combinations of matching parameters for host, path, and header elements of a request. In typical CDN implementations, this capability is provided by a rich expression language that can be embedded in the metadata configurations

The ExpressionMatch object contains the rich expression that must evaluate to True for the StageMetadata to be applied for the specific StageRules. Defining expressions as stand-alone objects

allows for sets of reusable match expressions to be reused via metadata reference linking.

Property: expression

-Description: A rich expression using CDNI-MEL to evaluate aspects of the HTTP request and/or response. See documentation on the Metadata Expression Language for details on the expression of matching variables and syntax.

-Type: String using CDNI-MEL syntax. See Section 4

-Mandatory-to-Specify: Yes

2.10.3. StageMetadata

The StageMetadata object contains GenericMetadata and HTTP request/response transformations that should be applied for a StageRules match. The following table defines the processing stages where request and response transformations are possible:

Stage	request-transform	response-transform
clientRequest	Yes	Yes
originRequest	Yes	Yes
originResponse	Yes	No
clientResponse	Yes	No

Table 1

Note that for the stages where both request and response transformations are allowed, it is possible to specify both. This may be the case if, for example, the request URI needs alteration for cache-key generation and the response headers need to be manipulated.

Property: generic-metadata

-Description: Specifies the set of GenericMetadata to be applied for a StageRules match. A typical use case would be the application of a CachePolicy or TimeWindowACL conditionally on matching HTTP headers. Support for this capability is optional and can be advertised via feature-flags in the FCI interface.

-Type: Array of GenericMetadata applied in order. Note that not all GenericMetadata object types may be applicable at all processing stages.

-Mandatory-to-Specify: No. The generic-metadata property would not be needed when StageMetadata is used to only specify

request or response transformations such as modifications of HTTP headers.

Property: request-transform

-Description: Specifies a transformation to be applied to the HTTP request for a StageRules match. The transformation can be the modification of any request header and/or the modification of the URI. Modifications are applied such that downstream processing stages receive the modified HTTP request as their input. Support for this capability is optional and can be advertised via feature-flags in the FCI interface.

-Type: RequestTransform object

-Mandatory-to-Specify: No

Property: response-transform

-Description: Specifies a transformation to be applied to the HTTP response for a StageRules match. The transformation can be the modification of any response header HTTP response status code or the generation of a synthetic response. Modifications are applied such that downstream processing stages receive the modified HTTP response as their input. Support for this capability is optional and can be advertised via feature-flags in the FCI interface.

-Type: ResponseTransform object

-Mandatory-to-Specify: No

2.10.4. RequestTransform

The RequestTransform object contains metadata for transforming the HTTP request for a specific StageRules object. The transformation can be the modification of any request header and/or the modification of the URI. Modifications are applied such that downstream processing stages receive the modified HTTP request as their input.

Property: headers

-Description: A HeaderTransform object specifying HTTP request headers to add replace or delete.

-Type: HeaderTransform object

-Mandatory-to-Specify: No

Property: uri

-Description: Replacement value for the HTTP request.

-Type: String. Either a literal static string or an expression using CDNI-MEL to dynamically construct a URI value from elements of the HTTP request and/or response.

-Mandatory-to-Specify: No

Property: uri-is-expression

-Description: Flag to signal whether the URI is a static string literal or a CDNI-MEL expression that needs to be dynamically evaluated.

-Type: Boolean

-Mandatory-to-Specify: No. The default is False indicating that the URI is a string literal and does not need to be evaluated.

2.10.5. ResponseTransform

The ResponseTransform object contains metadata for transforming the HTTP response for a StageRules match. The transformation can be the modification of any response header, HTTP response status code, or the generation of a synthetic response. Modifications are applied such that downstream processing stages receive the modified HTTP response as their input.

Property: headers

-Description: A HeaderTransform object specifying HTTP response headers to add replace or delete.

-Type: HeaderTransform object

-Mandatory-to-Specify: No

Property: response-status

-Description: Replacement value for the HTTP response status code.

-Type: Integer. Either a static integer or an expression using CDNI-MEL that evaluates to an integer to dynamically generate an HTTP status code based on elements of the HTTP request and/or response. Expressions that do not evaluate to an integer shall be considered invalid and result in no override of origin-provided response status.

-Mandatory-to-Specify: No

Property: status-is-expression

-Description: Flag to signal whether the response-status is a static integer or a CDNI-MEL expression that needs to be dynamically evaluated to generate an HTTP response status code.

-Type: Boolean

-Mandatory-to-Specify: No. The default is False indicating that the response-status is a static integer and does not need to be evaluated.

