Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 10, 2013

B. Niven-Jenkins
R. Murray
G. Watson
Velocix (Alcatel-Lucent)
M. Caulfield
K. Leung
Cisco Systems
July 9, 2012

CDN Interconnect Metadata
draft-cjlmw-cdni-metadata-00

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

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 10, 2013.

Copyright Notice

Table of Contents

1. Introduction

CDNI enables a downstream CDN to service content requests on behalf of an upstream CDN. In the simplest use case, a content request received by the downstream CDN provides sufficient information for sending a response. More complex use cases require additional context, i.e. metadata. 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 [I-D.ietf-cdni-problem-statement] along with three other interfaces that may be used to compose a CDNI solution (Control, Request Routing and Logging). [I-D.davie-cdni-framework] expands on the information provided in [I-D.ietf-cdni-problem-statement] 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 4.1 through Section 4.2).
- A RESTful web service for the transfer of CDNI Metadata (Section 5).

1.1. Terminology

This document reuses the terminology defined in [I-D.ietf-cdni-problem-statement].

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

- Object - a collection of properties

- Property - a key / value pair where the key is a property name and the value is the property value (possibly an object)

2. Design Principles

The proposed CDNI Metadata Interface aims to achieve the following design principles:

1. Cacheability of CDNI metadata objects
2. Deterministic mapping from 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 and therefore improves the latency of serving content requests. The CDNI Metadata Interface uses HTTP to achieve cacheability.

Deterministic mappings from content requests to metadata properties eliminates ambiguity and ensures that the same 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 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 authoritative 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 contain different sets of CDNI Metadata properties in order to describe the required behaviour when a dCDN surrogate is processing User Agent requests for content on 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 remaining CDNI Metadata objects and describes the relationships between those objects as well as the relationships of those objects to HostMetadata and PathMetadata objects.

Section 4.1 describes the specific properties of each object in more detail.

3.1. HostIndex, HostMetdata & PathMetadata objects

A HostIndex object contains a list of Hostnames that 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

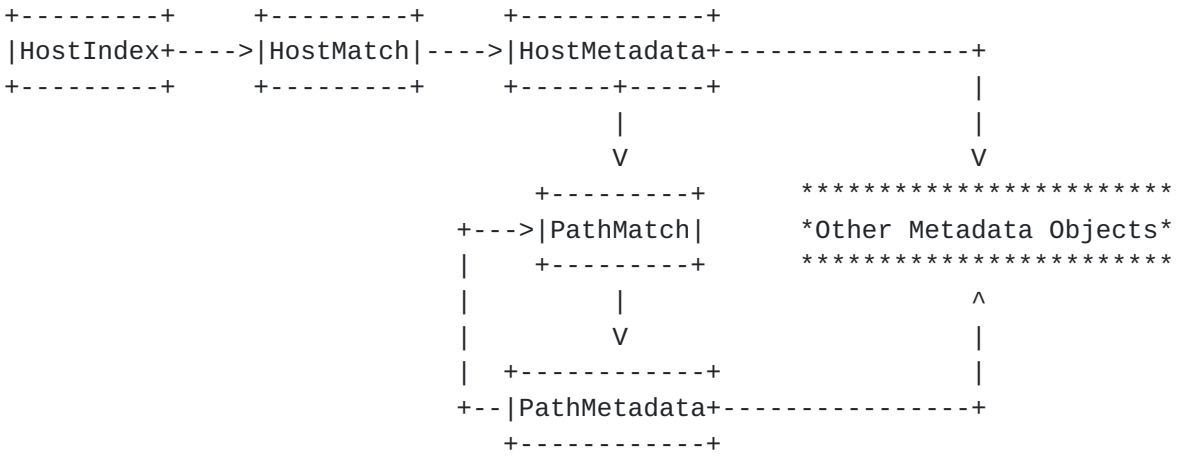
order to deliver the requested content.

