Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2015

B. Niven-Jenkins
R. Murray
Velocix (Alcatel-Lucent)
M. Caulfield
K. Leung
Cisco Systems
K. Ma
Ericsson

July 2, 2014

CDN Interconnection Metadata
draft-ietf-cdni-metadata-07

Abstract

The CDNI Metadata interface enables interconnected CDNs to exchange content distribution metadata in order to enable content acquisition and delivery. The CDNI metadata associated with a piece of content provides a downstream CDN with sufficient information for the downstream CDN to service content requests on behalf of an upstream CDN. This document describes both a base set of CDNI metadata and the protocol for exchanging that metadata.

Requirements Language

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 RFC 2119 [RFC2119].

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 http://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 January 3, 2015.

Table of Contents

1. Introduction

Content Delivery Networks Interconnection (CDNI) ([RFC6707]) enables a downstream CDN to service content requests on behalf of an upstream CDN.

The CDNI Metadata interface is discussed in [I-D.ietf-cdni-framework] along with four other interfaces that may be used to compose a CDNI solution (CDNI Control interface, CDNI Request Routing Redirection interface, CDNI Footprint & Capabilities Advertisement interface and CDNI Logging interface). [I-D.ietf-cdni-framework] describes each

interface, and the relationships between them. The requirements for the CDNI metadata interface are specified in [I-D.ietf-cdni-requirements].

The CDNI metadata associated with a piece of content (or with a set of content) provides a downstream CDN with sufficient information for servicing content requests on behalf of an upstream CDN in accordance with the policies defined by the upstream CDN.

This document focuses on the CDNI Metadata interface which enables a downstream CDN to obtain CDNI Metadata from an upstream CDN so that the downstream CDN can properly process and respond to:

- Redirection requests received over the CDNI Request Routing Redirection interface.
- Content requests received directly from User Agents.

Specifically, this document specifies:

- A data structure for mapping content requests and redirection requests to CDNI Metadata objects (Section 3 and Section 4.1).
- An initial set of CDNI Generic Metadata objects (Section 4.2).
- A RESTful web service for the transfer of CDNI Metadata (Section 6).

1.1. Terminology

This document reuses the terminology defined in [RFC6707].

Additionally, the following terms are used throughout this document and are defined as follows:

- Object - a collection of properties
- Property - a key and value pair where the key is a property name and the value is the property value or an object.

1.2. Supported Metadata Capabilities

Only the metadata for a small set of initial capabilities is specified in this document. This set provides the minimum amount of metadata for basic CDN interoperability while still meeting the requirements set forth by [I-D.ietf-cdni-requirements].

The following high-level functionality is configured via the metadata described in Section 4:

- Acquisition Source: Metadata for allowing a dCDN to fetch content from a uCDN.
- Delivery Access Control: Metadata for restricting (or permitting) access to content based on any of the following factors:
  - Location
  - Time Window
  - Delivery Protocol
- Delivery Authorization: Metadata for authorizing dCDN user agent requests.
- Cache Control: Metadata for controlling cache behavior of the dCDN.

The metadata encoding described by this document is extensible in order allow for future additions to this list.

This document supports HTTPv1.1 for delivery and both HTTPv1.1 and HTTPv1.1. over TLS for acquisition. All metadata is described in a protocol-agnostic manner.

Supporting unencrypted HTTPv2.0 for delivery (or unencrypted HTTPv2.0 or HTTPv2.0 over TLS for acquisition) only requires the registration of these protocol names in the CDNI Metadata Protocol Sub-Registry.

Supporting HTTPv1.1 over TLS or HTTPv2.0 over TLS for delivery requires specifying additional metadata objects to carry the properties required to establish a TLS session, for example metadata to describe the certificate to present as part of the TLS handshake.

2. Design Principles

The CDNI Metadata interface was designed to achieve the following objectives:

1. Cacheability of CDNI metadata objects.
2. Deterministic mapping from redirection requests and content requests to CDNI metadata properties.

3. Support for DNS redirection as well as application-specific redirection (for example HTTP redirection).

4. Minimal duplication of CDNI metadata.

5. Leveraging of existing protocols.

Cacheability improves the latency of acquiring metadata while maintaining its freshness, and therefore improves the latency of serving content requests and redirection requests, without sacrificing accuracy. The CDNI Metadata interface uses HTTP and its existing caching mechanisms to achieve CDNI metadata cacheability.

Deterministic mappings from content to metadata properties eliminates ambiguity and ensures that policies are applied consistently by all downstream CDNs.

Support for both HTTP and DNS redirection ensures that the CDNI Metadata interface can be used for HTTP and DNS redirection and also meets the same design principles for both HTTP and DNS based redirection schemes.

Minimal duplication of CDNI metadata provides space efficiency on storage in the CDNs, on caches in the network, and across the network between CDNs.

Leveraging existing protocols avoids reinventing common mechanisms such as data structure encoding (e.g. XML, JSON) and data transport (e.g. HTTP).

3. CDNI Metadata Data Model

The CDNI Metadata Model describes a data structure for mapping redirection requests and content requests to metadata properties. Metadata properties describe how to acquire content from an upstream CDN, authorize access to content, and deliver content from a downstream CDN. The data model relies on the assumption that these metadata properties may be aggregated based on the hostname of the content and subsequently on the resource path of the content. The data model associates a set of CDNI Metadata properties with a Hostname to form a default set of metadata properties for content delivered on behalf of that Hostname. That default set of metadata properties can be overridden by properties that apply to specific paths within a URI.

Different Hostnames and URI paths will be associated with different sets of CDNI Metadata properties in order to describe the required behaviour when a dCDN surrogate is processing User Agent requests for

content at that Hostname or URI path. As a result of this structure, significant commonality may exist between the CDNI Metadata properties specified for different Hostnames, different URI paths within a Hostname and different URI paths on different Hostnames. For example the definition of which User Agent IP addresses should be treated as being grouped together into a single network or geographic location is likely to be common for a number of different Hostnames. Another example is that although a uCDN is likely to have several different policies configured to express geo-blocking rules, it is likely that a single geo-blocking policy would be applied to multiple Hostnames delivered through the CDN.

In order to enable the CDNI Metadata for a given Hostname or URI Path to be decomposed into sets of CDNI Metadata properties that can be reused by multiple Hostnames and URI Paths, the CDNI Metadata interface specified in this document splits the CDNI Metadata into a number of objects. Efficiency is improved by enabling a single CDNI Metadata object (that is shared across Hostname and/or URI paths) to be retrieved and stored by a dCDN once, even if it is referenced by the CDNI Metadata of multiple Hostnames or of multiple URI paths.

Section 3.1 introduces a high level description of the HostIndex, HostMetadata and PathMetadata objects and describes the relationships between those objects.

Section 3.2 introduces a high level description of the CDNI GenericMetadata object which represents the level at which CDNI Metadata override occurs between HostMetadata and PathMetadata objects.

Section 4 describes in detail the specific CDNI Metadata objects and properties which may be contained within a CDNI GenericMetadata object.

3.1. HostIndex, HostMetadata and PathMetadata objects

A HostIndex object contains (or references) a list of Hostnames (and/or IP addresses) for which content requests may be delegated to the downstream CDN. The HostIndex is the starting point for accessing the uCDN CDNI Metadata data store. It enables the dCDN to deterministically discover, on receipt of a User Agent request for content, which other CDNI Metadata objects it requires in order to deliver the requested content.