Property: synthetic

-Description: Specification of a complete replacement of any HTTP response that may have been generated in an earlier processing stage with a synthetic response. Use of this property to specify a synthetic response would override any response transformations or status codes specified by other properties.

-Type: SyntheticResponse object

-Mandatory-to-Specify: No

2.10.6. SyntheticResponse

It is quite common in CDN configurations to specify a synthetic response be generated based on inspection of aspects of the original request or the origin response.

The SyntheticResponse object allows for the specification of a synthetic response to be generated in response to the HTTP request being processed. The synthetic response can contain a set of response headers, a status code, and a response body, and is a complete replacement for any HTTP response elements generated in an earlier processing stage.

A dynamically generated Content-Length HTTP response header is generated based on the length of the generated response body.

Property: headers

-Description: An array of HTTP header objects that specify the full set of headers to be applied to the synthetic response.

-Type: Array of HTTP header objects

-Mandatory-to-Specify: No although it would be unusual to not specify minimal standard response headers such as Content-Type.

Property: response-status

-Description: HTTP response status code.

-Type: Integer. Either a static integer or an expression using CDNI-MEL that evaluates to an integer to dynamically generate an HTTP status code based on elements of the upstream HTTP request and/or response. Expressions that do not evaluate to an integer shall be considered invalid and result in a 500 status for the synthetic response.

-Mandatory-to-Specify: Yes

Property: status-is-expression

-Description: Flag to signal whether the response-status is a static integer or a CDNI-MEL expression that needs to be dynamically evaluated to generate an HTTP response status code.

-Type: Boolean

-Mandatory-to-Specify: No. The default is False indicating that the response-status is a static integer and does not need to be evaluated.

Property: body

-Description: Body for the synthetic HTTP response. The response body can either be static or dynamically constructed from a rich expression.

-Type: String. Either a literal static string or an expression using CDNI-MEL to dynamically construct a response body from elements of the HTTP request and/or response.

-Mandatory-to-Specify: No. If absent an empty HTTP response with a zero-value Content-Length header is generated.

Property: body-is-expression

-Description: Flag to signal whether the synthetic response body is a static string literal or a CDNI-MEL expression that needs to be dynamically evaluated.

-Type: Boolean

-Mandatory-to-Specify: No. The default is False indicating that the body is a string literal and does not need to be evaluated.

2.10.7. HeaderTransform

In processing HTTP requests, it is often required to modify HTTP request or response headers at one of the processing stages, requiring CDNi metadata to have the capability to update any field in an HTTP request or response header. It should be noted that certain HTTP headers (such as Set-Cookie) have multiple occurrences in a request or response, thereby requiring that we allow for add and replace designations for header modification.

The HeaderTransform object specifies how HTTP headers should be added, replaced, or deleted from HTTP requests and responses.

Property: add

-Description: List of HTTP headers name/value pairs that should be added to the HTTP request or response. Note that any existing headers in the request or response with the same names of those added are not affected resulting in multiple headers with the same name.

-Type: Array of HTTPHeader objects containing header name/value pairs

-Mandatory-to-Specify: No

Property: replace

-Description: List of HTTP headers name/value pairs that should be added to the HTTP request or response replacing any previous headers with the same name.

-Type: Array of HTTPHeader objects containing header name/value pairs

-Mandatory-to-Specify: No

Property: delete

-Description: List of names of HTTP headers that should be deleted from the

-HTTP request or response. If a named header appears multiple times all occurrences are deleted.

-Type: Array of strings with each string naming an HTTP header to delete

-Mandatory-to-Specify: No

2.10.8. HttpHeader

The HTTPHeader object contains a name/value pair for an HTTP header to add or replace in a request or response. The CDNI-MEL Section 4 expression language can be used to dynamically generate response values.

Property: name

-Description: Name of the HTTP header.

-Type: String

-Mandatory-to-Specify: Yes

Property: value

-Description: New value of the named HTTP header.

-Type: String. Either a static string or an expression using CDNI-MEL to dynamically construct a header value from elements of the HTTP request and/or response.

-Mandatory-to-Specify: Yes

Property: value-is-expression

-Description: Flag to signal whether the value is a static string literal or a CDNI-MEL Section 4 expression that needs to be dynamically evaluated.

-Type: Boolean

-Mandatory-to-Specify: No. The default is False indicating that the value is a string literal and does not need to be evaluated.

2.11. RequestedCapacityLimits

MI.RequestedCapacityLimits is a new GenericMetadata object that allows the uCDN to communicate to the dCDN a desired change in the advertised traffic delegation limits for a given host and footprint.

Property: requested-limits

-Description: A set of requested changes to the dCDNs advertised FCI.CapacityLimits [capacity-insights-advertisement].

