Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 16, 2014

B. Niven-Jenkins
R. Murray
G. Watson
Velocix (Alcatel-Lucent)
M. Caulfield
K. Leung
Cisco Systems
K. Ma
Azuki Systems, Inc.
July 15, 2013

CDN Interconnect Metadata
draft-ietf-cdni-metadata-02

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 the core 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

Copyright Notice

Table of Contents

1. Introduction

CDNI enables a downstream CDN to service content requests on behalf of an upstream CDN. The CDNI metadata associated with a piece of content (or with a set of contents) 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.

The CDNI Metadata Interface is introduced by [RFC6707] along with three other interfaces that may be used to compose a CDNI solution (Control, Request Routing and Logging). [I-D.ietf-cdni-framework] expands on the information provided in [RFC6707] and describes each interface, and the relationships between them, in more detail. The requirements for the CDNI metadata interface are specified in [I-D.ietf-cdni-requirements].

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 protocol.

- Content Requests received directly from User Agents.

Specifically this document proposes:

- A data structure for mapping content requests to CDNI Metadata properties (Section 3).

- An initial set of CDNI Metadata properties (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.

2. Design Principles

The proposed CDNI Metadata Interface was designed to achieve the following objectives:

1. Cacheability of CDNI metadata objects

2. Deterministic mapping from redirection 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. Leverage existing protocols

Cacheability improves the latency of acquiring metadata while maintaining its freshness and therefore improves the latency of serving content requests. The CDNI Metadata Interface uses HTTP to achieve 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, authorize, 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 for 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 by a dCDN once, even if it is referenced by the CDNI Metadata of multiple Hostnames.

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 & PathMetadata objects

A HostIndex object contains 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's CDNI Metadata data store. It enables surrogates in 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) 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.

As well as containing the default CDNI Metadata for the specified Hostname, 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.

```
   +---------+      +---------+      +------------+
   |HostIndex+-(*)->|HostMatch|-(1)->|HostMetadata+-------(*)------+
   +---------+      +---------+      +------+-----+                |
                                            |                      |
                                           (*)                     |
                                            |                      |
   --> References                           V                      V
   (1) One and only one                +---------+
*************************
   (*) Zero or more               +--->|PathMatch|    *Generic Metadata
Objects*
                                  |    +---------+
*************************
                                  |         |                      ^
                                 (*)       (1)                     |
                                  |         |                      |
                                  |         V                      |
                                  |  +------------+                |
                                  +--|PathMetadata+-------(*)------+
                                     +------------+
```

Key: ----> = References

Figure 1: Relationships between the HostIndex, HostMetadata & PathMetadata CDNI Metadata Objects

The relationships in Figure 1 are summarised in Table 1 below.

Data Object	Objects it References

HostIndex	0 or more HostMatch objects.

HostMatch	1 HostMetadata object.
HostMetadata	0 or more PathMatch objects. 0 or more GenericMetadata objects.
PathMatch	1 PathMetadata object.
PathMetadata	0 or more PathMatch objects. 0 or more GenericMetadata objects.

Table 1: Relationships between CDNI Metadata Objects

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 to match against a requested host, and contains or references a HostMetadata object which contains 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 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.
PathMetadata	A PathMetadata object contains the CDNI GenericMetadata objects for content served with

	the associated URI path (defined in a PathMatch object). A PathMetadata object may also contain 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 individual CDNI Metadata objects which define the specific policies and attributes needed to properly deliver the associated content.

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 opaque 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 it is also not going to be able to properly propagate the Metadata for cascaded distribution. If the dCDN does not understand or support the property type and the property type is not mandatory to enforce, then the GenericMetadata object may be safely ignored.

Although a CDN cannot serve content to a User Agent if a mandatory 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 the delivery CDN. For Metadata which does not require customization, 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 require translations, transparent redistribution of the uCDN Metadata values may not be appropriate. Certain Metadata may be safely, though possibly not optimially, redistributed unmodified, e.g., source acquisition address may not be optimal if transparently redistributed, but may still work. Redistribution safety MUST be specified for each GenericMetadata.

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

The PathMetadata defined TimeWindowACL would override the TimeWindowACL defined in the HostMetadata for all User Agent requests for movies.