The HostIndex links Hostnames (and/or IP addresses) to HostMetadata objects via HostMatch objects. HostMetadata objects contain (or reference) the default CDNI Metadata required to serve content for that host. When looking up CDNI Metadata, the downstream CDN looks

up the requested Hostname (or IP address) against the HostMatch entries in the HostIndex, from there it can find HostMetadata which describes properties for a host and PathMetadata which may override those properties for given URI paths within the host.

HostMetadata and PathMetadata objects may also contain PathMatch objects which in turn contain PathMetadata objects. PathMatch objects override the CDNI Metadata in the HostMetadata object or one or more preceding PathMetadata objects with more specific CDNI Metadata that applies to content requests matching the pattern defined in that PathMatch object.

For the purposes of retrieving CDNI Metadata, all other required CDNI Metadata objects and their properties are discoverable from the appropriate HostMetadata, PathMatch and PathMetadata objects for the requested content.

The relationships between the HostIndex, HostMatch, HostMetadata, PathMatch and PathMetadata objects are described in Figure 1.

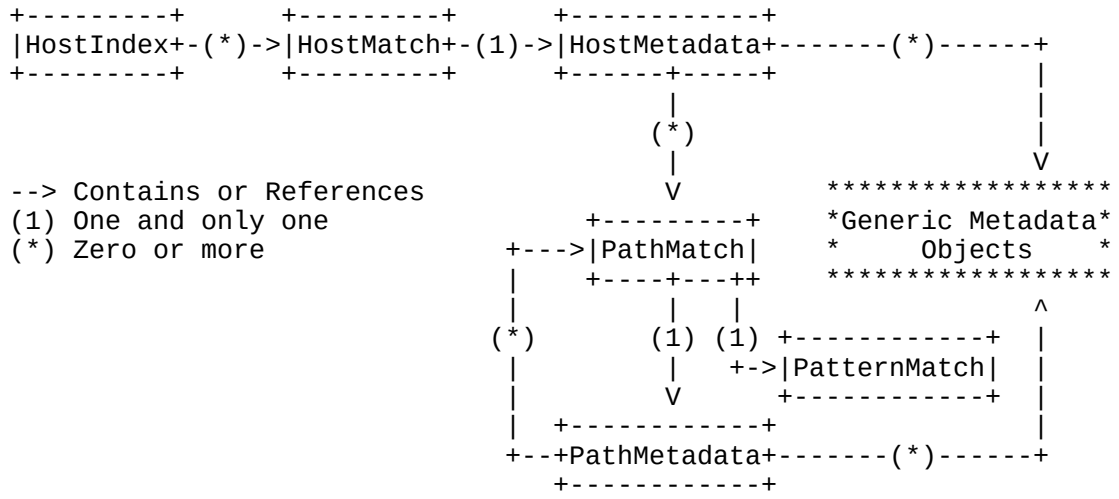

Figure 1: Relationships between CDNI Metadata Objects (Diagram Representation)

The relationships in Figure 1 are also represented in tabular format in Table 1 below.

Data Object	Objects it contains or references
HostIndex	0 or more HostMatch objects.
HostMatch	1 HostMetadata object.
HostMetadata	0 or more PathMatch objects. 0 or more GenericMetadata objects.
PathMatch	1 PatternMatch object. 1 PathMetadata object.
PatternMatch	Does not contain or reference any other objects.
PathMetadata	0 or more PathMatch objects. 0 or more GenericMetadata objects.

Table 1: Relationships between CDNI Metadata Objects (Table Representation)

The table below describes the HostIndex, HostMetadata and PathMetadata objects in more detail.

Data Object	Description
HostIndex	A HostIndex object lists HostMatch objects
HostMatch	A HostMatch object defines a hostname (or IP address) to match against a requested host, and contains (or references) a HostMetadata object which contains (or references) CDNI Metadata objects to be applied when a request matches against the hostname. For example, if "example.com" is a content provider, a HostMatch object may include an entry for "example.com" with the URI of the associated HostMetadata object.
HostMetadata	A HostMetadata object contains (or references) the default CDNI Metadata objects for content served from that host, i.e. the CDNI Metadata objects for content requests that do not match any of the PathMatch objects contained (or referenced) by that HostMetadata object. For example, a HostMetadata object may describe the metadata properties which apply to "example.com" and may contain PathMatches for "example.com/movies/*" and "example.com/music/*" which reference corresponding PathMetadata objects that contain the CDNI Metadata objects for those more specific URI paths.
PathMatch	A PathMatch object defines a pattern (inside a

	PatternMatch object which PathMatch object contains or references) to match against the requested URI path, and contains (or references) a PathMetadata object which contains (or references) the CDNI Metadata objects to be applied when a content request matches against the defined URI path pattern. For example, a PathMatch object may include a PatternMatch object containing a pattern for the path "/movies/*" and may reference a PathMetadata object which contains (or references) the CDNI Metadata for content with that path.
PatternMatch	A PatternMatch object contains the pattern string and flags that describe the URI path that a PathMatch applies to.
PathMetadata	A PathMetadata object contains (or references) the CDNI GenericMetadata objects for content served with the associated URI path (defined in a PathMatch object). A PathMetadata object may also contain (or reference) PathMatch objects in order to recursively define more specific URI paths that require different (e.g. more specific) CDNI Metadata to this one. For example, the PathMetadata object which applies to "example.com/movies/*" may describe CDNI Metadata which apply to that resource path and may contain a PathMatch object for "example.com/movies/hd/*" which would reference the corresponding PathMetadata object for the "example.com/movies/hd/" path prefix.
GenericMetadata	A GenericMetadata object contains (or references) individual CDNI Metadata objects which define the specific policies and attributes needed to properly deliver the associated content. For example, a GenericMetadata object may describe the source from which a CDN may acquire a piece of content. The GenericMetadata object is an atomic unit that may be referenced by HostMetadata and/or PathMetadata objects, but SHOULD NOT contain references to other CDNI Metadata objects. The member objects of a specific CDNI Metadata object MAY be referenced by the GenericMetadata object.

Table 2: HostIndex, HostMetadata and PathMetadata CDNI Metadata Objects

3.2. Generic CDNI Metadata Object Properties

The HostMetadata and PathMetadata objects contain or can reference other CDNI Metadata objects that contain properties which describe how User Agent requests for content should be processed, for example where to acquire the content, authorization rules that should be applied, delivery location restrictions and so on. Each such CDNI Metadata object is a specialization of a CDNI GenericMetadata object. The GenericMetadata object abstracts the basic information required for Metadata override and Metadata distribution, from the specifics of any given property (e.g., property semantics, enforcement options, etc.).

The GenericMetadata object defines the type of properties contained within it as well as whether or not the properties are "mandatory-to-enforce". If the dCDN does not understand or support the property type and the property type is "mandatory-to-enforce", the dCDN MUST NOT serve the content to the User Agent. If the dCDN does not understand or support the property type and the property type is not "mandatory-to-enforce", then that GenericMetadata object may be safely ignored and the dCDN MUST process the content request in accordance with the rest of the CDNI metadata.

Although a CDN MUST NOT serve content to a User Agent if a "mandatory-to-enforce" property cannot be enforced, it may be "safe-to-redistribute" that metadata to another CDN without modification. For example, in the cascaded CDN case, a transit CDN may pass through "mandatory-to-enforce" metadata to a downstream CDN. For Metadata which does not require customization or translation (i.e. metadata that is "safe-to-redistribute"), the data representation received off the wire MAY be stored and redistributed without being natively understood or supported by the transit CDN.However, for Metadata which requires translation, transparent redistribution of the uCDN Metadata values may not be appropriate. Certain Metadata may be safely, though possibly not optimally, redistributed unmodified. For example, source acquisition address may not be optimal if transparently redistributed, but may still work.