-Type: Array of RequestedCapacityLimit objects

-Mandatory-to-Specify: Yes

RequestedCapacityLimit objects contain the following properties:

-Property: limit-type

oDescription: The limit type for which the change is requested corresponding to the limit types of FCI.CapacityLimits [capacity-insights-advertisement].

oType: String. One of egress requests storage-size storage-objects sessions or cache-size

oMandatory-to-Specify: Yes

-Property: limit-value

oDescription: The requested new value for the limit. Units of this value are dependent upon the limit-type and are defined in the SVA Capacity Insights Interface Specification.

oType: Integer

oMandatory-to-Specify: Yes

-Property: footprints

oDescription: dCDN footprints with advertised FCI.CapacityLimits [capacity-insights-advertisement] for which this request applies.

oType: List of CDNI footprint objects from the CDNI Metadata Footprint Types registry of [RFC8006]

oMandatory-to-Specify: Yes

2.12. RequestRouting

MI.RequestRouting is a new GenericMetadata object that allows the uCDN to force the dCDN request routing mode(s) to be applied when working in iterative redirection mode. The list of redirection modes supported by the dCDN is advertised through the FCI.RedirectionMode object. See Section 5.5 of [RFC8006]. The list of request routing modes supported by the dCDN is advertised through the FCI.RequestRouting object (Section 2.12)

Property: request-routing-modes

-Description: Instructs the dCDN to perform request routing according to one or more preferred modes among those supported and advertised by the dCDN through the FCI.RequestRouting object. One must understand that forcing instead of letting the dCDN request router select one particular request routing mode may trigger some inefficiency in the request routing process.

-Type: List array of iterative request routing modes, defined as Enumeration of [DNS|HTTP|MANIFESTREWRITE] encoded as an uppercase string

-Mandatory-to-Specify: No. By default all request routing modes supported by the dCDN can be used by the dCDN as part of its request routing process.

2.13. ServiceIDs

CDN configurations typically have multiple layers of identifiers that group configurations by customer account to facilitate logging, billing, and support operations. The metadata model sis extended to allow for the association service identifier metadata to a host or path match and to allow for these IDs to be dynamically generated via an expression language. For example, it may be necessary to extract a portion of the Request URI path to derive a service identifier (e.g.: /news/* maps to one serviceID and /movies/* maps to a different serviceID)

A MI.ServiceIDs is a new GenericMetadata object that allows for the specification of two tiers of CDN-specific identifiers and names. The interpretation of these identifiers is implementation specific.

Property: service-id

-Description: A provider-specific identifier for the service typically a customer account identifier.

-Type: String. Either a literal static string or an expression using CDNI-MEL Section 4 to dynamically construct the ID from elements of the HTTP request and/or response.

-Mandatory-to-Specify: No

Property: service-id-is-expression

-Description: Flag to signal whether the service-id is a static string literal or a CDNI-MEL Section 4 expression that needs to be dynamically evaluated.

-Type: Boolean

-Mandatory-to-Specify: No. The default is False indicating that the service-id is a string literal and does not need to be evaluated.

Property: service-name

-Description: Human-readable name for the service-id.

-Type: String

-Mandatory-to-Specify: No

Property: property-id

-Description: A provider-specific identifier for the property typically identifies a child configuration within the parent service-id.

-Type: String. Either a literal static string or an expression using CDNI-MEL Section 4 to dynamically construct the ID from elements of the HTTP request and/or response.

-Mandatory-to-Specify: No

Property: property-id-is-expression

-Description: Flag to signal whether the property-id is a static string literal or a CDNI-MEL Section 4 expression that needs to be dynamically evaluated.

-Type: Boolean

-Mandatory-to-Specify: No. The default is False indicating that the property-id is a string literal and does not need to be evaluated.

Property: property-name

-Description: Human-readable name for the property-id.

-Type: String

-Mandatory-to-Specify: No

2.14. SourceMetadataExtended

MI.SourceMetadataExtended is an alternative to the CDNi standard MI.SourceMetadata object (See Section 4.2.1 of [RFC8006]), which adds a property to specify load balancing across multiple sources, as well as a SourceExtended sub-object with additional attributes to the CDNi standard Source object (See Section 4.2.1.1 of [RFC8006]). While both SourceMetadataExtended and SourceMetadata can be provided for backward compatibility, a dCDN that advertises capability for SourceMetadataExtended will ignore SourceMetadata if both are provided for a given host or path match.

Property: sources

-Description: Sources from which the dCDN can acquire content listed in order of preference.

-Type: Array of SourceExtended objects

-Mandatory-to-Specify: No. Default is to use static configuration out-of-band from the CDNI metadata interface.