The HostIndex links end-user facing Hostnames to HostMetadata objects, which contain (or reference) the default CDNI Metadata required to serve content for that Hostname. When looking up CDNI Metadata, the downstream CDN looks up the requested Hostname in the HostIndex, from there it can find HostMetadata which describes delivery rules for a Hostname and PathMetadata which may override those rules for given URI paths within the Hostname.

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.



Key: ----> = References

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

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

Data Object	Description
HostIndex	A HostIndex object lists the Hostnames that an upstream CDN can provide CDNI metadata for and the URIs to use for retrieving that CDNI Metadata. For example, if "example.com" is a content provider, the HostIndex object may include an entry for "example.com" with the URI of the associated HostMetadata object. These hostnames are contained inside a list of 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 properties to be applied when a content request matches against the hostname.
HostMetadata	A HostMetadata object contains (or references) the default CDNI Metadata properties for content served from that hostname, i.e. the CDNI Metadata properties 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 properties for those specific URI paths.
PathMatch	A PathMatch object defines a pattern to match against the requested path, and contains or references a PathMetadata object which contains (or references) the CDNI Metadata properties to be applied when a content request matches against the defined URI path pattern.

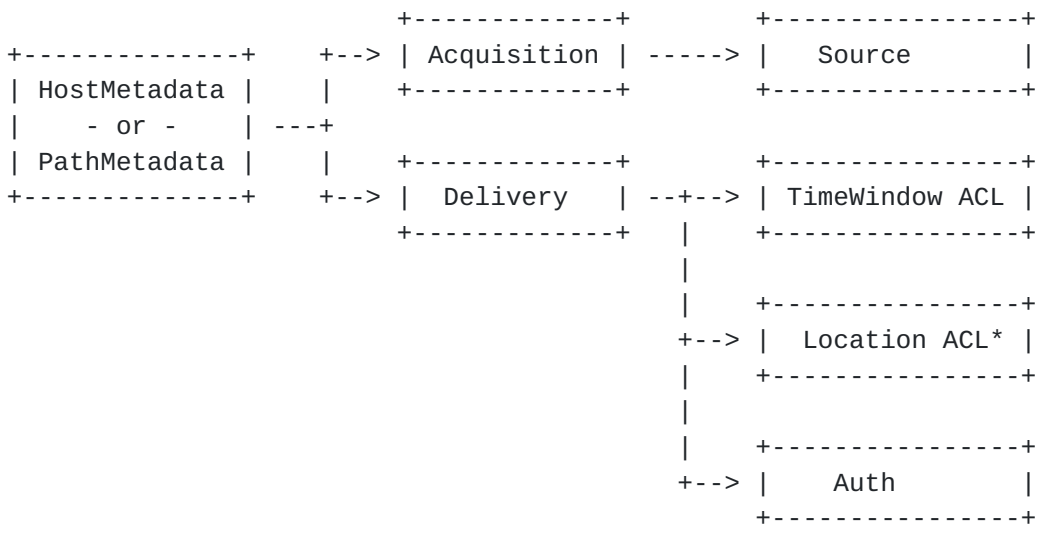
PathMetadata	A PathMetadata object contains the CDNI Metadata properties for content served with the associated URI path (defined in a PathMatch object). A PathMetadata object may also contain PathMatch objects in order to recursively define more specific URI paths that require different (e.g. more specific) 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.