Redistribution safety MUST be specified for each GenericMetadata. If a CDN does not understand or support a given GenericMetadata property type and the property type is not "safe-to-redistribute", before redistributing the metadata, the CDN MUST set the "incomprehensible" flag for the GenericMetadata property that it did not understand and was marked as not "safe-to-redistribute". The "incomprehensible" flag signals to a dCDN that the metadata was not properly transformed

by the transit CDN. A CDN MUST NOT attempt to use metadata that has been marked as "incomprehensible" by a uCDN.

[[Editors' Note: Do we need to clarify what is meant by "Redistribution safety MUST be specified"? In Section 4.1.7 (GenericMetadata) we say that StR is not Mandatory-to-Specify and defaults to StR=True. A strict interpretation of "MUST be specified" could be that StR is Mandatory-to-Specify and could lead to dCDNs rejecting requests/metadata that leave it out as the default applies which would be an issue for interop. Maybe change first sentence to "If a GenericMetadata object cannot be redistributed safely then it MUST be marked as not safe-to-redistribute (i.e. Safe-to-redistribute is set to False).]]

Transit CDNs MUST NOT change the value of "mandatory-to-enforce" or "safe-to-redistribute" when propogating metadata to a dCDN. Although a transit CDN may set the value of "incomprehensible" to true, a transit CDN MUST NOT change the value of "incomprehensible" from true to false.

[[Editors' Note: Should a transit CDN be allowed to change the value of "mandatory-to-enforce" or "safe-to-redistribute"? Changing MtE from false to true should be safe from an enforcement perspective as it makes delivery more restrictive? Changing StR may be ok, depending upon what the metadata is (e.g., perhaps URL rewrite is only needed in certain cases and the transit CDN is the one to make that decision)? For simplicity, prohibiting transit CDNs from changing MtE or StR seems like the simplest approach.]]

The following table describes the action to be taken by a transit CDN (tCDN) for the different "mandatory-to-enforce" (MtE) and "safe-to-redistribute" (StR) cases, when the tCDN either does or does not understand the metadata in question:

MtE	StR	Metadata Understood	Actions allowed
False	True	True	Can serve and redistribute.
False	True	False	Can serve and redistribute.
False	False	False	Can serve but MUST set "incomprehensible" to True when redistributing.
False	False	True	Can serve. Can redistribute either by transforming not StR metadata (if the CDN know how to do so safely), otherwise MUST set "incomprehensible" to True when redistributing.
True	True	True	Can serve and can redistribute.
True	True	False	MUST NOT serve but can redistribute.
True	False	True	Can serve and can redistribute.
True	False	False	MUST NOT serve. MUST set "incomprehensible" to True when redistributing.

The following table describes the action to be taken by a dCDN for the different "mandatory-to-enforce" (MtE), "safe-to-redistribute" (StR) and "incomprehensible" (Inc) cases, when the dCDN either does or does not understand the metadata in question:

MtE	StR	Metadata Understood	Incomprehensible	Actions allowed
False	True	True	False	Can serve.
False	True	True	True	Can serve but MUST NOT interpret/apply any metadata marked incomprehensible.
False	True	False	False	Can serve.
False	True	False	True	Can serve but MUST NOT interpret/apply any metadata marked incomprehensible.
False	False	True	False	Can serve.
False	False	True	True	Can serve but

				MUST NOT interpret/apply any metadata marked incomprehensible.
False	False	False	False	Can serve.
False	False	False	True	Can serve but MUST NOT interpret/apply any metadata marked incomprehensible.
True	True	True	False	Can serve.
True	True	True	True	Can serve but MUST NOT interpret/apply any metadata marked incomprehensible.
True	True	False	False	MUST NOT serve.
True	True	False	True	MUST NOT serve.
True	False	True	False	Can serve.
True	False	True	True	Can serve but MUST NOT interpret/apply any metadata marked incomprehensible.
True	False	False	False	MUST NOT serve.
True	False	False	True	MUST NOT serve.

3.3. Metadata Inheritance and Override

In the data model, a HostMetadata object may contain (or reference) multiple PathMetadata objects (via PathMatch objects). Each PathMetadata object may in turn contain (or reference) other PathMetadata objects. HostMetadata and PathMetadata objects form an inheritance tree where each node in the tree inherits or overrides the property values set by its parent.

GenericMetadata objects of a given type override all GenericMetadata objects of the same type previously defined by any parent object in the tree. GenericMetadata objects of a given type previously defined by a parent object in the tree are inherited when no object of the same type is defined by the child object. For example, if HostMetadata for the host "example.com" contains GenericMetadata objects of type LocationACL and TimeWindowACL, while a PathMetadata

object which applies to "example.com/movies/*" defines an alternate GenericMetadata object of type TimeWindowACL, then:

- the TimeWindowACL defined in the PathMetadata would override the TimeWindowACL defined in the HostMetadata for all User Agent requests for content under "example.com/movies", and

- the LocationACL defined in the HostMetadata would be inherited for all User Agent requests for content under "example.com/movies".

4. Encoding-Independent CDNI Metadata Object Descriptions

Section 4.1 provides the definitions of each object type declared in Section 3. These objects are described as structural objects as they provide the structure for the inheritance tree and identifying which specific properties apply to a given User Agent content request.

Section 4.2 provides the definitions for a base set of core metadata objects which may be contained within a GenericMetadata object. These objects are described as property objects as they define the structure, semantics, and enforcement options for specific properties of the metadata being described. Property objects govern how User Agent requests for content are handled. Property objects may be composed of or contain references to other property sub-objects (i.e. property objects contained within or referenced by the property object that refers to that property sub-object). In those cases the value of the property sub-objects can be either a complete serialized representation of the sub-object, or a Link object that contains a URI and relationship that can be dereferenced to retrieve the complete serialized representation of the property sub-object.

Section 6.5 discusses the ability to extend the base set of metadata objects specified in this document with additional standards based or vendor specific property objects that may be defined in the future in separate documents.

Downstream CDNs MUST support parsing of all CDNI metadata objects specified in this document. A dCDN does not have to implement the underlying functionality represented by the metadata object, which may restrict the content which that dCDN can serve. uCDNs as generators of CDNI Metadata only need to support generating the CDNI metadata that they need in order to express the policies and treatment the content they are describing requires.

Note: In the following sections, the term "mandatory-to-specify" is used to convey which property sub-objects MUST be specified for a given structural or property object. When mandatory-to-specify is set to true on a sub-object, it implies that if the property object

containing that property sub-object is specified, then the mandatory-to-specify property sub-object MUST also be specified, e.g., a HostMatch property object without a host to match against does not make sense, therefore, the host is mandatory-to-specify inside a HostMatch property object.

4.1. Descriptions of the CDNI Structural Metadata Objects

Each of the sub-sections below describe the structural objects defined in Table 2.

4.1.1. HostIndex

The HostIndex object is the entry point into the CDNI Metadata hierarchy. It contains (or references) a list of HostMatch objects. An incoming content request is matched against the hostname inside of each of the listed HostMatch objects to find the HostMatch object which applies to the request.

Property: hosts

Description: List of HostMatch objects, in priority order.

Type: List of HostMatch objects

Mandatory-to-Specify: Yes.

4.1.2. HostMatch

The HostMatch object contains a hostname or IP address to match against content requests. The HostMatch object also contains or references a HostMetadata object to apply if a match is found.

Property: host

Description: String (hostname or IP address) to match against the requested host.

Type: String

Mandatory-to-Specify: Yes.

Property: host-metadata

Description: CDNI Metadata to apply when delivering content that matches this host.