Property: load-balance

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

-Type: LoadBalanceMetadata object

-Mandatory-to-Specify: No

2.14.1. SourceExtended

SourceExtended is an alternative to the CDNi standard Source (Section 4.2.1.1 of [RFC8006]) object with additional metadata. Open Caching use cases requires additional capabilities not covered in [RFC8006] as:

1. Web root path specification for the source.

2. Support for additional forms of origin authentication

3. Validate a host header in the request different than the defined in endpoints

4. Whether a Source response of HTTP redirect is passed through to the user agent or followed by the dCDN to acquire a content

It inherits all the attributes of the Source object (acquistion-auth, endpoints, and protocol), with the following additions:

Property: origin-host

-Description: 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: webroot

-Description: The path element that is prepended to a resources URI before

-retrieving content from a uCDN.

-Type: String

-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: 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 sources array is tried. Once a connection is established timeout-ms is used on subsequent reads of data from the uCDN.

-Type: Integer

-Mandatory-to-Specify: No. The default is to revert to RFC-8006 behavior.

Property: failover-errors

-Description: Array of HTTP response error status codes (See Sections 6.5 and 6.6 of [RFC7231]) that if returned from the uCDN will trigger a failover to the next source in the MI.SourceMetadataExtended (Section 2.14) sources 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.

-Type: Array of HTTP response error status codes.

-Mandatory-to-Specify: No. The default is to revert to RFC-8006 behavior.

2.14.2. LoadBalanceMetadata

The LoadBalanceMetadata object defines how content acquisition requests are distributed over the SourceExtended objects listed in the SourceMetadataExtended object.

It permits the following capabilities:

1. Multi-origin failover: ability to specify a list of origins that can act as fallbacks to the primary origin. Failure rules can specify types of errors and timeout values that trigger failover

2. Multi-origin load balancing: The ability to specify a list of origins that can be selected by one of several balancing rules (round robin, content hash, IP hash).

3. Origin Shielding Definitions - The ability to designate CDN origin shield locations associated with a specific origin

Parameters for LoadBalanceMetadata are:

Property: balance-algorithm

-Description: Specifies the algorithm to be used when distributing content acquisition requests over the sources in

a SourceMetadataExtended object. The available algorithms are random content-hash and ip-hash

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

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

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

-Type: Enumeration [random|content-hash|ip-hash] encoded as a lowercase string.

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

Property: balance-weights

-Description: This property specifies the relative frequency that a source is used when distributing requests. For example if there are three SourceExtended objects in a SourceMetadataExtended object with balance-weights 1 2 1 source 1 will receive 14 of the requests source 2 will receive 24 of the requests source 3 will receive 14 of the requests.

-Type: Array of integers

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

Property: balance-path-pattern

-Description: This property specifies a 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 excluding the prod prefix and the .ts segment file.

-Type: String regular expression

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

2.15. StaleContentCachePolicy

MI.StaleContentCachePolicy is a new GenericMetadata object that allows the uCDN to specify the policy how the dCDN should process requests for stale content. For example, this policy allows the content provider to specify that stale content be served from cache for a specified time period while refreshes from the origin occur asynchronously.

Property: stale-while-revalidating

- Description: Instructs the dCDN to serve a stale version of a resource while refreshing the resource with the uCDN. When set to True the dCDN will return a previously cached version of a resource while the resource is refreshed with the uCDN in the background.

- Type: Boolean

- Mandatory-to-Specify: No. The default is False which waits for the uCDN to refresh a resource before responding to the client.

Property: stale-if-error

- Description: Instructs the dCDN to serve a stale version of a resource if an error was received when trying to refresh the resource with the uCDN. When set the dCDN will return a previously cached version of a resource instead of caching the error response. Per [RFC5861] Section 4 an error is any situation that would result in a 500 502 503 or 504 HTTP response status code being returned

- Type: Array of HTTP response error status codes

- Mandatory-to-Specify: No. The default is to cache the error response received from the uCDN.

Property: failed-refresh-ttl

- Description: Instructs the dCDN to serve a stale version of a resource for the number of seconds specified in failed-refresh-ttl before trying to revalidate the resource with the uCDN. Use of failed-refresh-ttl allows the load to be reduced on the uCDN during times of system stress.

- Type: Integer

- Mandatory-to-Specify: No

2.16. TrafficType

Content delivery networks often apply different infrastructure, network routes, and internal metadata for different types of traffic. Knowing those differences, a dCDN provider can implement specific strategies that will maximize performance and thereby provide more available capacity to the upstream provider. it should be noted that the dCDNs handling of the traffic types is implementation-specific and not prescribed here.

