Internet Engineering Task Force (IETF)
Request for Comments: 8006
Category: Standards Track
ISSN: 2070-1721

B. Niven-Jenkins
R. Murray
Nokia
M. Caulfield
Cisco Systems
K. Ma
Ericsson

December 2016

Content Delivery Network Interconnection (CDNI) Metadata

Abstract

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

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc8006.

Copyright Notice

Table of Contents

1. Introduction

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

The CDNI Metadata interface (MI) is discussed in [RFC7336] along with four other interfaces that can be used to compose a CDNI solution (the CDNI Control interface, the CDNI Request Routing Redirection interface, the CDNI Footprint & Capabilities Advertisement interface (FCI), and the CDNI Logging interface). [RFC7336] describes each interface and the relationships between them. The requirements for the CDNI Metadata interface are specified in [RFC7337].

The CDNI Metadata associated with a piece of content (or with a set of content) provides a dCDN with sufficient information for servicing content requests on behalf of a uCDN, in accordance with the policies defined by the uCDN.

This document defines a CDNI Metadata interface that enables a dCDN to obtain CDNI Metadata from a uCDN so that the dCDN can properly process and respond to:

- Redirection requests received over the CDNI Request Routing Redirection interface [RFC7975].
- Content requests received directly from User Agents.

Specifically, this document defines:

- A data structure for mapping content requests and redirection requests to CDNI Metadata objects (Sections 3 and 4.1).
- An initial set of CDNI GenericMetadata objects (Section 4.2).
- An HTTP 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 another object.

This document uses the phrase "[Object] A contains [Object] B" for simplicity when a strictly accurate phrase would be "[Object] A contains or references (via a Link object) [Object] B".

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

1.2. Supported Metadata Capabilities

Only the metadata for a small set of initial capabilities is specified in this document. This set provides the minimum amount of metadata for basic CDN interoperability while still meeting the requirements set forth by [RFC7337].

The following high-level functionality can be configured via the CDNI Metadata objects specified in Section 4:

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

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

The set of metadata specified in this document covers the initial capabilities above. It is only intended to support CDNI for the delivery of content by a dCDN using HTTP/1.1 [RFC7230] and for a dCDN to be able to acquire content from a uCDN using either HTTP/1.1 or HTTP/1.1 over Transport Layer Security (TLS) [RFC2818].

Supporting CDNI for the delivery of content using unencrypted HTTP/2 [RFC7540] (as well as for a dCDN to acquire content using unencrypted HTTP/2 or HTTP/2 over TLS) requires the registration of these protocol names in the "CDNI Metadata Protocol Types" registry (Section 7.3).

Delivery of content using HTTP/1.1 over TLS or HTTP/2 over TLS SHOULD follow the guidelines set forth in [RFC7525]. Offline configuration of TLS parameters between CDNs is beyond the scope of this document.

2. Design Principles

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

1. Cacheability of CDNI Metadata objects;

2. Deterministic mapping from redirection requests and content requests to CDNI Metadata properties;

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

4. Minimal duplication of CDNI Metadata; and

5. Leveraging of existing protocols.

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

Deterministic mapping from content to metadata properties eliminates ambiguity and ensures that policies are applied consistently by all dCDNs.

Support for both HTTP and DNS redirection ensures that the CDNI Metadata meets the same design principles for both HTTP-based and DNS-based redirection schemes.

Minimal duplication of CDNI Metadata improves storage efficiency in the CDNs.

Leveraging existing protocols avoids reinventing common mechanisms such as data structure encoding (by leveraging I-JSON (Internet JSON) [RFC7493]) and data transport (by leveraging HTTP [RFC7230]).

3. CDNI Metadata Object Model

The CDNI Metadata object model describes a data structure for mapping redirection requests and content requests to metadata properties. Metadata properties describe how to acquire content from a uCDN, authorize access to content, and deliver content from a dCDN. The object model relies on the assumption that these metadata properties can be grouped based on the hostname of the content and subsequently on the resource path (URI) of the content. The object model associates a set of CDNI Metadata properties with a hostname to form a default set of metadata properties for content delivered on behalf of that hostname. That default set of metadata properties can be overridden by properties that apply to specific paths within a URI.