Type: HostMetadata

Mandatory-to-Specify: Yes.

4.1.3. HostMetadata

The HostMetadata object contains (or references) both Metadata that applies to content requests for a particular host and a list of pattern matches for finding more specific Metadata based on the resource path in a content request.

Property: metadata

Description: List of host related metadata.

Type: List of GenericMetadata objects

Mandatory-to-Specify: Yes.

Property: paths

Description: Path specific rules. Path patterns (PathMatch objects) MUST be evaluated in the order they appear and the first PathMatch object that matches the content request being process is applied.

Type: List of PathMatch objects

Mandatory-to-Specify: No.

4.1.4. PathMatch

The PathMatch object contains (or references) an expression given as a PatternMatch object to match against a resource URI path and Metadata objects to apply if a match is found.

Property: path-pattern

Description: Pattern to match against the requested path, i.e. against the [RFC3986] path-absolute.

Type: PatternMatch

Mandatory-to-Specify: Yes.

Property: path-metadata

Description: CDNI Metadata to apply when delivering content that matches this pattern.

Type: PathMetadata

Mandatory-to-Specify: Yes.

4.1.5. PathMetadata

A PathMetadata object contains (or reference) the CDNI Metadata properties for content served with the associated URI path (defined in a PathMatch object) and possibly children PathMatch objects.

Note that if DNS-based redirection is employed, then any metadata at the PathMetadata level or below will be inaccessible at request routing time because only the content request hostname is available at request routing time.

Property: metadata

Description: List of path related metadata.

Type: List of GenericMetadata objects

Mandatory-to-Specify: Yes.

Property: paths

Description: Path specific rules. First match applies.

Type: List of PathMatch objects

Mandatory-to-Specify: No.

4.1.6. PatternMatch

A PatternMatch object contains the pattern string and flags that describe the PathMatch expression.

Property: pattern

Description: A pattern for string matching. The pattern may contain the wildcards * and ?, where * matches any sequence of characters (including the empty string) and ? matches exactly one character. The three literals \ , * and ? should be escaped as \\, * and \?. All other characters are treated as literals.

Type: String

Mandatory-to-Specify: Yes.

Property: case-sensitive

Description: Flag indicating whether or not case-sensitive matching should be used.

Type: Boolean

Mandatory-to-Specify: No. Default is case-insensitive match.

Property: ignore-query-string

Description: List of query parameters which should be ignored when searching for a pattern match. If all query parameters should be ignored then the list MUST be empty.

Type: List of String

Mandatory-to-Specify: No. Default is to include query strings when matching.

4.1.7. GenericMetadata

A GenericMetadata object is an abstraction for managing individual CDNI Metadata properties in an opaque manner.

Property: generic-metadata-type

Description: CDNI Metadata property object type.

Type: String

Mandatory-to-Specify: Yes.

Property: generic-metadata-value

Description: CDNI Metadata property object.

Type: Format/Type is defined by the value of generic-metadata-type property above.

Mandatory-to-Specify: Yes.

Property: mandatory-to-enforce

Description: Flag identifying whether or not the enforcement of the property Metadata is required.

Type: Boolean

Mandatory-to-Specify: No. Default is to treat metadata as mandatory to enforce.

Property: safe-to-redistribute

Description: Flag identifying whether or not the property Metadata may be safely redistributed without modification.

Type: Boolean

Mandatory-to-Specify: No. Default is allow transparent redistribution.

4.2. Description of the CDNI Generic Metadata Objects

The property objects defined below are intended to be used in the GenericMetadata object generic-metadata-value field as defined in Section 4.1.7 and their generic-metadata-type property MUST be set to the appropriate Media Type as defined in [[REF]].

[[Editors' note: We don't specify media types for the Generic Metadata object we define anywhere. Need to do that - at a minimum in the IANA section, but should we introduce/explain them elsewhere in the document too?]]

4.2.1. Source Metadata

Source Metadata provides the dCDN information about content acquisition e.g. how to contact an uCDN Surrogate or an Origin Server to obtain the content to be served. The sources are not necessarily the actual Origin Servers operated by the CSP but might be a set of Surrogates in the uCDN.

Endpoints within a source should be treated as equivalent/equal so one can specify a list of sources in preference order and for each source/preference rank one can specify a list of endpoints that are equivalent, e.g. a pool of servers that are not behind a load balancer.

Property: sources

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

Type: List of Source objects

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

4.2.1.1. Source

A Source object describes the Source which should be used by the dCDN for content acquisition, e.g. a Surrogate within the uCDN or an alternate Origin Server, the protocol to be used and any authentication method.

Property: auth

Description: Authentication method to use when requesting content from this source.

Type: Auth

Mandatory-to-Specify: No. Default is no authentication is required.

Property: endpoints

Description: Origins from which the dCDN can acquire content. If multiple endpoints are specified they are all equal, i.e. the list is not in preference order, for example a pool of servers behind a load balancer.

Type: List of EndPoint objects

Mandatory-to-Specify: Yes.

Property: protocol

Description: Network retrieval protocol to use when requesting content from this source.

Type: Protocol

Mandatory-to-Specify: Yes.

4.2.2. LocationACL Metadata

LocationACL Metadata defines location-based restrictions.

A LocationACL which does not include a locations property results in an action of allow, meaning that delivery can be performed regardless of the User Agent’s location. The action from the first footprint to match against the User Agent’s location is the action a CDN MUST take. If two or more footprints overlap, the first footprint that matches against the User Agent’s location determines the action a CDN MUST take. If the locations property is included but is empty, or if

none of the listed footprints matches the User Agent's location then the result is an action of deny.

Property: locations

Description: Description: Access control list which allows or blocks delivery based on client location.

Type: List of LocationRule objects

Mandatory-to-Specify: No. Default is allow all locations.

4.2.2.1. LocationRule

A LocationRule contains or references a list of Footprint objects and the corresponding action.

Property: footprints

Description: List of footprints to which the rule applies.

Type: List of Footprint objects

Mandatory-to-Specify: Yes.

Property: action

Description: Defines whether the rule specifies locations to allow or deny.

Type: Enumeration [allow|deny]

Mandatory-to-Specify: No. Default is deny.

4.2.2.2. Footprint

A Footprint object describes the footprint to which a LocationRule may be applied by, e.g. an IPv4 address range or a geographic location.

Property: footprint-type

Description: Registered footprint type (see Section 7.1.1.1).

Type: String

Mandatory-to-Specify: Yes.

Property: footprint-value

Description: Footprint object conforming to the specification associated with the registered footprint type.

Type: String

Mandatory-to-Specify: Yes.

4.2.3. TimeWindowACL Metadata

TimeWindowACL Metadata defines time-based restrictions.

Property: times

Description: Description: Access control list which allows or blocks delivery based on request time.

Type: List of TimeWindowRule objects

Mandatory-to-Specify: No. Default is allow all time windows.

4.2.3.1. TimeWindowRule

A TimeWindowRule contains or references a list of TimeWindow objects and the corresponding action.

Property: windows

Description: List of time windows to which the rule applies.

Type: List of TimeWindow objects

Mandatory-to-Specify: Yes.

Property: action

Description: Defines whether the rule specifies time windows to allow or deny.

Type: Enumeration [allow|deny]

Mandatory-to-Specify: No. Default is deny.

4.2.3.2. TimeWindow

A TimeWindow object describes a time range which may be applied by an ACLRule, e.g. Start 09:00AM 01/01/2000 UTC End 17:00PM 01/01/2000 UTC.

Property: start

Description: The start time of the window.

Type: Time

Mandatory-to-Specify: Yes.