Table 1: HostIndex, HostMetadata and PathMetadata CDNI Metadata Objects

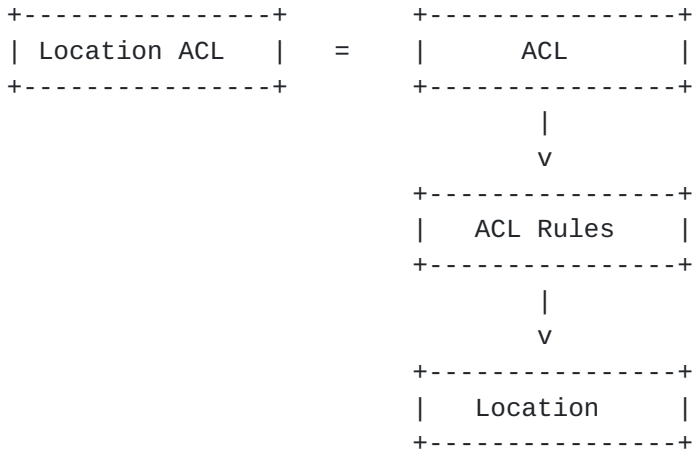
3.2. Remaining CDNI Metadata objects

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. The properties associated with the processing of User Agent requests fall into two categories, Delivery and Acquisition. Delivery properties, such as Location based restrictions, are contained or referenced within a Delivery object. Acquisition properties, such as which Origin Server to use to acquire the content, are contained or referenced within an Acquisition Object. Delivery and Acquisition objects contain or reference other CDNI Metadata objects to define the properties and rules which should be applied when processing requests for content. In some cases the rules that should be applied are complex but also likely to be reusable and repeated across many HostMetadata or PathMetadata objects.

The relationships between the HostMetadata and PathMetadata objects and the other CDNI Metadata objects (Delivery object, Acquisition object, etc.) required for CDNI request routing and delivery are illustrated in Figure 2.



*example ACL



Key: ----> = References

Figure 2: Relationships between HostMetadata and PathMetadata and the other CDNI Metadata Objects

The table below describes the remaining CDNI Metadata objects that were not defined in Section 3.1.

Data Object	Description
Acquisition	Container object for metadata that applies to content acquisition.
Delivery	Container object for metadata that applies to content delivery.
Source	Information needed by a dCDN to acquire content. For example the host to contact, the protocol to use for acquisition and any authentication and authorization methods that should be used.
ACL	Contains or references a list of ACLRules that are used to define any delivery restrictions that must be applied e.g. Location restrictions or Time based restrictions.
ACLRule	Contains or references a list of objects which define to what the restrictions should be applied e.g. an ACLRule may reference a Location Object if a location based ACL is required.
TimeWindow	Start and end time used to specify windows of availability or unavailability for the content.
Location	Geographic or network location identified by country code, BGP AS number, or subnet to which content may (or may not) be delivered.
Auth	Method and credentials for authentication and authorization including URI-signing, token-base, etc.

Table 2: Content Distribution Metadata Data Objects

The relationships in Figure 1 and Figure 2 are summarised in Table 3 below and the properties of each object are described in Section 4.1.

Data Object	Objects it References
HostIndex	0 or more HostMetadata objects.
HostMetadata	0 or more PathMatch objects. 0 or 1 Delivery objects. 0 or 1 Acquisition objects.
PathMatch	1 PathMetadata object.
PathMetadata	0 or more PathMatch objects. 0 or 1 Delivery objects. 0 or 1 Acquisition objects.
Acquisition	0 or more Source objects.
Delivery	0 or more ACL objects. 0 or more Auth objects.
ACL	0 or more ACLRule objects.

ACLRule	0 or more Location objects or 0 or more TimeWindow objects.

Table 3: Relationships between CDNI Metadata Objects

3.3. Metadata Inheritance

In the data model, a HostMetadata object may contain (or reference) multiple PathMetadata objects. Each PathMetadata object may in turn contain (or reference) other PathMetadata objects. These relationships form a tree.

The tree of HostMetadata objects and PathMetadata objects forms an inheritance tree. Each node in the tree inherits the property values set by its parent.

In the tree, a child may override any property value which has been set by its parent. If a HostMetadata object sets the value of a property, that value may be overridden by a PathMetadata object (the child of the HostMetadata object). If a PathMetadata object contains (or references) other PathMetadata objects as children, then those children PathMetadata objects may override the property values set by the parent PathMetadata object.

If a child node overrides the value of a list, then the entire list is replaced with the value set by the child node. If a child node overrides the value of an object, then the whole object is replaced with the value set by the child node.