Different hostnames and URI paths will be associated with different sets of CDNI Metadata properties in order to describe the required behavior when a dCDN Surrogate or request router is processing User Agent requests for content at that hostname and URI path. As a result of this structure, significant commonality could 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 grouped together into a single network or geographic location is likely to be common for a number of different hostnames; 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 could be applied to multiple hostnames delivered through the CDN.

In order to enable the CDNI Metadata for a given hostname and URI path to be decomposed into reusable sets of CDNI Metadata properties, the CDNI Metadata interface splits the CDNI Metadata into separate objects. Efficiency is improved by enabling a single CDNI Metadata object (that is shared across hostnames and/or URI paths) to be retrieved and stored by a dCDN once, even if it is referenced by the CDNI Metadata for multiple hostnames and/or URI paths.

Important Note: Any CDNI Metadata object A that contains another CDNI Metadata object B can include a Link object specifying a URI that can be used to retrieve object B, instead of embedding object B within object A. The remainder of this document uses the phrase "[Object] A contains [Object] B" for simplicity when a strictly accurate phrase would be "[Object] A contains or references (via a Link object) [Object] B". It is generally a deployment choice for the uCDN implementation to decide when to embed CDNI Metadata objects and when to reference separate resources via Link objects.

Section 3.1 introduces a high-level description of the HostIndex, HostMatch, HostMetadata, PathMatch, PatternMatch, and PathMetadata objects, and describes the relationships between them.

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 specified by this document that can be contained within a CDNI GenericMetadata object.

3.1. HostIndex, HostMatch, HostMetadata, PathMatch, PatternMatch, and PathMetadata Objects

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

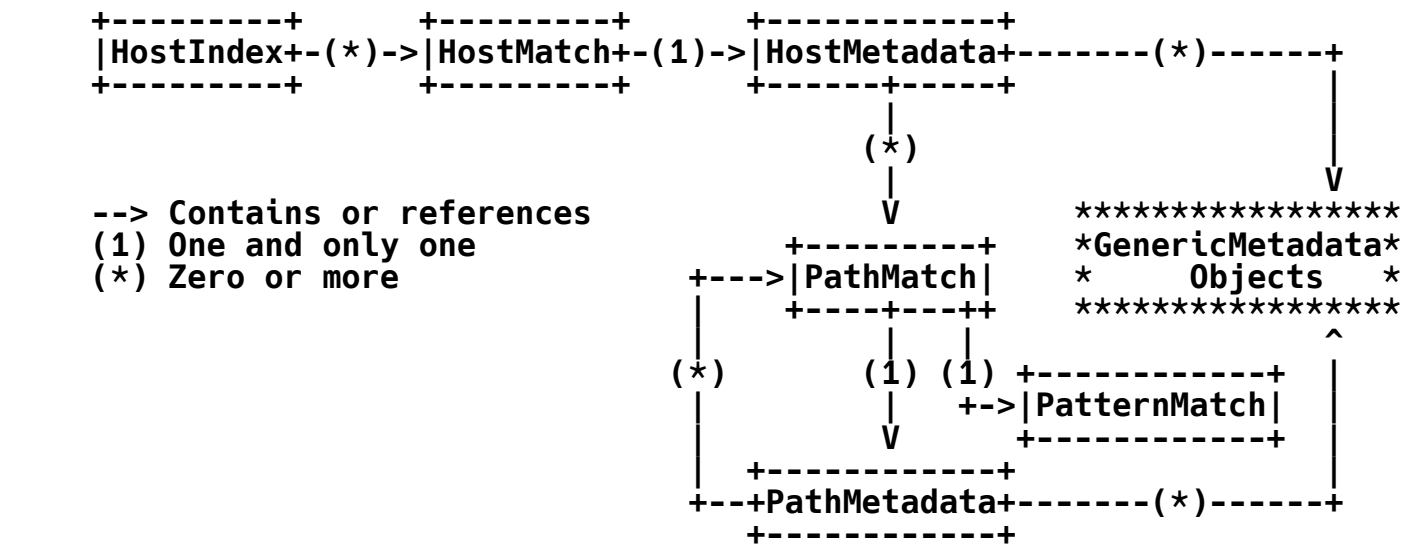

Figure 1: Relationships between CDNI Metadata Objects (Diagram Representation)