Property: end

Description: The end time of the window.

Type: Time

Mandatory-to-Specify: Yes.

4.2.4. ProtocolACL Metadata

ProtocolACL Metadata defines delivery protocol restrictions.

Property: protocols

Description: Description: Access control list which allows or blocks delivery based on delivery protocol.

Type: List of ProtocolRule objects

Mandatory-to-Specify: No. Default is allow all protocols.

4.2.4.1. ProtocolRule

A ProtocolRule contains or references a list of Protocol objects. ProtocolRule objects are used to construct a ProtocolACL to apply restrictions to content acquisition or delivery.

Property: protocols

Description: List of protocols to which the rule applies.

Type: List of protocol objects

Mandatory-to-Specify: Yes.

Property: action

Description: Defines whether the rule specifies protocols to allow or deny.

Type: Enumeration [allow|deny]

Mandatory-to-Specify: No. Default is allow all protocols.

4.2.5. Authorization Metadata

Authorization Metadata define content authorization methods.

Property: methods

Description: Options for authenticating content requests. All options in the list are equally valid.

Type: List of Auth objects

Mandatory-to-Specify: No. Default is no authorization required.

4.2.6. Auth

An Auth object defines authentication and authorization methods to be used during content delivery and content acquisition.

Property: auth-type

Description: Registered Auth type (see Section 7.1.1.3).

Type: String

Mandatory-to-Specify: Yes.

Property: auth-value

Description: Auth object conforming to the specification associated with the registered Auth type.

Type: String

Mandatory-to-Specify: Yes.

4.2.6.1. Credentials Auth Type

Credentials Auth is a type of Auth object with type "credentials" (see Section 7.1.1.3). The CredentialsAuth object contains the following properties:

Property: username

Description: Identification of user.

Type: String

Mandatory-to-Specify: Yes.

Property: password

Description: Password for user identified by username property.

Type: String

Mandatory-to-Specify: Yes.

4.2.7. Cache

A Cache object describes the cache control parameters to be applied to the content by intermediate caches.

Property: ignore-query-string

Description: Allows a cache to ignore URI query string parameters while comparing URIs for equivalence. Each query parameter to ignore is specified in the list. If all query parameters should be ignored, then the list MUST be empty.

Type: List of String

Mandatory-to-Specify: No. Default is to consider query string parameters when comparing URIs.

4.2.8. Grouping

A Grouping object identifies a large group of content to which this content belongs.

Property: ccid

Description: Content Collection identifier for an application-specific purpose such as logging.

Type: String

Mandatory-to-Specify: No. Default is an empty string.

Property: sid

Description: Session identifier for an application-specific purpose such as logging.

Type: String

Mandatory-to-Specify: No. Default is an empty string.

4.3. CDNI Metadata Simple Data Type Descriptions

This section describes the simpler data types that are used for properties of CDNI Metadata objects.

4.3.1. Link

A link object may be used in place of any of the objects or properties described above. Links can be used to avoid duplication if the same metadata information is repeated within the metadata tree. When a link replaces an object, its href property is set to the URI of the resource and its type property is set to the type of the object it is replacing.

Property: href

Description: The URI of the of the addressable object being referenced.

Type: URI

Mandatory-to-Specify: Yes

Property: metadata-object-type

Description: The type of the object being referenced.

Type: String

Mandatory-to-Specify: No

4.3.2. Protocol

Protocol objects are used to specify registered protocols for content acquisition or delivery (see Section 7.1.1.2).

Type: String

4.3.3. Endpoint

A hostname (with optional port) or an IP address (with optional port).

Note: All implementations MUST support IPv4 addresses encoded as specified by the 'IPv4address' rule in Section 3.2.2 of [RFC3986] and MUST support all IPv6 address formats specified in [RFC4291]. Server implementations SHOULD use IPv6 address formats specified in [RFC5952].

Type: String

4.3.4. URI

A URI as specified in [RFC3986].

Type: String

4.3.5. Time

A time value expressed in seconds since Unix epoch in the UTC timezone.

Type: Integer

5. CDNI Metadata Capabilities

CDNI Metadata is used to convey information pertaining to content delivery from uCDN to dCDN. For optional metadata, it may be useful for the uCDN to know if the dCDN supports the metadata, prior to delegating any content requests to the dCDN. If optional-to-implement metadata is "mandatory-to-enforce" and the dCDN does not support it, any delegated requests for that content will fail, so the uCDN is likely to want to avoid delegating those requests to that dCDN. Likewise, for any metadata which may be assigned optional values, it may be useful for the uCDN to know which values the dCDN supports, prior to delegating any content requests to the dCDN. If the optional value assigned to a given piece of content's metadata is not supported by the dCDN, any delegated requests for that content

may fail, so again the uCDN is likely to want to avoid delegating those requests to that dCDN.

The CDNI Footprint and Capabilities Interface (FCI) [I-D.ietf-cdni-framework] provides a means of advertising capabilities from dCDN to uCDN. Support for optional metadata and support for optional metadata values may be advertised using the FCI.

This section describes the capabilities advertisement requirements for the metadata defined in this document.

5.1. Protocol ACL Capabilities

The ProtoclACL object contains a list of Protocol values. The dCDN MUST advertise which delivery protocols it supports so that the uCDN knows what type of content requests it can redirect to the dCDN. If the dCDN does not support a given acquisition or delivery protocol, the uCDN should not delegate requests requiring those protocols to the dCDN as the dCDN will not be able to properly acquire or deliver the content.

ProtocolRules are defined for either acquisition or delivery. For some CDNs, certain combinations of acquisition and delivery protocols may not make sense (e.g., RTSP acquisition for HTTP delivery), while other CDNs may support customized protocol adaptation. ProtocolACL capabilities are not intended to define which combinations of protocols should be used. ProtocolACL capabilties are only intended to describe which protocols the dCDN does or does not support. Protocol combination restrictions are specified in the metadata itself and associated with specific groups of content assets.

5.2. Authorization Metadata Capabilities

The Authorization object contains a list of Auth values. The dCDN MUST advertise in its FCI capabilities which authorization types it supports.

The definition of the corresponding capabilities and the protocol used to advertise them are outside the scope of this document and are expected to be specified as part of the CDNI Footprint and Capabilities Interface.

6. CDNI Metadata interface

This section specifies an interface to enable a Downstream CDN to retrieve CDNI Metadata objects from an Upstream CDN.

The interface can be used by a Downstream CDN to retrieve CDNI Metadata objects either

- Dynamically as required by the Downstream CDN to process received requests. For example in response to a query from an Upstream CDN over the CDNI Request Routing Redirection interface (RI) [I-D.ietf-cdni-redirection] or in response to receiving a request for content from a User Agent. Or;

- In advance of being required. For example in the case of Pre-positioned CDNI Metadata acquisition.

The CDNI Metadata interface is built on the principles of RESTful web services. In particular, this means that requests and responses over the interface are built around the transfer of representations of hyperlinked resources. A resource in the context of the CDNI Metadata interface is any object in the Data Model (as described in Section 3 through Section 4).

To retrieve CDNI metadata, a CDNI Metadata client (i.e. a client in the dCDN) first makes a HTTP GET request for the URI of the HostIndex which provides the CDNI Metadata client with a list of Hostnames for which the upstream CDN may delegate content delivery to the downstream CDN. The CDNI Metadata client can then obtain any other CDNI Metadata objects by making a HTTP GET requests for any linked Metadats objects it requires.