4. Encoding-Independent CDNI Metadata Object Descriptions

This section provides the definitions of each object type declared in Section 3. The definition of each object contains an unordered set of properties. The type of some properties is another CDNI Metadata object and 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 describes a URI and relationship that can be dereferenced to retrieve the CDNI Metadata object that should be used as the value of that property.

4.1. CDNI Metadata Data Object Descriptions

Each of the sub-sections below describes the properties associated with the data objects defined in Table 2.

4.1.1. HostIndex

The HostIndex object is the entry point into the CDNI Metadata hierarchy. An incoming content request is matched against the list of hosts to find the HostMatch object which applies to the request.

Property: hosts
Description: List of HostMatch objects
Type: List of HostMatch
Mandatory: Yes.

4.1.2. HostMatch

The HostMatch object contains a hostname to match against and a metadata object to apply if a match is found.

Property: hostname
Description: String to match against the requested host.
Type: String
Mandatory: Yes
Property: hostmetadata
Description: CDNI Metadata to apply when delivering content that matches this pattern.
Type: HostMetadata
Mandatory: 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: acquisition
Description: Container for content acquisition related metadata.
Type: Acquisition
Mandatory: No. No default.
Property: delivery
Description: Container for content delivery related metadata.
Type: Delivery
Mandatory: No. No default.
Property: paths
Description: Path specific rules. First match applies.
Type: List of PathMatch
Mandatory: No. Default apply the properties defined in this HostMetadata object to all paths.

Property: hostname
Description: The end-user facing Hostname for this HostMetadata object.
Type: Hostname
Mandatory: Yes.

4.1.4. Acquisition

Metadata which provides the dCDN information about content acquisition e.g. how to contact an uCDN Surrogate or an Origin Server. 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.
Type: List of Source
Mandatory: No. Defaults to empty list.

4.1.5. Delivery

Metadata related to content delivery, e.g. delivery restrictions or content authorization methods.

Property: locations
Description: Access control list which applies restrictions to delivery based on client location.
Type: ACL
Mandatory: No. Defaults is allow all locations.
Property: times
Description: Access control list which applies restrictions to delivery based on request time.
Type: ACL
Mandatory: No. Defaults is allow all times.
Property: auth
Description: Options for authenticating content requests. All options in the list are equally valid.
Type: List of Auth
Mandatory: No. Defaults is no auth.
Property: protocol
Description: The delivery protocol to be used for content requests that match this HostMetadata object.
Type: protocol
Mandatory: Yes.
Property: active
Description: Enable or disable delivery from this host.
Type: boolean

Mandatory: No. Default yes.

4.1.6. PathMatch

The PathMatch object contains an expression to match against and a metadata object to apply if a match is found.

Property: pattern
Description: String to match against the requested path, i.e. against the [RFC3986] path-absolute.
Type: Pattern
Mandatory: Yes
Property: patternflags
Description: Flags to control the pattern match.
Type: List of PatternFlags
Mandatory: No. Default Case-sensitive infix matching.
Property: pathmetadata
Description: CDNI Metadata to apply when delivering content that matches this pattern.
Type: PathMetadata
Mandatory: Yes

4.1.7. 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.

PathMetadata objects may contain any of the properties of a HostMetadata object with the following exceptions:

- PathMetadata objects MUST NOT contain a hostname property.
- PathMetadata objects MUST NOT contain a protocol property.
- The presence of an sources property is OPTIONAL.

4.1.8. ACL

An ACL object contains or references a list of ACLRule objects which define a set of restrictions to apply to content delivery e.g. Location restrictions. An ACL may reference or contain ACLRules referencing or containing Location or TimeWindow objects but not both.

Property: aclrules
Description:
Type: List of ACLRule
Mandatory: No. Default no rules.

4.1.9. ACLRule

An ACLRule contains or references a list of either TimeWindow or Location objects. ACLRule objects are used to construct ACL to apply restrictions to content delivery.