A HostIndex object (see Section 4.1.1) contains an array of HostMatch objects (see Section 4.1.2) that contain hostnames (and/or IP addresses) for which content requests might be delegated to the dCDN. The HostIndex is the starting point for accessing the uCDN CDNI Metadata data store. It enables the dCDN to deterministically discover which CDNI Metadata objects it requires in order to deliver a given piece of content.

The HostIndex links hostnames (and/or IP addresses) to HostMetadata objects (see Section 4.1.3) via HostMatch objects. A HostMatch object defines a hostname (or IP address) to match against a requested host and contains a HostMetadata object.

HostMetadata objects contain the default GenericMetadata objects (see Section 4.1.7) required to serve content for that host. When looking up CDNI Metadata, the dCDN looks up the requested hostname (or IP address) against the HostMatch entries in the HostIndex; from there, it can find HostMetadata, which describes the default metadata properties for each host as well as PathMetadata objects (see Section 4.1.6), via PathMatch objects (see Section 4.1.4). PathMatch objects define patterns, contained inside PatternMatch objects (see Section 4.1.5), to match against the requested URI path. PatternMatch objects contain the pattern strings and flags that describe the URI path to which a PathMatch applies. PathMetadata objects contain the GenericMetadata objects that apply to content requests matching the defined URI path pattern. PathMetadata properties override properties previously defined in HostMetadata or less-specific PathMatch paths. PathMetadata objects can contain additional PathMatch objects to recursively define more-specific URI paths to which GenericMetadata properties might be applied.

A GenericMetadata object contains individual CDNI Metadata objects that define the specific policies and attributes needed to properly deliver the associated content. For example, a GenericMetadata object could describe the source from which a CDN can acquire a piece of content. The GenericMetadata object is an atomic unit that can be referenced by HostMetadata or PathMetadata objects.

For example, if "example.com" is a content provider, a HostMatch object could include an entry for "example.com" with the URI of the associated HostMetadata object. The HostMetadata object for "example.com" describes the metadata properties that apply to "example.com" and could contain PathMatches for "example.com/movies/*" and "example.com/music/*", which in turn reference corresponding PathMetadata objects that contain the properties for those more-specific URI paths. The PathMetadata object for "example.com/movies/*" describes the properties that apply to that URI path. It could also contain a PathMatch object for "example.com/movies/hd/*", which would reference the corresponding PathMetadata object for the "example.com/movies/hd/" path prefix.

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

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

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

3.2. Generic CDNI Metadata Objects

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

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

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

Redistribution safety MUST be specified for each GenericMetadata property. If a CDN does not understand or support a given GenericMetadata property that is not safe-to-redistribute, the CDN MUST set the "incomprehensible" flag to true for that GenericMetadata object before redistributing the metadata. The "incomprehensible" flag signals to a dCDN that the metadata was not properly transformed by the tCDN. A CDN MUST NOT attempt to use metadata that has been marked as "incomprehensible" by a uCDN.

tCDNs MUST NOT change the value of mandatory-to-enforce or safe-to-redistribute when propagating metadata to a dCDN. Although a tCDN can set the value of "incomprehensible" to true, a tCDN MUST NOT change the value of "incomprehensible" from true to false.