TrafficType metadata defines a set of descriptors that characterize either the type or usage of the traffic, enabling dCDNs to apply any internal configuration rules without exposing an unnecessary amount of internal details.

Property: traffic-type

-Description: A literal that defines the traffic type. uCDN will use the literal that is most representative of the traffic being delegated. The following literals can be specified:

ovod

olive

oobject-download

-Type: String

-Mandatory-to-Specify: Yes

Property: hints

-Description: Other traffic characteristics that the uCDN can indicate to the dCDN as suggestions for service optimization. Accepts free-form unconstrained values.

-Type: Array of strings

-Mandatory-to-Specify: No

3. CDNI Additional FCI Objects

Section 5 of [RFC8008] describes the FCI Capability Advertisement Object, which includes a CDNI Capability Object as well as the capability object type (a CDNI Payload Type). The section also defines the Capability Objects per such type. Below we define additional Capability Objects.

In most cases, the presence or absence of a GenericMetadata object name in FCI.Metadata (as described above), is sufficient to convey support for a capability. There are cases, however, where more fine-grained capabilities declarations are required. Specifically, a dCDN may support some, but not all, of the capabilities specified by one of the new GenericMetadata objects. In these cases, new FCI objects are created to allow a dCDN to express these fine-grained capabilities.

Note: In the following sections, the term "mandatory-to-specify" is used to convey which properties MUST be included when serializing a given capability object. When mandatory-to-specify is defined as "Yes" for an individual property, it means that if the object containing that property is included in an FCI message, then the mandatory-to-specify property MUST also be included.

3.1. FCI.AuthTypes

This object is used to to indicate the support of authentication methods to be used for content acquisition (while interacting with an origin server) and authorization methods to be used for content delivery.

This documents defines two new authentication methods (See Section 2.1), while there is one other authorization method under specification in CDNI called URI signing [URI.signing]

Property: authe-types

- Description: List of supported authentication methods possibly required for content acquisition

- Type: List of supported authentication methods (e.g. as defined in Section 2.1)

- Mandatory-to-Specify: No. No authentication method is supported in this case.

Property: autho-types

- Description: List of supported authorization methods possibly required for content delivery

- Type: List of supported authorization methods (e.g. URI signing [URI.signing])

- Mandatory-to-Specify: No. No authorization method is supported in this case.

3.2. FCI.ProcessingStages

This object is used to signal the set of features that are supported in relation with the ProcessingStages (Section 2.10) configuration object. Those optional features depend on the CDNI-MEL language (Section 4) support.

Property: features

-Description: List of supported optional processing stages features. Note that these features all have some dependencies on support of the CDNi MEL expression language.

-Type: Enumeration [ExpressionMatch|RequestTransform|ResponseTransform] encoded as string

-Mandatory-to-Specify: No. None of these optional features are supported in this case.

3.3. FCI.SourceMetadataExtended

This object is used to signal the supported features related to the SourceMetadataExtended configuration object.

Property: load-balance

-Description: List of supported load balancing algorithms in relation to the SourceMetadataExtended configuration object see GenericMetadata: SourceMetadataExtended

-Type: Enumeration [random|content-hash|ip-hash] encoded as lowercase strings

-Mandatory-to-Specify: No. load balancing is not supported among sources.

If the FCI.SourceMetadataExtended object is not exposed neither advertised or if the load-balance array is empty, the dCDN does not support the usage of the load-balance property attached to the SourceMetadataExtended configuration object (see SourceMetadataExtended (Section 2.14)).

3.4. FCI.RequestRouting

This object is used by the dCDN to signal/announce the supported request routing modes. This can be optionally used by the uCDN to

further select a subset of those modes when operating one of the iterative delegation modes. See Section 2.12

Property: request-routing-modes

-Description: List of supported request routing modes by the dCDN. This information is useful when the uCDN decides to perform a delegation in iterative mode.

-Type: Enumeration [DNS|HTTP-R|MANIFESTREWRITE] encoded as uppercase strings

-Mandatory-to-Specify: No. If the dCDN does not advertise the supported request routing modes they are all supported by default.

3.5. FCI.PrivateFeatures

This object is used by the dCDN to signal/announce the list of supported private features. See Section 2.9

Property: features

-Description: The list of supported private feature

-Type: List array of nested objects each containing the following properties:

oProperty: feature-oid

oDescription: The owner/organization that has specified the feature.

oType: String

oMandatory-to-Specify: Yes

oProperty: feature-type

oDescription: Indicates the typename of the private feature configuration object.

oType: String

oMandatory-to-Specify: Yes

3.6. FCI.OcnSelection

This object is used by the dCDN to signal/announce the supported OCN types and/or their transport arrangement and/or medium supported by OCNs.