Note: Although both the allow and deny properties are optional, one and only one of them MUST be present in an ACLRule. An ACLRule must also only refer to one of Location or TimeWindow but not both and should only refer to the objects relevant to the ACL type as defined by Delivery Metadata i.e. a Delivery Metadata object with an ACL with relationship of LocationACL must not reference TimeWindow objects further down in the Metadata hierarchy.

Property: allow
Description: List of either Locations (Location ACL) or Time Windows (TimeWindow ACL) which must be allowed.
Type: List of Location or TimeWindow
Mandatory: No. Default implicit Allow.
Property: deny
Description: List of either Locations (Location ACL) or Time Windows (TimeWindow ACL) which must be denied.
Type: List of Location or TimeWindow
Mandatory: No. Default implicit Deny.

4.1.10. TimeWindow

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

Property: start
Description: The start time of the window.
Type: Time
Mandatory: Yes
Property: end
Description: The end time of the window.
Type: Time
Mandatory: Yes

4.1.11. Location

A Location object describes a Location which may be applied by an ACLRule, 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.
Mandatory: 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.]

4.1.12. 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: No. Default is no authentication.
Property: endpoints
Description: Origins from which the dCDN can acquire content.
Type: List of EndPoint
Mandatory: Yes.
Property: protocol
Description: Protocol to use for content acquisition.
Type: Protocol
Mandatory: Yes.

4.1.13. 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.

Property: type
Description: A string containing the authentication type "url-signing", "url-token", "http-basic", or "http-digest". The type dictates which optional fields are present and valid in the rest of the object. The "url-signing" type refers to URL signing authentication. The "url-token" type refers to token-

based authentication. The "basic" and "digest" types refer to HTTP Basic and Digest access authentication.
Type: String
Mandatory: Yes.

Property: algo
Description: A string containing the signature algorithm (e.g. "md5", "sha-1", etc.).
Type: String
Mandatory: Yes, if type is "url-signing".

Property: symmetric
Description: A boolean if true, URL signing uses symmetric keys, otherwise asymmetric.
Type: boolean
Mandatory: Yes, if type is "url-signing".

Property: key
Description: A hex-encoded number containing the public key for verifying signatures, only valid if "symmetric" field is set to false.
Type: boolean
Mandatory: Yes, if type is "url-signing".

Property: username
Description: A string containing the username for "basic" and "digest" types.
Type: String
Mandatory: Yes, if type is "basic" or "digest".

Property: password
Description: A string containing the password for "basic" and "digest" types.
Type: String
Mandatory: Yes, if type is "basic" or "digest".

4.1.14. Link

A link object may be used in place of any of the objects 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
```

4.2. CDNI Metadata Simple Data Type Descriptions

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

4.2.1. Protocol

This type only appears in Links. Links with this type are not machine readable but rather represent particular feature sets of a protocol defined in a specification and implemented in code. The URI contained in the link needs to be defined for each delivery protocol with an associated interoperable feature set.

The following examples are illustrative:

- http://url.cdni.ietf.example/protocol/delivery/http/rfcABCD
- http://url.cdni.ietf.example/protocol/delivery/rtmp/rfcEFGH
- http://url.vendorY.ietf.example/protocol/delivery/rtmp/releaseP.Q

[Editor's Note: It may be more appropriate to use the 'tag' URI scheme [RFC4151] for these URIs.]

4.2.2. Endpoint

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

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].

4.2.3. 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].

4.2.4. Pattern

A pattern for string matching paths. The string may contain the wildcards * and ?.

- * matches any sequence of characters (including the empty string).
- ? matches exactly one character.

Escaping: The three literals \ , * and ? should be escaped as \\, * and \?

4.2.5. PatternFlags

A set of flags indicating how a pattern match is made. The flags are:

- Case-insensitive - Perform a case insensitive match (absence indicates case-sensitive match).
- Prefix - Match against the start of the string (absence indicates that a match may start anywhere in the string).
- Suffix - Match against the end of the string (absence indicates that a match may end anywhere in the string).

Absence of both Prefix and Suffix results in a match against any part of the string (infix).

4.2.6. URI

A URI as specified in [RFC3986].

4.2.7. Time

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

5. 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.

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 4.1).

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 impose a definitive structure on CDNI Metadata interface implementations.

5.1. 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.

5.2. 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 Hosts (along with their public facing hostnames) that the upstream CDN may delegate 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 HostIndex). The CDNI metadata client then makes a GET request for the URI specified in the href key of that Host's entry in the HostIndex.

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, 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 metadata should handle a particular User Agent request.

In the case of where 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 in case of HTTP redirection across CDNs). With knowledge of which upstream CDN routed the request, the downstream CDN can choose the correct metadata server.

In the case of DNS redirection there is not sufficient information carried in the DNS request from User Agents to determine the upstream CDN that redirected a particular request and therefore downstream CDNs may have to apply local policy when deciding which upstream CDN's metadata to apply.

5.3. 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 via the CDNI Control interface. An upstream CDN would advertise the URI of the HostIndex object to the downstream CDN via the CDNI Control Interface.

5.4. 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.

5.4.1. 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 4 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 4: MIME Media Types for CDNI Metadata resources

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

5.4.2. 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 any dictionary key defined by this document (for example the names of CDNI Metadata object properties) MUST always be represented in lowercase.

In addition to the properties of the object, the following three additional 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
Mandatory: Yes

5.4.2.1. JSON Example

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