3.4. Metadata Naming

GenericMetadata objects are identified by their type. The type SHOULD be descriptive, and MAY be hierarchical to support aggregating groups of properties for the purpose of readability and for avoiding name conflicts between vendor extensions. A dotted alpha-numeric notation is suggested for human readability.

Metadata types defined by this document are not hierarchical.

Examples of GenericMetadata object type names:

LocationACL

ext.vendor1.featurex

ext.vendor1.featurey

ext.vendor2.featurex

[Ed. It is intended that Metadata capability advertisements will allow either individual Metadata names or Metadata bundle identifiers to be used. Need to have a procedure for defining and distributing bundle information to be used in Metadata capability advertisement.]

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 the set of core metadata objects which may be contained within a GenericMetadata object. These objects are described as property objects as they define the semantics, enforcement options, and serialization rules for specific properties. These properties govern how User Agent requests for content are handled. Property objects may be composed of or contain references to other objects. In those cases the value of the property can be either an object of that type (the object is embedded) or a Link object that contains a URI and relationship that can be dereferenced to retrieve the CDNI Metadata object that represents the value of that property.

Note: In the following sections, the term "mandatory-to-specify" is used to convey which objects or properties must be specified for a given parent object or property. When mandatory-to-specify is set to true, it implies that if the parent object is specified, then the defined object or property MUST also be specified, e.g., a HostMatch object without a host to match against does not make sense,

therefore, the host is mandatory-to-specify inside a parent HostMatch object.

4.1. CDNI Metadata Structural Object Descriptions

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 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 a reference to Metadata objects 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 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. First match applies.

Type: List of PathMatch objects

Mandatory-to-Specify: No.

Property: modes

Description: Defines which redirection methods are supported.

Type: List of RedirectionMethod

Mandatory-to-Specify: Yes.

4.1.4. PathMatch

The PathMatch object contains 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 the CDNI Metadata properties for content served with the associated URI path (defined in a PathMatch object). Note that if CDNI metadata is used as an input to CDNI request routing and DNS-based redirection is employed, then any metadata at the PathMetadata level or below will be inaccessible 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 \?

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: match-query-string

Description: Flag indicating whether or not the query string should be included in the pattern match.

Type: Boolean

Mandatory-to-Specify: No. Default is not to include query strings when matching.

4.1.7. GenericMetadata

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

Property: type

Description: CDNI Metadata property object type.

Type: String

Mandatory-to-Specify: Yes.

Property: value

Description: CDNI Metadata property object.

Type: matches the 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: Yes.

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. CDNI Metadata Property Object Descriptions

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.

Property: sources

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

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.

Type: List of EndPoint objects

Mandatory-to-Specify: Yes.

4.2.2. LocationACL Metadata

LocationACL Metadata defines location-based restrictions.

Property: locations

Description: Access control list which applies restrictions to 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 Location objects and the corresponding action.

Property: locations

Description: List of locations to which the rule applies.

Type: List of Location objects

Mandatory-to-Specify: Yes.

[Ed: reusing locations as a property name is confusing and should likely be changed]

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. Location

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

Property: iprange

Description: A set of IP Addresses.

Type: List of IPRange objects

Mandatory-to-Specify: Yes.

[Ed: Location as specified above only supports the Class 1a names described in [I-D.jenkins-cdni-names]. Need to add support for Class 1b names to a later version.]

4.2.3. TimeWindowACL Metadata

TimeWindowACL Metadata defines time-based restrictions.

Property: times

Description: Access control list which applies restrictions to 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: times

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: Access control list which applies restrictions to 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.

Property: direction

Description: Defines whether the ProtocolRule specifies protocols for acquisition or delivery.

Type: Enumeration [acquisition|delivery]

Mandatory-to-Specify: No. Default is to apply the rule to both acquisition and delivery.

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, e.g. methods such as tokenization and URL Signing.

[Ed. Need to synchronize authentication configuration with CDNI URL signing draft definitions.]

[Ed. Need to consider how to separate protocol specific method configuration (e.g., HTTP basic/digest authentication), which must match the HostMatch protocol, from protocol agnostic method configurations (e.g., URL signing/tokenization).]

Property: direction

Description: Defines whether the Auth object applies to acquisition or delivery requests.