Property: ocn-delivery-list

-Description: List of supported medium and/or transport arrangements.

-Type: List of OcnDeliveryList objects Section 3.6.1

Property: ocn-type-list

-oDescription: List of supported OCN types. Examples include: HOME or EDGE.

-oType: Array of strings

-oMandatory-to-Specify: No

3.6.1. OcnDeliveryList

Property: ocn-medium

-Description: This property lists the supported mediums.

-Type: List of strings. The following values are specified: SATELLITE

-Mandatory-to-Specify: No

Property: ocn-transport

-Description: Instructs the dCDN to perform delegation operating a particular transport arrangement. The following values are specified: MABR.

-Type: List of strings

-Mandatory-to-Specify: No

4. Metadata Expression Language

The CDNI Metadata Expression Language provides a syntax with a rich set of variables, operators, and built-in functions to facilitate use cases within the extended CDNi metadata model.

> Enables expression matching to dynamically determine if StageMetadata (Section 2.10.3) should be applied at a StageRules match.
>
> Enables the dynamic construction of a value to be used in scenarios such as constructing a service identifier or cache key, setting an HTTP header, rewriting a request URI, setting a response status code, or dynamically generating a response body for a SyntheticResponse.

Expressions can evaluate to a Boolean, string, or integer, depending on the use case:

Usage	Description	Evaluation Results
ExpressionMatch.expression	Dynamically determines if StageMetadata should be applied at a specific StageRules.	Boolean. Expressions that do not evaluate to True or False shall be considered as False.
RequestTransform.uri	Rewrites request URI that will be presented to all downstream stages.	String
ResponseTransform.response-status	Dynamically sets a response status code to replace the status-code returned by the origin.	Integer (HTTP status code)
SyntheticResponse.response-status	Dynamically sets a response status code for a synthetically constructed response.	Integer (HTTP status code)
SyntheticResponse.body	Dynamically constructs a response body.	String
HTTPHeader.value		String

Usage	Description	Evaluation Results
	Dynamically constructs a header value.	
ComputedCacheKey.expression	Dynamically constructs a cache key.	String
ServiceIDs.properry-id,ServiceIDs.service-id	Dynamically constructs service and property identifiers.	String

Table 2: CDNI MEL expressions

4.1. Expression Variables

Variable	Meaning
req.h.<name>	Request header <name>
req.uri	Request URI (includes query string and fragment identifier, if any)
req.uri.path	Request URI path
req.uri.pathquery	Request path and query string
req.uri.query	Request query string
req.uri.query.<key>	Request query string value associated with <key>
req.method	Request HTTP method (GET, POST, others)
resp.h.<name>	Response header <name>
resp.status	Response status code

Table 3: CDNI MEL variables

4.2. Expression Operators and keywords

Operator	Type	Result Type	Meaning
==	infix	Boolean	Equality test
!=	infix	Boolean	Inequality test
!	infix	Boolean	Logical NOT operator
>	infix	Boolean	Greater than test
<	infix	Boolean	Less than test
>=	infix	Boolean	Greater than or equal test
<=	infix	Boolean	Less than or equal
*=	infix	Boolean	Glob style match
~=	infix	Boolean	Regular expression match (see https://www.pcre.org/ for details on PCRE RegEx matching)
ipmatch	infix	Boolean	

Operator	Type	Result Type	Meaning
			Match against IP address or CIDR (IPv4 and IPv6)
+	infix	Numeric	Addition
-	infix	Numeric	Subtraction
*	infix	Numeric	Multiplication
/	infix	Numeric	Division
%	infix	Unsigned or Integer	Modulus
.	infix	String	Concatenation
? :	ternary	*	Conditional operator: <e> ? <v1> : <v2> Evaluates <v1> if <e> is true, <v2> otherwise.
( )	grouping		Used to override precedence and for function calls.

Table 4: CDNI MEL expression operators

Keyword	Meaning
and	Logical AND
or	Logical OR
not	Logical NOT (see also the ! operator)
nil	No value (distinct from empty value)
true	Boolean constant: true
false	Boolean constant: false

Table 5: CDNI MEL expression keywords

4.3. Expression Built-in Functions

4.3.1. Basic Functions: Type Conversions

Function	Action	Argument(s)	Returns
integer(e)	Converts expression to integer.	1	integer
real(e)	Converts expression to real.	1	real
string(e)	Converts expression to string.	1	string
boolean(e)	Converts expression to Boolean.	1	Boolean

Table 6: CDNI MEL type conversions

4.3.2. Basic Functions: String Conversions

Function	Action	Argument(s)	Returns
upper(e)	Converts a string to uppercase. Useful for case-insensitive comparisons.	1	string
lower(e)		1	string

Function	Action	Argument(s)	Returns
	Converts a string to lowercase. Useful for case-insensitive comparisons.		