```
{
  "host": [
    {
      "hostname": "video.example.com",
      "links": [
        {
          "rel": "hostmetadata",
          "type": "HostMetadata",
          "href": "http://metadata.ucdn.com/video"
        }
      ]
    },
    {
      "hostname": "images.example.com",
      "links": [
        {
          "rel": "hostmetadata",
          "type": "HostMetadata",
          "href": "http://metadata.ucdn.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.com/video" expecting a MIME type of "application/cdni.HostMetadata+json":

```
{
  "hostname": "video.example.com",
  "acquisition": {
    "source": [
      {
        "links": [{
          "rel": "auth",
          "type": "Auth",
          "href": "http://metadata.ucdn.com/auth1234"
        }],
```

```
          "endpoint": "acq1.ucdn.com",
          "protocol": "ftp"
        },
        {
          "links": [{
            "rel": "auth",
            "type": "Auth",
            "href": "http://metadata.ucdn.com/auth1234"
          }],
          "endpoint": "acq2.ucdn.com",
          "protocol": "http"
        }
      ]
    },
    "delivery": {
      "location": {
        "aclrule": {
          "deny": { "iprange": "192.168.0.0/16" }
        }
      },
      "auth": {

      },
      "protocol": "http",
      "active": "true"
    },
    "path": [
      {
        "pattern": "/videos/trailers/*",
        "patternflags": "prefix",
        "links": [{
          "rel": "pathmetadata",
          "type": "PathMetadata",
          "href": "http://metadata.ucdn.com/videos/trailers"
        }]
      },
      {
        "pattern": "/videos/movies/*",
        "patternflags": "prefix",
        "links": [{
          "rel": "pathmetadata",
          "type": "PathMetadata",
          "href": "http://metadata.ucdn.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.com/video/movies" with an expected type of "application/cdni.PathMetadata":

```
{
  "delivery": {
    "auth": {

    }
  },
  "path": {
    "pattern": "/videos/movies/hd/*",
    "patternflags": "prefix",
    "links": [{
      "rel": "pathmetadata",
      "type": "PathMetadata",
      "href": "http://metadata.ucdn.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.com/videos/movies/hd" with MIME type "application/cdni.PathMetadata":

```
{
  "delivery": {
    "time": {
      "aclrule": {
        "allow": {
          "start": "1213948800",
          "end": "1327393200"
        }
      }
    }
  }
}
```

5.4.3. 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.