Type: Enumeration [acquisition|delivery]

Mandatory-to-Specify: No. Default is to apply the rule to both acquisition and delivery.

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.

Type: Boolean

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, its rel property is set to the name of the property it is replacing, 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: Yes

Property: rel

Description: The Relationship between the referring object and the object it is referencing.

Type: String

Mandatory: Yes

Property: type

Description: The type of the object being referenced.

Type: String

Mandatory: Yes

4.3.2. Protocol

Protocol objects are used to specify registered protocols for content acquisition or delivery.

[Ed. Need to reference protocol registry.]

Type: Enumeration [HTTP|RTSP|RTMP]

4.3.3. RedirectionMethod

RedirectionMethod objects are used to specify registered content redirection modes.

[Ed. Need to reference redirection method registry.]

Type: Enumeration [HTTP-I|HTTP-R|DNS-I|DNS-R]

4.3.4. 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].

4.3.5. IPRange

One of:

- A range of consecutive IP addresses (IPv4 or IPv6) expressed as Address1-Address2 which does not have to be to power of two aligned, for example the range 192.0.2.1-192.0.2.10 is valid. The first Address in the range MUST be 'lower' than the final address in the range.

- A valid IP subnet (IPv4 or IPv6) expressed using CIDR notation.

- A single IP address (IPv4 or IPv6).

Note: Client 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].

4.3.6. URI

A URI as specified in [RFC3986].

4.3.7. Time

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

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 there is no reason to delegate those requests. 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 a 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 there is likely no reason to delegate those requests.

The CDNI Footprint and Capabilities Interface 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 capabilities interface. This section describes the capabilities advertisement requirements for the metadata defined in Section 4.2

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.

[Ed. Need to register delivery protocol capability ID.]

[Ed. Need to reference protocol registry, and discuss specification of overlapping protocol values.]

5.2. Authorization Metadata Capabilities

The Authorization object contains a list of Auth values. The dCDN MUST advertise which authorization algorithms 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 authorization algorithm, the uCDN should not delegate requests requiring that algorithm to the dCDN as the dCDN will not be able to properly acquire the content or enforce delivery restrictions.

[Ed. Need to register authorization algorithm capability ID.]

[Ed. Need to reference auth registry, and discuss specification of overlapping auth values.]

5.3. Host Metadata Capabilities

The HostMetadata object contains a list of redirection method values. The dCDN MUST advertise which redirection modes it supports so that the uCDN knows how to redirect content requests to the dCDN. If the dCDN does not support a given redirection method, the uCDN should not delegate requests to the dCDN using that method as the dCDN will not be able to properly handle the redirection.

[Ed. Need to register redirection method capability ID.]

[Ed. Need to reference redirection method registry.]

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 receiving a CDNI Request Routing request from an Upstream CDN or in response to receiving a request for content from a User Agent) or in advance of being required (for example in case of prepositioned CDNI Metadata acquisition).

The CDNI Metadata interface is built on the principles of RESTful web services. 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).

In the general case a CDNI Metadata server makes each instance of an addressable CDNI Metadata object available via a unique URI that returns a representation of that instance of that CDNI Metadata object. When an object needs to reference another addressable CDNI Metadata object (for example a HostIndex object referencing a HostMetadata object) it does so by including a link to the referenced object.

CDNI Metadata servers 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. Servers implementing 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. Server implementations of this interface SHOULD reject all methods other than GET and HEAD.

As the CDNI Metadata interface builds on top of HTTP, CDNI Metadata servers 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 must decide 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 discovered by or configured in 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 (or inlined) within a property of an addressable object.

In the descriptions of objects we use the term "X contains Y" to mean either Y is directly embedded in X or that Y 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 separately addressable.

6.4.1. MIME Media Types

All MIME types are prefixed with "application/cdni." The MIME type for each object matches the type name of that object as defined by this document. Table 3 lists a few examples of the MIME Media Type for each object (resource) that is retrievable through the CDNI Metadata interface. The MIME type suffix depends on the metadata encoding, either "+xml" or "+json".