Table 2 describes the action to be taken by a tCDN for the different combinations of mandatory-to-enforce ("MtE") and safe-to-redistribute ("StR") properties when the tCDN either does or does not understand the metadata in question:

MtE	StR	Metadata Understood by tCDN	Action
False	True	True	Can serve and redistribute.
False	True	False	Can serve and redistribute.
False	False	False	Can serve. MUST set "incomprehensible" to true when redistributing.
False	False	True	Can serve. Can redistribute after transforming the metadata (if the CDN knows how to do so safely); otherwise, MUST set "incomprehensible" to true when redistributing.
True	True	True	Can serve and redistribute.
True	True	False	MUST NOT serve but can redistribute.
True	False	True	Can serve. Can redistribute after transforming the metadata (if the CDN knows how to do so safely); otherwise, MUST set "incomprehensible" to true when redistributing.
True	False	False	MUST NOT serve. MUST set "incomprehensible" to true when redistributing.

Table 2: Action to Be Taken by a tCDN for the Different Combinations of MtE and StR Properties

Table 3 describes the action to be taken by a dCDN for the different combinations of mandatory-to-enforce and "incomprehensible" (Incomp) properties, when the dCDN either does or does not understand the metadata in question:

MtE	Incomp	Metadata Understood by dCDN	Action
False	False	True	Can serve.
False	True	True	Can serve but MUST NOT interpret/apply any metadata marked as "incomprehensible".
False	False	False	Can serve.
False	True	False	Can serve but MUST NOT interpret/apply any metadata marked as "incomprehensible".
True	False	True	Can serve.
True	True	True	MUST NOT serve.
True	False	False	MUST NOT serve.
True	True	False	MUST NOT serve.

Table 3: Action to Be Taken by a dCDN for the Different Combinations of MtE and Incomp Properties

3.3. Metadata Inheritance and Override

In the metadata object model, a HostMetadata object can contain multiple PathMetadata objects (via PathMatch objects). Each PathMetadata object can in turn contain 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 types LocationACL and TimeWindowACL (where "ACL" means "Access Control List") while a PathMetadata object that applies to "example.com/movies/*" defines an alternate GenericMetadata object of type TimeWindowACL, then:

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

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

A single HostMetadata or PathMetadata object MUST NOT contain multiple GenericMetadata objects of the same type. If an array of GenericMetadata contains objects of duplicate types, the receiver MUST ignore all but the first object of each type.

4. CDNI Metadata Objects

Section 4.1 provides the definitions of each metadata object type introduced in Section 3. These metadata objects are described as structural metadata objects, as they provide the structure for host and URI path-based inheritance and identify which GenericMetadata objects apply to a given User Agent content request.

Section 4.2 provides the definitions for a base set of core metadata objects that can be contained within a GenericMetadata object. These metadata objects govern how User Agent requests for content are handled. GenericMetadata objects can contain other GenericMetadata objects as properties; these can be referred to as sub-objects. As with all CDNI Metadata objects, the value of the GenericMetadata sub-objects can be either a complete serialized representation of the sub-object or a Link object that contains a URI that can be dereferenced to retrieve the complete serialized representation of the property sub-object.

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

dCDNs and tCDNs MUST support the parsing of all CDNI Metadata objects specified in this document. A dCDN does not have to implement the underlying functionality represented by non-structural GenericMetadata objects (though that might restrict the content that a given dCDN will be able to serve). uCDNs as generators of CDNI Metadata only need to support generating the CDNI Metadata that they

need in order to express the policies required by the content they are describing. See Section 6.4 for more details on the specific encoding rules for CDNI Metadata objects.

Note: In the following sections, the term "mandatory-to-specify" is used to convey which properties MUST be included for a given structural or GenericMetadata object. When mandatory-to-specify is specified as "Yes" for an individual property, it means that if the object containing that property is included in a metadata response, then the mandatory-to-specify property MUST also be included (directly or by reference) in the response. For example, a HostMatch property object without a host to match against does not make sense; therefore, the "host" property is mandatory-to-specify inside a HostMatch object.

4.1. Definitions of the CDNI Structural Metadata Objects

The subsections below describe the structural objects introduced in Section 3.1.

4.1.1. HostIndex

The HostIndex object is the entry point into the CDNI Metadata hierarchy. It contains an array of HostMatch objects. An incoming content request is checked against the hostname (or IP address) specified by each of the listed HostMatch objects to find the HostMatch object that applies to the request.