CDNI Metadata servers (i.e. servers in the uCDN) are free to assign whatever structure they desire to the URIs for CDNI Metadata objects and CDNI Metadata clients MUST NOT make any assumptions regarding the structure of CDNI Metadata URIs or the mapping between CDNI Metadata objects and their associated URIs. Therefore any URIs present in the examples below are purely illustrative and are not intended to impose a definitive structure on CDNI Metadata interface implementations.

6.1. Transport

The CDNI Metadata interface uses HTTP as the underlying protocol transport.

The HTTP Method in the request defines the operation the request would like to perform. A server implementation of the CDNI Metadata interface MUST support the HTTP GET and HEAD methods.

The corresponding HTTP Response returns the status of the operation in the HTTP Status Code and returns the current representation of the resource (if appropriate) in the Response Body. HTTP Responses from servers implementing the CDNI Metadata interface that contain a

response body SHOULD include an ETag to enable validation of cached versions of returned resources.

The CDNI Metadata interface specified in this document is a read-only interface. Therefore support for other HTTP methods such as PUT, POST and DELETE etc. is not specified. An server implementation of the CDNI Metadata interface SHOULD reject all methods other than GET and HEAD.

As the CDNI Metadata interface builds on top of HTTP, CDNI Metadata server implementations MAY make use of any HTTP feature when implementing the CDNI Metadata interface, for example a CDNI Metadata server MAY make use of HTTP's caching mechanisms to indicate that the returned response/representation can be reused without re-contacting the CDNI Metadata server.

6.2. Retrieval of CDNI Metadata resources

In the general case a CDNI Metadata server makes each instance of an addressable CDNI Metadata object available via a unique URI and therefore in order to retrieve CDNI Metadata, a CDNI Metadata client first makes a HTTP GET request for the URI of the HostIndex which provides the CDNI Metadata client with a list of Hostnames for which the upstream CDN may delegate content delivery to the downstream CDN.

In order to retrieve the CDNI Metadata for a particular request the CDNI Metadata client processes the received HostIndex object and finds the corresponding HostMetadata entry (by matching the hostname in the request against the hostnames in the HostMatch). If the HostMetadata is linked (rather than embedded), the CDNI metadata client then makes a GET request for the URI specified in the href property of the Link object which points to the HostMetadata object itself.

In order to retrieve the most specific metadata for a particular request, the CDNI metadata client inspects the HostMetadata for references to more specific PathMetadata objects. If any PathMetadata match the request (and are linked rather than embedded), the CDNI metadata client makes another GET request for the PathMetadata. Each PathMetadata object may also include references to yet more specific metadata. If this is the case, the CDNI metadata client continues requesting PathMetadata recursively.

Where a downstream CDN is interconnected with multiple upstream CDNs, the downstream CDN needs to determine which upstream CDN's CDNI metadata should be used to handle a particular User Agent request.

When application level redirection (e.g. HTTP 302 redirects) is being used between CDNs, it is expected that the downstream CDN will be able to determine the upstream CDN that redirected a particular request from information contained in the received request (e.g. via the URI). With knowledge of which upstream CDN routed the request, the downstream CDN can choose the correct metadata server from which to obtain the HostIndex. Note that the HostIndex served by each uCDN may be unique.

In the case of DNS redirection there is not always sufficient information carried in the DNS request from User Agents to determine the upstream CDN that redirected a particular request (e.g. when content from a given host is redirected to a given downstream CDN by more than one upstream CDN) and therefore downstream CDNs may have to apply local policy when deciding which upstream CDN’s metadata to apply.

6.3. Bootstrapping

The URI for the HostIndex object of a given upstream CDN needs to be either configured in, or discovered by, the downstream CDN. All other objects/resources are then discoverable from the HostIndex object by following the links in the HostIndex object and the referenced HostMetadata and PathMetadata objects.

If the URI for the HostIndex object is not manually configured in the downstream CDN then the HostIndex URI could be discovered. A mechanism allowing the downstream CDN to discover the URI of the HostIndex is outside the scope of this document.

6.4. Encoding

Object are resources that may be:

- Addressable, where the object is a resource that may be retrieved or referenced via its own URI.

- Embedded, where the object is contained within a property of an addressable object.

In the descriptions of objects we use the term "X contains Y" to mean that Y is either directly embedded in X or is linked to by X. It is generally a deployment choice for the uCDN implementation to decide when and which CDNI Metadata objects to embed and which are made separately addressable.

6.4.1. MIME Media Types

All MIME types for CDNI Metadata objects are prefixed with "application/cdni.". The MIME type for each object then contains the object name of that object as defined by this document. The object type name is followed by ".v" and the version number of the object type (e.g. ".v1"). Finally, the encoding type "+json" is appended. Table 3 3 lists a few examples of the MIME Media Type for some object (resource) that are retrievable through the CDNI Metadata interface.

Data Object	MIME Media Type
HostIndex	application/cdni.HostIndex.v1+json
HostMatch	application/cdni.HostMatch.v1+json
HostMetadata	application/cdni.HostMetadata.v1+json
PathMatch	application/cdni.PathMatch.v1+json
PathMetadata	application/cdni.PathMetadata.v1+json
Source	application/cdni.Source.v1+json
LocationACL	application/cdni.LocationACL.v1+json
LocationRule	application/cdni.LocationRule.v1+json

Table 3: Example MIME Media Types for CDNI Metadata objects

6.4.2. JSON Encoding of Objects

A CDNI Metadata objects is encoded as a JSON object containing a dictionary of (key,value) pairs where the keys are the property names and the values are the associated property values.

The keys of the dictionary are the names of the properties associated with the object and are therefore dependent on the specific object being encoded (i.e. dependent on the MIME Media Type of the returned resource). Likewise, the values associated with each key are dependent on the specific object being encoded (i.e. dependent on the MIME Media Type of the returned resource).

Dictionary keys in JSON are case sensitive. By convention any dictionary key defined by this document (for example the names of CDNI Metadata object properties) MUST be represented in lowercase.

In addition to the properties specified for each object type, the keys defined below may be present in any object.

Key: base

Description: Provides a prefix for any relative URLs in the object. This is similar to the XML base tag [XML-BASE]. If absent, all URLs in the remainder of the response MUST be absolute URLs.

Type: URI

Mandatory: No

Key: _links

Description: The links from this object to other addressable objects. Any property whose value is an object may be replaced by a link to an object with the same type as the property it replaces. The keys of the _links dictionary are the names of the properties being replaced. The values of the dictionary are Link objects with href set to the URI of the object and type set to the MIME type of the object being replaced.

Type: Dictionary object of Link objects

Mandatory: Yes

6.4.2.1. Encoded CDNI Metadata Example

[[Editor’s Note: We need to double-check that the example(s) below are correct and use the latest property names/values/structures defined in the document.]]