Data Object	MIME Media Type
HostIndex	application/cdni.HostIndex
HostMatch	application/cdni.HostMatch
HostMetadata	application/cdni.HostMetadata
PathMatch	application/cdni.PathMatch
PathMetadata	application/cdni.PathMetadata

Table 3: Example MIME Media Types for CDNI Metadata objects

See http://www.iana.org/assignments/media-types/index.html for reference.

6.4.2. JSON Encoding of Objects

One possible encoding for a CDNI Metadata object is 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 and therefore 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 specific to 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 document must be absolute URLs.

Type: URI

Mandatory: No

Key: links

Description: The links of this object to other addressable objects. Any property may be replaced by a link to an object with the same type as the property it replaces.

Type: List of Link objects

Mandatory: Yes

6.4.2.1. JSON Example

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

```
{
  "hosts": [
    {
      "host": "video.example.com",
      "links": [
        {
          "rel": "host-metadata",
          "type": "application/cdni.HostMetadata",
          "href": "http://metadata.example.ucdn.com/video"
        }
      ]
    },
    {
      "host": "images.example.com",
      "links": [
        {
          "rel": "host-metadata",
          "type": "application/cdni.HostMetadata",
          "href": "http://metadata.ucdn.example.com/images"
        }
      ]
    }
  ]
}
```

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

```
{
  "metadata": [
    {
      "type": "application/cdni.SourceMetadata",
```

```
         "value": {
           "sources": [
             {
               "links": [{
                 "rel": "auth",
                 "type": "application/cdni.Auth",
                 "href": "http://metadata.ucdn.example.com/auth1234"
               }],
               "endpoint": "acq1.ucdn.example.com",
               "protocol": "ftp"
             },
             {
               "links": [{
                 "rel": "auth",
                 "type": "application/cdni.Auth",
                 "href": "http://metadata.ucdn.example.com/auth1234"
               }],
               "endpoint": "acq2.ucdn.example.com",
               "protocol": "http"
             }
           ]
         }
       },
       {
         "type": "application/cdni.LocationACL",
         "value": {
           "locations": [
             {
               "locations": [
                 { "iprange": "192.168.0.0/16" }
               ],
               "action": "deny"
             }
           ]
         }
       },
       {
         "type": "application/cdni.ProtocolACL",
         "value": {
           "protocols": [
             {
               "protocols": [
                 "ftp"
               ],
               "action": "deny"
             }
           ]
         }
```

```
    }
  ],
  "paths": [
    {
      "path-pattern": {
        "pattern": "/videos/trailers/*"
      },
      "links": [{
        "rel": "path-metadata",
        "type": "application/cdni.PathMetadata",
        "href": "http://metadata.ucdn.example.com/videos/trailers"
      }]
    },
    {
      "path-pattern": {
        "pattern": "/videos/movies/*"
      },
      "links": [{
        "rel": "pathmetadata",
        "type": "application/cdni.PathMetadata",
        "href": "http://metadata.ucdn.example.com/videos/movies"
      }]
    }
  ]
}
```

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

```
{
  "metadata": [],
  "paths": [
    {
      "path-pattern": {
        "pattern": "/videos/movies/hd/*"
      },
      "links": [{
        "rel": "pathmetadata",
        "type": "application/cdni.PathMetadata",
        "href": "http://metadata.ucdn.example.com/videos/movies/hd"
      }]
    }
  ]
}
```

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.com/videos/movies/hd" with MIME type "application/cdni.PathMetadata":

```
{
  "metadata": [
    {
      "type": "application/cdni.TimeWindowACL",
      "value": {
        "times": [
          "times": [
            {
              "start": "1213948800",
              "end": "1327393200"
            }
          ],
          "type": "allow"
        ]
      }
    }
  ]
}
```

6.4.3. XML Encoding of Objects

Another possible encoding for a CDNI Metadata object is an XML document containing elements with tag names which match property names and values which match the associated property values.

Tag names of elements 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 element are dependent on the specific object being encoded (i.e. dependent on the MIME Media Type of the returned resource).

Lists are encoded by repeating the singular form of a property name. For example the "hosts" property is a list of "HostMatch" objects. This list would be encoded as multiple "host" elements.

Link objects are a special case. If a Link object replaces a property then a "link" element replaces the expected element. The properties of the Link object are encoded as XML attributes. The type attribute is set to the MIME type of the target object. The href attribute is set to the URI of the target object. The rel attribute is set to the name of the element being replaced.