5.4.3.1. XML Example

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

```
<HostIndex>
  <host>
    <hostname>video.example.com</hostname>
    <link rel="hostmetadata" type="HostMetadata"
      href="http://metadata.ucdn.com/video"/>
  </host>
  <host>
    <hostname>images.example.com</hostname>
    <link rel="hostmetadata" type="HostMetadata"
      href="http://metadata.ucdn.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.com/video" expecting a MIME type of "application/cdni.HostMetadata+json":

```
<HostMetadata>
  <hostname>video.example.com</hostname>
  <acquisition>
    <source>
      <link rel="auth" type="Auth"
        href="http://metadata.ucdn.com/auth1234"/>
      <endpoint>acq1.ucdn.com</endpoint>
      <protocol>ftp</protocol>
    </source>
    <source>
      <link rel="auth" type="Auth"
        href="http://metadata.ucdn.com/auth1234"/>
      <endpoint>acq2.ucdn.com</endpoint>
      <protocol>http</protocol>
    </source>
  </acquisition>
  <delivery>
    <location>
      <aclrule>
        <deny>
          <iprange>192.168.0.0/16</iprange>
        </deny>
      </aclrule>
    </location>
    <auth></auth>
    <protocol>http</protocol>
    <active>true</active>
  </delivery>
  <path>
    <pattern>/videos/trailers/*</pattern>
    <patternflags>prefix</patternflags>
    <link rel="pathmetadata" type="PathMetadata"
      href="http://metadata.ucdn.com/videos/trailers"/>
  </path>
  <path>
    <pattern>/videos/movies/*</pattern>
    <patternflags>prefix</patternflags>
    <link rel="pathmetadata" type="PathMetadata"
      href="http://metadata.ucdn.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.com/video/movies" with an expected type of "application/cdni.PathMetadata":

```
<PathMetadata>
  <delivery>
    <auth></auth>
  </delivery>
  <path>
    <pattern>/videos/movies/hd/*</pattern>
    <patternflags>prefix</patternflags>
    <link rel="pathmetadata" type="PathMetadata"
      href="http://metadata.ucdn.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.com/videos/movies/hd" with MIME type "application/cdni.PathMetadata":

```
<PathMetadata>
  <delivery>
    <time>
      <aclrule>
        <allow>
          <start>1213948800</start>
          <end>1327393200</end>
        </allow>
      </aclrule>
    </time>
  </delivery>
</PathMetadata>
```

5.5. Extensibility

The set of metadata properties may be extended with proprietary and / or custom properties. New properties may be added to any existing object.

The names of such properties MUST begin with an "x-" prefix. If a property is vendor specific, then "x-vendor-" SHOULD be used as the name prefix, where the "vendor" string is replaced by the name of the vendor.

The values of new properties MAY include an "ignorable" property with a boolean type. If "ignorable" is set to true, then request routers and surrogates in any interconnected CDN MAY safely ignore the new property. If "ignorable" is set to false, then a CDN which does not understand the property MUST NOT service a request for the corresponding content.

6. IANA Considerations

This document requests the registration of the "application/cdni" MIME type.

7. Security Considerations

The CDNI Metadata Interface is expected to be secured as a function of the transport protocol (e.g. HTTP authentication).

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.

A malicious metadata client could request metadata for a piece of content from an upstream CDN. However, given the current set of metadata properties, no useful information would be compromised.

8. Acknowledgements

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

9. References

9.1. Normative References

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

[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.

9.2. Informative References

[I-D.davie-cdni-framework] Davie, B. and L. Peterson, "Framework for CDN Interconnection", draft-davie-cdni-framework-00 (work in progress), July 2011.

[I-D.ietf-cdni-problem-statement]
Niven-Jenkins, B., Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", draft-ietf-cdni-problem-statement-03 (work in progress), January 2012.

[I-D.ietf-cdni-requirements]
Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", draft-ietf-cdni-requirements-02 (work in progress), December 2011.

[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.

[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. The following paragraphs describe notable exceptions.

Requirements related to pre-positioning of metadata are not met directly by this document. 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-13 relating to feedback from the downstream CDN to the upstream CDN with respect to metadata is not directly supported by the pull-based interface described in this document. As an alternative, the downstream CDN may use the CDNI Logging interface to convey error conditions related to 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.

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