A downstream CDN may request the HostIndex and receive the following object of type "application/cdni.HostIndex.v1+json":

```
   {
     "hosts": [
       {
         "host": "video.example.com",
         "_links": {
           "host-metadata" : {
             "type": "application/cdni.HostMetadata.v1+json",
             "href": "http://metadata.ucdn.example/host1234"
           }
         }
       },
       {
         "host": "images.example.com",
         "_links": {
           "host-metadata" : {
             "type": "application/cdni.HostMetadata.v1+json",
             "href": "http://metadata.ucdn.example/host5678"
           }
         }
       }
     ]
   }
```

If the incoming request has a Host header with "video.example.com" then the downstream CDN would fetch the next metadata object from "http://metadata.ucdn.example/host1234" expecting a MIME type of "application/cdni.HostMetadata.v1+json":

```
{
  "metadata": [
    {
      "generic-metadata-type": "application/cdni.SourceMetadata.v1+json",
      "generic-metadata-value": {
        "sources": [
          {
            "_links": {
              "auth": {
                "auth-type": "application/cdni.Auth.v1+json",
                "href": "http://metadata.ucdn.example/auth1234"
              }
            },
            "endpoint": "acq1.ucdn.example",
            "protocol": "ftp"
          },
          {
            "_links": {
              "auth": {
                "auth-type": "application/cdni.Auth.v1+json",
```

```
                "href": "http://metadata.ucdn.example/auth1234"
              }
            },
            "endpoint": "acq2.ucdn.example",
            "protocol": "http"
          }
        ]
      }
    },
    {
      "generic-metadata-type": "application/cdni.LocationACL.v1+json",
      "generic-metadata-value": {
        "locations": [
          {
            "locations": [
              {
                "footprint-type": "IPv4CIDR",
                "footprint-value": "192.168.0.0/16"
              }
            ],
            "action": "deny"
          }
        ]
      }
    },
    {
      "generic-metadata-type": "application/cdni.ProtocolACL.v1+json",
      "generic-metadata-value": {
        "protocols": [
          {
            "protocols": [
              "ftp"
            ],
            "action": "deny"
          }
        ]
      }
    }
  ],
  "paths": [
    {
      "path-pattern": {
        "pattern": "/video/trailers/*"
      },
      "_links": {
        "path-metadata": {
          "type": "application/cdni.PathMetadata.v1+json",
          "href": "http://metadata.ucdn.example/host1234/pathABC"
```

```
        }
      }
    },
    {
      "path-pattern": {
        "pattern": "/video/movies/*"
      },
      "_links": {
        "path-metadata": {
          "type": "application/cdni.PathMetadata.v1+json",
          "href": "http://metadata.ucdn.example/host1234/pathDCE"
        }
      }
    }
  ]
}
```

Suppose the path of the requested resource matches the "/video/movies/*" pattern, the next metadata requested would be for "http://metadata.ucdn.example/host1234/movies" with an expected type of "application/cdni.PathMetadata.v1+json":

```
{
  "metadata": [],
  "paths": [
    {
      "path-pattern": {
        "pattern": "/videos/movies/hd/*"
      },
      "_links": {
        "pathmetadata": {
          "type": "application/cdni.PathMetadata.v1+json",
          "href":
            "http://metadata.ucdn.example/host1234/pathABC/path123"
        }
      }
    }
  ]
}
```

Finally, if the path of the requested resource also matches the "/videos/movies/hd/*" pattern, the downstream CDN would also fetch the following object from "http://metadata.ucdn.example/host1234/movies/hd" with MIME type "application/cdni.PathMetadata.v1+json":

```
{
  "metadata": [
    {
      "generic-metadata-type": "application/cdni.TimeWindowACL.v1+json",
      "generic-metadata-value": {
        "times": [
          "windows": [
            {
              "start": "1213948800",
              "end": "1327393200"
            }
          ],
          "action": "allow"
        ]
      }
    }
  ]
}
```

6.5. Extensibility

The set of property Metadata may be extended with additional (standards based or vendor specific) property Metadata. The GenericMetadata object defined in Section 4.1.7 allows any Metadata property to be included in either the HostMetadata or PathMetadata lists. It is expected that additional property Metadata will be defined in the future and that the documents defining those property Metadata will be registered in the CDNI GenericMetadata Types registry Section 7.1.

Note: Identification, within the type name defined for a property Metadata object, of the organization that defined the extension property Metadata decreases the possibility of property Metadata type collisions.

6.5.1. Metadata Enforcement

At any given time, the set of property Metadata supported by the uCDN may not match the set of property Metadata supported by the dCDN. The uCDN may or may not know which property Metadata the dCDN supports.

In the cases where the uCDN supports Metadata that the dCDN does not, the dCDN MUST enforce the semantics of the "mandatory-to-enforce" property. If a dCDN does not understand or is unable to perform the functions associated with any "mandatory-to-enforce" Metadata, the dCDN MUST NOT service any requests for the corresponding content.

Any specification which defines a new GenericMetadata Type MUST also define whether or not the new metadata is "mandatory-to-enforce" and whether or not it is "safe-to-distribute".

Note: Ideally, uCDNs would not delegate content requests to a dCDN which does not support the "mandatory-to-enforce" Metadata associated with the content being requested. However, even if the uCDN has a priori knowledge of the Metadata supported by the dCDN (e.g., via the CDNI capabilities interface or through out-of-band negotiation between CDN operators) Metadata support may fluctuate or be inconsistent (e.g., due to mis-communication, mis-configuration, or temporary outage). Thus, the dCDN MUST always evaluate all Metadata associated with content requests and reject any requests where "mandatory-to-enforce" Metadata associated with the content cannot be enforced.

6.5.2. Metadata Override

It is possible that new Metadata definitions may obsolete or override existing property Metadata (e.g., a future revision of the CDNI Metadata interface may redefine the Auth Metadata or a custom vendor extension may implement an alternate Auth Metadata option). If multiple Metadata (e.g., cdni.v2.Auth, vendor1.Auth, and vendor2.Auth) all override an existing Metadata (e.g., cdni.Auth) and all are marked as "mandatory-to-enforce", it may be ambiguous which Metadata should be applied, especially if the functionality of the Metadata conflict.

As described in Section 3.3, Metadata override only applies to Metadata objects of the same exact type, found in HostMetadata and nested PathMetadata structures. The CDNI Metadata interface does not support enforcement of dependencies between different Metadata types. It is the responsibility of the CSP and the CDN operators to ensure that Metadata assigned to a given content do not conflict.

Note: Because Metadata is inherently ordered in GenericMetadata lists, as well as in the PathMetadata hierarchy and PathMatch lists, multiple conflicting Metadata types MAY be used, however, Metadata hierarchies MUST ensure that independent PathMatch root objects are used to prevent ambiguous or conflicting Metadata definitions.

6.6. Versioning

The version of CDNI Metadata Structural objects is conveyed inside the MIME-Type that is included in the HTTP Content-Type header. Upon responding to a request for an object, a metadata server MUST include a Content-Type header with the MIME-type containing the version number of the object. HTTP requests sent to a metadata server SHOULD

include an Accept header with the MIME-type (which includes the version) of the expected object. Metadata clients can specify multiple MIME-types in the Accept header, for example if a metadata client is capable of processing two different versions of the same type of object (defined by different MIME-types) it may decide to include both in the Accept header. The version of each object defined by this document is version 1. For example: "Content-Type: application/cdni.HostIndex.v1+json".

GenericMetadata objects include a "type" property which specifies the MIME-type of the GenericMetadata value. This MIME-type should also include a version. Any document which defines a new type of GenericMetadata MUST specify the version number which it describes. For example: "application/cdni.Location.v1+json".

7. IANA Considerations

This document requests the registration of the prefix "application/cdni" MIME Media Type under the IANA MIME Media Type registry (http://www.iana.org/assignments/media-types/index.html).

7.1. GenericMetadata Type Registry

CDNI Metadata is distributed as a list of GenericMetadata objects which specify a type field and a type-specific value field, as described in Section 4.1.7. In order to prevent namespace collisions for GenericMetadata object types a new IANA registry is requested for the "CDNI GenericMetadata Types" namespace. The namespace shall be split into two partitions: standard and optional.

The standard namespace partition is intended to contain mandatory to implement capabilities and conforms to the "IETF Review" policy as defined in [RFC5226]. The registry contains the generic metadata type name, the RFC number of the specification defining the metadata type, the version number of the GenericMetadata set to which the standard capability applies, and boolean values indicating whether or not the new type is considered "mandatory-to-enforce" or "safe-to-redistribute" (as defined in Section 4.1.7).