6.4.3.1. XML Example

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

```
<HostIndex>
  <host>
    <host>video.example.com</host>
    <link rel="host-metadata" type="application/cdni.HostMetadata"
      href="http://metadata.ucdn.example.com/video"/>
  </host>
  <host>
    <host>images.example.com</host>
    <link rel="host-metadata" type="application/cdni.HostMetadata"
      href="http://metadata.ucdn.example.com/images"/>
  </host>
</HostIndex>
```

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

```
<HostMetadata>
  <metadata>
    <type>application/cdni.SourceMetadata</type>
    <value>
      <sources>
        <link rel="auth" type="application/cdni.Auth"
          href="http://metadata.ucdn.example.com/auth1234"/>
        <endpoint>acq1.ucdn.example.com</endpoint>
        <protocol>ftp</protocol>
      </source>
      <source>
        <link rel="auth" type="application/cdni.Auth"
          href="http://metadata.ucdn.example.com/auth1234"/>
        <endpoint>acq2.ucdn.example.com</endpoint>
        <protocol>http</protocol>
      </source>
    </value>
  </metadata>
  <metadata>
    <type>application/cdni.LocationACL</type>
    <value>
      <location>
        <location>
          <iprange>192.168.0.0/16</iprange>
```

```
        </location>
        <action>deny</type>
      </location>
    </value>
  </metadata>
  <metadata>
    <type>application/cdni.ProtocolACL</type>
    <value>
      <protocol>
        <protocol>ftp</protocol>
        <action>deny</action>
      </protocol>
    </value>
  </metadata>
  <path>
    <path-pattern>
      <pattern>/videos/trailers/*"</pattern>
    </path-pattern>
    <link rel="path-metadata" type="application/cdni.PathMetadata"
      href="http://metadata.ucdn.example.com/videos/trailers"/>
  </path>
  <path>
    <path-pattern>
      <pattern>/videos/movies/*"</pattern>
    </path-pattern>
    <link rel="path-metadata" type="application/cdni.PathMetadata"
      href="http://metadata.ucdn.example.com/videos/movies"/>
  </path>
</HostMetadata>
```

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

```
<PathMetadata>
  <path>
    <path-pattern>
      <pattern>/videos/movies/hd/*</pattern>
    </path-pattern>
    <link rel="path-metadata" type="application/cdni.PathMetadata"
      href="http://metadata.ucdn.example.com/videos/movies/hd"/>
  </path>
</PathMetadata>
```

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.com/videos/movies/hd" with MIME type "application/cdni.PathMetadata":

```
<PathMetadata>
  <metadata>
    <type>application/cdni.TimeWindowACL</type>
    <value>
      <time>
        <time>
          <start>1213948800</start>
          <end>1327393200</end>
        </time>
        <type>allow</type>
      </time>
  </metadata>
</PathMetadata>
```

6.5. Extensibility

The set of property Metadata may be extended with proprietary and/or custom 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. As described in Section 3.4, any proprietary and/or custom property Metadata SHOULD be identified by the "ext." prefix in an appropriately descriptive type which conveys the organization defining the property Metadata and the function of the property Metadata.

Note: Identification of the property Metadata defining organization in the property Metadata type decreases the possibility of property Metadata type collision. The fully-qualified domain name of the organization in reverse order may be used for this purpose.

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

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 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, ext.vendor1.Auth, and ext.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 specified by 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 and verison number of the object. HTTP requests sent to a metadata server SHOULD include an Accept header with the MIME-type and version of the expected object. Unless stated otherwise, the verison of each object defined by this document is version 1. For example: "Content-Type: application/cdni.HostIndex.v1":.

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 should specify the version number which it describes. For example: "application/cdni.Location.v1".

7. IANA Considerations

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

[Ed. Need to consider a registry for Metadata type identifiers.]

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. References

10.1. 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.

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

10.2. Informative References

[I-D.ietf-cdni-framework]
Peterson, L. and B. Davie, "Framework for CDN Interconnection", draft-ietf-cdni-framework-03 (work in progress), February 2013.

[I-D.ietf-cdni-requirements]
Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", draft-ietf-cdni-requirements-05 (work in progress), February 2013.