Table 7: CDNI MEL string conversions

4.3.3. Convenience Functions

Function	Action	Argument(s)	Returns
match(string Input, string Match)	Regular expression Match is applied to Input and the matching element (if any) is returned. Empty string is returned if there is no match. See https://www.pcre.org/ for details on PCRE RegEx matching.	2	string
match_replace(string Input, string Match, string Replace)	Regular expression Match is applied to Input arg and replaced with the Replace arg upon successful match. Returns updated (replaced) version of Input.	3	string
add_query(string Input, string q, string v)	Add query string element q with value v to the Input string. If v is nil, then just add the query string element q. The query element q and value v must conform to the format defined in: https://datatracker.ietf.org/doc/html/rfc3986	2	string
remove_query(string Input, string q)	Remove (all occurrences of) query string element q from the Input string.	2	string
path_element(string Input, integer n)	Return the path element n from Input. -1 returns the last element.	2	string
		3	string

Function	Action	Argument(s)	Returns
path_element(string Input, integer n, integer m)	Return the path elements from position n to m.		

Table 8: CDNI MEL convenience functions

4.4. Error Handling

4.4.1. Compile Time Errors

To ensure reliable service, all CDNI Metadata configurations MUST be validated for syntax errors before they are ingested into a dCDN. That is, existing configurations should be kept as the live running configuration until the new configuration has passed validation. If errors are detected in a new configuration, the configuration MUST be rejected. A HTTP 500 Internal Server Error should be returned with a message that indicates the source of the error (line number, and configuration element that caused the error).

Examples of compile-time errors:

1. Configuration does not parse relative to the CDNI Metadata JSON schema

2. Unknown CDNI Metadata object referenced in the configuration

3. CDNI Metadata object parse error

    a. Missing mandatory CDNI Metadata property

    b. Unknown CDNI Metadata property

    c. Incorrect type for a CDNI Metadata property value

4. CDNI-MEL

    a. Unknown CDNI-MEL variable name referenced in an expression

    b. Unknown CDNI-MEL operator, key-word, or functions referenced in an expression

    c. Incorrect number of arguments used in a CDNI-MEL expression operator or function

    d. Incorrect type of argument used in a CDNI-MEL expression operator or function

4.4.2. Runtime Errors

If a runtime error is detected when processing a request, the request should be terminated, and a HTTP 500 'Internal Server Error' returned to the caller. To avoid security leaks, sensitive information MUST be removed from the error message before it is returned to an external client. In addition to returning the HTTP 500 error, the dCDN SHOULD log additional diagnostics information to assist in troubleshooting.

Examples of runtime errors:

1. Failure to allocate memory (or other server resources) when evaluating a CDNI-MEL expression

2. Incorrect runtime argument type in a CDNI-MEL expression. E.g., trying to convert a non-numeric string to a number

4.5. Expression Examples

4.5.1. ComputedCacheKey

Sets the MI.ComputedCacheKey to the value of the X-Cache-Key header from the client request.

```
{
  "generic-metadata-type": "MI.ComputedCacheKey",
  "generic-metadata-value": {
    "expression": "$req.h.x-cache-key"
  }
}
```

Sets the MI.ComputedCacheKey to the lowercase version of the URI.

```
{
  "generic-metadata-type":  "MI.ComputedCacheKey",
  "generic-metadata-value": {
    "expression": "$lower(req.uri)"
  }
}
```

4.5.2. ExpressionMatch

ExpressionMatch where the expression is true if the user-agent (glob) matches *Safari* and the referrer equals www.example.com.

```
{
  "expression": "req.h.user-agent *= '*Safari*'
      and req.h.referrer == 'www.example.com'"
}
```

4.5.3. ResponseTransform

Adds X-custom-response-header with a value equal to the value of user-agent - host header.

```
{
  "response-transform": {
    "headers": {
      "add": [
        {
          "name": "X-custom-response-header",
          "value": "$req.h.user-agent - $req.h.host",
          "value-is-expression": true
        }
      ],
      "response-status": "403"
    }
  }
}
```

Adds a Set-Cookie header with a dynamically computed cookie value (concatenating user agent and host name) and forces a 403 response.

```
{
   "response-transform":{
      "headers":{
         "add":[
            {
               "name":"Set-Cookie",
               "value":"$req.h.user-agent - $req.h.host",
               "value-is-expression":true
            }
         ]
      }
   }
}
```