Property: hosts

Description: Array of HostMatch objects. Hosts (HostMatch objects) MUST be evaluated in the order they appear, and the first HostMatch object that matches the content request being processed MUST be used.

Type: Array of HostMatch objects

Mandatory-to-Specify: Yes.

Example HostIndex object containing two HostMatch objects, where the first HostMatch object is embedded and the second HostMatch object is referenced:

```
{
  "hosts": [
    {
      <Properties of embedded HostMatch object>
    },
    {
      "type": "MI.HostMatch",
      "href": "https://metadata.ucdn.example/hostmatch1234"
    }
  ]
}
```

4.1.2. HostMatch

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

Property: host

Description: Hostname or IP address and optional port to match against the requested host, i.e., the host and port as described in [RFC3986]. In order for a hostname or IP address in a content request to match the hostname or IP address in the "host" property, the value from the content request when converted to lowercase MUST be identical to the value of the "host" property when converted to lowercase. All implementations MUST support IPv4 addresses encoded as specified by the "IPv4address" rule in Section 3.2.2 of [RFC3986]. IPv6 addresses MUST be encoded in one of the IPv6 address formats specified in [RFC5952], although receivers MUST support all IPv6 address formats specified in [RFC4291]. Hostnames MUST conform to the Domain Name System (DNS) syntax defined in [RFC1034] and [RFC1123]. Internationalized Domain Names (IDNs) must first be transformed to the A-label form [RFC5890] as per [RFC5891].

Type: Endpoint

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.

Example HostMatch object with an embedded HostMetadata object:

```
{
  "host": "video.example.com",
  "host-metadata": {
    <Properties of embedded HostMetadata object>
  }
}
```

Example HostMatch object referencing (via a Link object; see Section 4.3.1) a HostMetadata object:

```
{
  "host": "video.example.com",
  "host-metadata": {
    "type": "MI.HostMetadata",
    "href": "https://metadata.ucdn.example/host1234"
  }
}
```

4.1.3. HostMetadata

A HostMetadata object contains the CDNI Metadata properties for content served for a particular host (defined in the HostMatch object) and possibly child PathMatch objects.

Property: metadata

Description: Array of host-related metadata.

Type: Array of GenericMetadata objects

Mandatory-to-Specify: Yes.

Property: paths

Description: Path-specific rules. Path patterns (PathMatch objects) MUST be evaluated in the order they appear, and the first (and only the first) PathMatch object that matches the content request being processed MUST be used.

Type: Array of PathMatch objects

Mandatory-to-Specify: No. Default is that there are no more-specific paths to evaluate (i.e., an empty list).

Example HostMetadata object containing a number of embedded GenericMetadata objects that will describe the default metadata for the host and an embedded PathMatch object that contains a path for which metadata exists that overrides the default metadata for the host:

```
{
  "metadata": [
    {
      <Properties of first embedded GenericMetadata object>
    },
    {
      <Properties of second embedded GenericMetadata object>
    },

 ...

    {
      <Properties of Nth embedded GenericMetadata object>
    }
  ],
  "paths": [
    {
      <Properties of embedded PathMatch object>
    }
  ]
}
```

4.1.4. PathMatch

A PathMatch object contains a PatternMatch object with a path to match against a resource's URI path, as well as how to handle URI query parameters. The PathMatch object also contains a PathMetadata object with GenericMetadata to apply if the resource's URI matches the pattern within the PatternMatch object.

Property: path-pattern

Description: Pattern to match against the requested resource's URI.

Type: PatternMatch

Mandatory-to-Specify: Yes.

Property: path-metadata

Description: CDNI Metadata to apply when delivering content that matches the associated PatternMatch.

Type: PathMetadata

Mandatory-to-Specify: Yes.