The following table defines the initial values for the standard partition:

Type name	Specification	Version	MTE	STR
SourceMetadata	RFCthis	1	true	true
LocationACL	RFCthis	1	true	true
TimeWindowACL	RFCthis	1	true	true
ProtocolACL	RFCthis	1	true	true
Auth	RFCthis	1	true	true
Cache	RFCthis	1	true	true
Grouping	RFCthis	1	true	true

The initial MI version number is set to 1. All of the initial GenericMetadata types are considered mandatory to implement for version 1. The version field should be incremented when new GenericMetadata type sets are added to the registry.

The "optional" namespace partition conforms to the "Expert Review" policy as defined in [RFC5226]. The expert review is intended to prevent namespace hoarding and to prevent the definition of redundant GenericMetadata types. Vendors defining new GenericMetadata types which conflict with existing GenericMetadata types follow the guidelines for the "Specification Required" policy as defined in [RFC5226]. The Version field in the registry is set to "-1" (negative one) for non-standard GenericMetadata types.

As with the initial GenericMetadata types defined in Section 4.2, future GenericMetadata type registrations will specify the information necessary for constructing and decoding the GenericMetadata object. This information includes the list of properties contained within the GenericMetadata object, and for each property, the specification should include a description, a type, and whether or not the given property is mandatory to specify.

Any document which defines a new GenericMetadata has to:

1. Allocate a new type in the GenericMetadata Type Registry (Section 7). Generic Metadata types should be descriptive and may be hierarchnical to support aggregating groups of properties for the purpose of readability and for avoiding conflicts between vendor defined extensions. A dotted alpha-numeric notation is suggested for human readability.

2. Define the set of properties associated with the new type.

3. For each property, define a name, description, type, and whether or not the property is mandatory-to-specify.

4. Specify whether or not the new type is "mandatory-to-enforce" (vs optional-to-enforce).

5. Describe the semantics of the new type including its purpose and example of a use case to which it applies.

7.1.1. GenericMetadata Sub-Registries

Some of the initial standard GenericMetadata objects contain enumerated types which require registration (i.e., LocationACL footprint types, ProtocolACL protocols, and Auth protocols). The following sections define the initial values for these GenericMetadata type sub-registries.

7.1.1.1. Footprint Sub-Registry

The "CDNI Metadata Footprint Types" namespace defines the valid Footprint object type values used by the Footprint object in Section 4.2.2.2. Additions to the Footprint type namespace conform to the "Expert Review" policy as defined in [RFC5226]. The expert review should verify that new type definitions do not duplicate existing type definitions and prevent gratuitous additions to the namespace.

The following table defines the initial Footprint type values:

Type name	Description	Specification
IPv4CIDR	IPv4 address block using slash prefix length notation (e.g., 192.168.0.16/28). Single IP addresses can be expressed as /32.	RFCthis
IPv6CIDR	IPv6 address block using slash prefix length notation (e.g., fc80::0010/124). Single IP addresses can be expressed as /128.	RFCthis
ASN	Autonomous System (AS) Number	RFCthis
CountryCode	ISO 3166-1 alpha-2 code	RFCthis

7.1.1.2. Protocol Sub-Registry

The "CDNI Metadata Protocols" namespace defines the valid Protocol object values in Section 4.3.2, used by the SourceMetadata and ProtocolACL objects. Additions to the Protocol namespace conform to the "Expert Review" policy as defined in [RFC5226]. The expert review should verify that new type definitions do not duplicate

existing type definitions and prevent gratuitous additions to the namespace.

The following table defines the initial Protocol values:

Protocol	Description	Specification
HTTP	Hypertext Transfer Protocol -- HTTP/1.1	RFC2616
HTTPS	HTTP Over TLS	RFC2818
RTSP	Real Time Streaming Protocol	RFC2326
RTMP	Real-Time Messaging Protocol	http://www.adobe.com/devnet/rtmp.html
FTP	FILE TRANSFER PROTOCOL	RFC959
SFTP	SSH File Transfer Protocol	N/A
SCP	Secure Copy	N/A
fasp	Aspera fast, adaptive, secure protocol	N/A

7.1.1.3. Authentication Sub-Registry

The "CDNI Metadata Auth" namespace defines the valid Auth object types used by the Auth object in Section 4.2.6. Additions to the Auth namespace conform to the "Expert Review" policy as defined in [RFC5226]. The expert review should verify that new type definitions do not duplicate existing type definitions and prevent gratuitous additions to the namespace.

The following table defines the initial Auth type values:

Type name	Description	Specification
credentials	Simple username and password authentication as defined by Section 4.2.6.1.	RFCthis

8. Security Considerations

The CDNI Metadata interface is expected to be secured as a function of the transport protocol (e.g. HTTP authentication [RFC2617], HTTPS [RFC2818], or inter-domain IPSec).

If a malicious metadata server is contacted by a downstream CDN, the malicious server may provide metadata to the downstream CDN which denies service for any piece of content to any user agent. The malicious server may also provide metadata which directs a downstream CDN to a malicious origin server instead of the actual origin server. The dCDN is expected to authenticate the server to prevent this situation (e.g. by using HTTPS and validating the server's certificate).

A malicious metadata client could request metadata for a piece of content from an upstream CDN. The metadata information may then be used to glean information regarding the uCDN or to contact an upstream origin server. The uCDN is expected to authenticate client requests to prevent this situation.

9. Acknowledgements

The authors would like to thank David Ferguson and Francois Le Faucheur for their valuable comments and input to this document.

10. Contributing Authors

[RFC Editor Note: Please move the contents of this section to the Authors' Addresses section prior to publication as an RFC.]

Grant Watson
Velocix (Alcatel-Lucent)
3 Ely Road
Milton, Cambridge CB24 6AA
UK

Email: gwatson@velocix.com

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, August 2010.

11.2. Informative References

[I-D.ietf-cdni-framework] Peterson, L., Davie, B., and R. Brandenburg, "Framework for CDN Interconnection", draft-ietf-cdni-framework-14 (work in progress), June 2014.

[I-D.ietf-cdni-redirection] Niven-Jenkins, B. and R. Brandenburg, "Request Routing Redirection Interface for CDN Interconnection", draft-ietf-cdni-redirection-02 (work in progress), April 2014.

[I-D.ietf-cdni-requirements] Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", draft-ietf-cdni-requirements-17 (work in progress), January 2014.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC6707] Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", RFC 6707, September 2012.

[XML-BASE]
Marsh, J., Ed. and R. Tobin, Ed., "XML Base (Second Edition) - http://www.w3.org/TR/xmlbase/", January 2009.

Authors' Addresses

Ben Niven-Jenkins
Velocix (Alcatel-Lucent)
3 Ely Road
Milton, Cambridge CB24 6AA
UK

Email: ben@velocix.com

Rob Murray
Velocix (Alcatel-Lucent)
3 Ely Road
Milton, Cambridge CB24 6AA
UK

Email: rmurray@velocix.com

Matt Caulfield
Cisco Systems
1414 Massachusetts Avenue
Boxborough, MA 01719
USA

Phone: +1 978 936 9307
Email: mcaulfie@cisco.com

Kent Leung
Cisco Systems
3625 Cisco Way
San Jose 95134
USA

Phone: +1 408 526 5030
Email: kleung@cisco.com

Kevin J. Ma
Ericsson
43 Nagog Park
Acton, MA 01720
USA

Phone: +1 978-844-5100
Email: kevin.j.ma@ericsson.com