4.5.4. MI.ServiceIDs

Extracts the first path element from the URI. For example, if the URI = /789/second/third/test.txt, property-id is set to the first-path (789).

```
{
   "generic-metadata-type":"MI.ServiceIDs",
   "generic-metadata-value":{
      "service-id":"12345",
      "service-name":"My Streaming Service",
      "property-id":"$path_element(req.uri, 1)",
      "property-id-is-expression":true
   }
}
```

5. IANA Considerations

5.1. CDNI Payload Types

TBD. Two new auth-types will be introduced for use with SourceMetadataExtended: AWSv4Auth and HeaderAuth

6. Security Considerations

This specification is in accordance with the CDNI Request Routing: Footprint and Capabilities Semantics. As such, it is subject to the security and privacy considerations as defined in Section 8 of [RFC8006] and in Section 7 of [RFC8008] respectively.

MORE - TBD

7. Acknowledgements

The authors would like to express their gratitude to the members of the Streaming Video Alliance Open Caching Working Group for their guidance / contribution / reviews ...)

8. References

8.1. Normative References

[capacity-insights-advertisement] Ryan, A., Rosenblum, B., and N. Sopher, "CDNI Capacity Insights Capability Advertisment Extensions", 5 July 2021, <https://www.ietf.org/id/draft-ryan-cdni-capacity-insights-extensions-00.txt>.

[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>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[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>.

[RFC8007] Murray, R. and B. Niven-Jenkins, "Content Delivery Network Interconnection (CDNI) Control Interface / Triggers", RFC 8007, DOI 10.17487/RFC8007, December 2016, <https://www.rfc-editor.org/info/rfc8007>.

[RFC8008] Seedorf, J., Peterson, J., Previdi, S., van Brandenburg, R., and K. Ma, "Content Delivery Network Interconnection (CDNI) Request Routing: Footprint and Capabilities Semantics", RFC 8008, DOI 10.17487/RFC8008, December 2016, <https://www.rfc-editor.org/info/rfc8008>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8804] Finkelman, O. and S. Mishra, "Content Delivery Network Interconnection (CDNI) Request Routing Extensions", RFC 8804, DOI 10.17487/RFC8804, September 2020, <https://www.rfc-editor.org/info/rfc8804>.

[URI.signing] van Brandenburg, R., Leung, K., and P. Sorber, "URI Signing for CDN Interconnection (CDNI)", 8 October 2019, <http://www.ietf.org/internet-drafts/draft-ietf-cdni-uri-signing-19.txt>.

[W3C] "Cross-Origin Resource Sharing", <https://www.w3.org/TR/2020/SPSD-cors-20200602/>.

8.2. Informative References

[AWSv4Method] "Authenticating Requests (AWS Signature Version 4)", <https://docs.aws.amazon.com/AmazonS3/latest/API/sig-v4-authenticating-requests.html>.

[OC-CI] Goldstein, G., Ed., Power, W., Bichot, G., and A. Siloniz, "Open Caching - Configuration Interface Functional Specification (Parts 1,2,3)", Version 0.1, 2 July 2021.

[OCWG] "Open Caching Home Page", <https://www.streamingvideoalliance.org/technical-groups/open-caching/>.

[RFC5861] Nottingham, M., "HTTP Cache-Control Extensions for Stale Content", RFC 5861, DOI 10.17487/RFC5861, May 2010, <https://www.rfc-editor.org/info/rfc5861>.

[RFC6707] Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", RFC 6707, DOI 10.17487/RFC6707, September 2012, <https://www.rfc-editor.org/info/rfc6707>.

[RFC7336] Peterson, L., Davie, B., and R. van Brandenburg, Ed., "Framework for Content Distribution Network Interconnection (CDNI)", RFC 7336, DOI 10.17487/RFC7336, August 2014, <https://www.rfc-editor.org/info/rfc7336>.

[RFC7694] Reschke, J., "Hypertext Transfer Protocol (HTTP) Client-Initiated Content-Encoding", RFC 7694, DOI 10.17487/RFC7694, November 2015, <https://www.rfc-editor.org/info/rfc7694>.

[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>.

[SVA] "Streaming Video Alliance Home Page", <https://www.streamingvideoalliance.org>.

Authors' Addresses

Glenn Goldstein
Lumen Technologies
United States of America

Email: glenng1215@gmail.com

Will Power
Lumen Technologies
United States of America

Email: wrpower@gmail.com

Guillaume Bichot
Broadpeak
France

Email: guillaume.bichot@gmail.com

Alfonso Siloniz
Telefonica
Spain

Email: alfonsosiloniz@gmail.com