Example PathMatch object referencing the PathMetadata object to use for URIs that match the case-sensitive URI path pattern "/movies/*" (contained within an embedded PatternMatch object):

```
{
  "path-pattern": {
    "pattern": "/movies/*",
    "case-sensitive": true
  },
  "path-metadata": {
      "type": "MI.PathMetadata",
      "href": "https://metadata.ucdn.example/host1234/pathDCE"
  }
}
```

4.1.5. PatternMatch

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

Property: pattern

Description: A pattern for matching against the URI path, i.e., against the path-absolute [RFC3986]. The pattern can contain the wildcards "*" and "?", where "*" matches any sequence of pchar [RFC3986] or "/" characters (including the empty string) and "?" matches exactly one pchar character. The three literals "$", "*", and "?" MUST be escaped as "$$", "$*", and "$?" (where "$" is the designated escape character). All other characters are treated as literals.

Type: String

Mandatory-to-Specify: Yes.

Property: case-sensitive

Description: Flag indicating whether or not case-sensitive matching should be used. Note: Case insensitivity applies to ALPHA characters in the URI path prior to percent-decoding [RFC3986].

Type: Boolean

Mandatory-to-Specify: No. Default is case-insensitive match (i.e., a value of False).

Example PatternMatch object that matches the case-sensitive URI path pattern "/movies/*":

```
{
  "pattern": "/movies/*",
  "case-sensitive": true
}
```

4.1.6. PathMetadata

A PathMetadata object contains the CDNI Metadata properties for content requests that match against the associated URI path (defined in a PathMatch object).

Note that if DNS-based redirection is employed, then a dCDN will be unable to evaluate any metadata at the PathMetadata level or below because only the hostname of the content request is available at Request Routing time. dCDNs SHOULD still process all PathMetadata for the host before responding to the redirection request to detect if any unsupported metadata is specified. If any metadata not supported by the dCDN is marked as mandatory-to-enforce, the dCDN SHOULD NOT accept the content redirection request, in order to avoid receiving content requests that it will not be able to satisfy/serve.

Property: metadata

Description: Array of path-related metadata.

Type: Array of GenericMetadata objects

Mandatory-to-Specify: Yes.

Property: paths

Description: Path-specific rules. Path patterns (PathMatch objects) MUST be evaluated in the order they appear, and the first (and only the first) PathMatch object that matches the content request being processed MUST be used.

Type: Array of PathMatch objects

Mandatory-to-Specify: No. Default is that there are no more-specific paths to evaluate (i.e., an empty list).

Example PathMetadata object containing a number of embedded GenericMetadata objects that describe the metadata to apply for the URI path defined in the parent PathMatch object, as well as a more-specific PathMatch object.

```
{
  "metadata": [
    {
      <Properties of first embedded GenericMetadata object>
    },
    {
      <Properties of second embedded GenericMetadata object>
    },

 ...

    {
      <Properties of Nth embedded GenericMetadata object>
    }
  ],
  "paths": [
    {
      <Properties of embedded PathMatch object>
    }
  ]
}
```

4.1.7. GenericMetadata

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

Property: generic-metadata-type

Description: Case-insensitive CDNI Metadata object type.

Type: String containing the CDNI Payload Type [RFC7736] of the object contained in the generic-metadata-value property (see Table 4).

Mandatory-to-Specify: Yes.

Property: generic-metadata-value

Description: CDNI Metadata object.

Type: Format/Type is defined by the value of the generic-metadata-type property above.
Note: generic-metadata-values MUST NOT name any properties "href" (see Section 4.3.1).

Mandatory-to-Specify: Yes.

Property: mandatory-to-enforce

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

Type: Boolean

Mandatory-to-Specify: No. Default is to treat metadata as mandatory-to-enforce (i.e., a value of True).

Property: safe-to-redistribute

Description: Flag identifying whether or not the property metadata can be safely redistributed without modification.

Type: Boolean

Mandatory-to-Specify: No. Default is to allow transparent redistribution (i.e., a value of True).

Property: incomprehensible

Description: Flag identifying whether or not any CDN in the chain of delegation has failed to understand and/or failed to properly transform this metadata object. Note: This flag only applies to metadata objects whose safe-to-redistribute property has a value of False.

Type: Boolean

Mandatory-to-Specify: No. Default is comprehensible (i.e., a value of False).