[I-D.zyp-json-schema]
Zyp, K. and G. Court, "A JSON Media Type for Describing the Structure and Meaning of JSON Documents", draft-zyp-json-schema-03 (work in progress), November 2010.

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

[RFC4151] Kindberg, T. and S. Hawke, "The 'tag' URI Scheme", RFC 4151, October 2005.

[RFC4287] Nottingham, M., Ed. and R. Sayre, Ed., "The Atom Syndication Format", RFC 4287, December 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.

Appendix A. Relationship to the CDNI Requirements

Section 6 of [I-D.ietf-cdni-requirements] lists the requirements for the CDNI Metadata Distribution interface. This section outlines which of those requirements are met by the CDNI Metadata interface specified in this document.

All metadata requirements are met either directly or indirectly by the CDNI Metadata Interface described in this document, with the clarifications or exceptions described in the following paragraphs.

Requirements related to pre-positioning of metadata are met by this document on the assumption that other CDNI Interfaces are to be used by the upstream CDN to trigger the pre-positioning of metadata by the downstream CDN via the CDNI Metadata Interface. Triggering metadata pre-positioning is beyond the scope of the CDNI Metadata interface. However, the interface as described by this document supports pulling metadata on-demand for the purpose of pre-positioning.

Requirement META-7 relating to modification of metadata by the upstream CDN is met both by allowing timeouts on the cacheability of metadata objects and by allowing other CDNI interfaces to initiate a refetch or purge of metadata.

Requirement META-18 relating to surrogate cache behavior parameters is supported via extensibility. However, the example parameters in META-18 are not described in this document.

Appendix B. Metadata Rewriting

For some use cases, one CDN in a chain of interconnected CDNs must be able to rewrite CDNI Metadata received from its upstream CDN before presenting that CDNI Metadata to its downstream CDN.

The CDN which is performing the metadata rewriting is referred to as the 'Transit' CDN (tCDN), its upstream CDN as the uCDN and its downstream CDN as the dCDN.

Two (non-exhaustive) examples of when rewriting are:

> Allowing the dCDN is to acquire content from the tCDN instead of (or as well as) the uCDN. The tCDN must modify the appropriate CDNI Source Metadata objects to include itself as a possible source for the content.

If the tCDN is transforming the original URI as part of CDNI request redirection on-route to the dCDN, the tCDN may need to modify the PatternMatch objects in any PathMetadata to take account of any URI path transformation it has performed.

When performing HTTP redirection between CDNs, the dCDN must be able to map an UA request to a host and path which are meaningful to the tCDN. The dCDN needs only to identify its immediate upstream neighbor and does not need to map (or understand) the entire chain of CDNs that precede the tCDN.

A dCDN may encode the identity of the tCDN in the URI it returns to the UA as part of request redirection (either directly or via the CDNI Request Routing Redirection interface). The exact method the dCDN uses to encode the information it requires is a local implementation decision provided it enables the dCDN to identify the correct upstream CDN (tCDN) and to map the request to the appropriate host and path so that the dCDN can find and retrieve the correct CDNI Metadata from tCDN.

B.1. Example

The example in this section is not necessarily representative of URL rewriting in practice.

The UA requests the following URI from the uCDN:

http://video.example/foo/bar

The uCDN makes a CDNI Request Routing Redirection request to tCDN and tCDN returns a redirection URI of:

http://tcdn.example/tcdn-prefix/foo/bar

The tcdn-prefix/ encodes sufficient information for tCDN to identify uCDN as its upstream CDN neighbor. The tCDN makes a CDNI Request Routing Redirection request to dCDN and dCDN returns a redirection URI of:

http://dcdn.example/dcdn-prefix/tcdn-prefix/foo/bar

Therefore when dCDN receives a request for:

http://dcdn.example/dcdn-prefix/tcdn-prefix/foo/bar

The dCDN can use /dcdn-prefix/ to identify tCDN as its upstream CDN neighbor and reconstruct the URI tCDN expects. The tCDN can in turn use /tcdn-prefix/ to identify uCDN as its upstream CDN neighbour and reconstruct the URI uCDN expects.

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

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

Email: gwatson@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
Azuki Systems, Inc.
43 Nagog Park
Acton, MA 01720
USA

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