Example GenericMetadata object containing a metadata object that applies to the applicable URI path and/or host (within a parent PathMetadata and/or HostMetadata object, respectively):

```
{
  "mandatory-to-enforce": true,
  "safe-to-redistribute": true,
  "incomprehensible": false,
  "generic-metadata-type": <CDNI Payload Type of this metadata object>,
  "generic-metadata-value":
    {
      <Properties of this metadata object>
    }
}
```

4.2. Definitions of the Initial Set of CDNI GenericMetadata Objects

The objects defined below are intended to be used in the GenericMetadata object's generic-metadata-value field as defined in Section 4.1.7, and their generic-metadata-type property MUST be set to the appropriate CDNI Payload Type as defined in Table 4.

4.2.1. SourceMetadata

Source metadata provides the dCDN with information about content acquisition, i.e., how to contact a 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 Content Service Provider (CSP) but might be a set of Surrogates in the uCDN.

Property: sources

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

Type: Array of Source objects (see Section 4.2.1.1)

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

Example SourceMetadata object (which contains two Source objects) that describes which servers the dCDN should use for acquiring content for the applicable URI path and/or host:

```
{
  "generic-metadata-type": "MI.SourceMetadata",
  "generic-metadata-value":
    {
      "sources": [
        {
          "endpoints": [
            "a.service123.ucdn.example",
            "b.service123.ucdn.example"
            ],
          "protocol": "http/1.1"
        },
        {
          "endpoints": ["origin.service123.example"],
          "protocol": "http/1.1"
        }
      ]
    }
}
```

4.2.1.1. Source

A Source object describes the source to 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 to be used when contacting that source.

Endpoints within a Source object MUST be treated as equivalent/equal. A uCDN can specify an array of sources, ordered by preference, within a SourceMetadata object. Then, for each Source object ranked by preference, a uCDN can specify an array of endpoints that are equivalent (e.g., a pool of servers that are not behind a load balancer).

Property: acquisition-auth

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

Type: Auth (see Section 4.2.7)

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

Property: endpoints

Description: Origins from which the dCDN can acquire content. If multiple endpoints are specified, they are all equal, i.e., the list is not ordered by preference.

Type: Array of Endpoint objects (see Section 4.3.3)

Mandatory-to-Specify: Yes.

Property: protocol

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

Type: Protocol (see Section 4.3.2)

Mandatory-to-Specify: Yes.

Example Source object that describes a pair of endpoints (servers) the dCDN can use for acquiring content for the applicable host and/or URI path:

```
{
  "endpoints": [
    "a.service123.ucdn.example",
    "b.service123.ucdn.example"
  ],
  "protocol": "http/1.1"
}
```

4.2.2. LocationACL Metadata

LocationACL metadata defines which locations a User Agent needs to be in, in order to be able to receive the associated content.

A LocationACL that does not include a "locations" property results in an action of "allow all", meaning that delivery can be performed regardless of the User Agent's location; otherwise, a CDN MUST take the action from the first footprint to match against the User Agent's location. If two or more footprints overlap, the first footprint that matches against the User Agent's location determines the action

a CDN MUST take. If the "locations" property is included but is empty or if none of the listed footprints match the User Agent's location, then the result is an action of "deny".

Although the LocationACL, TimeWindowACL (see Section 4.2.3), and ProtocolACL (see Section 4.2.4) are independent GenericMetadata objects, they can provide conflicting information to a dCDN, e.g., a content request that is simultaneously allowed based on the LocationACL and denied based on the TimeWindowACL. The dCDN MUST use the logical AND of all ACLs (where "allow" is true and "deny" is false) to determine whether or not a request should be allowed.

Property: locations

Description: ACL that allows or denies (blocks) delivery based on the User Agent's location.

Type: Array of LocationRule objects (see Section 4.2.2.1)

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

Example LocationACL object that allows the dCDN to deliver content to any location / IP address:

```
{
  "generic-metadata-type": "MI.LocationACL",
  "generic-metadata-value":
    {
    }
}
```

Example LocationACL object (which contains a LocationRule object that in turn contains a Footprint object) that only allows the dCDN to deliver content to User Agents in the USA:

```
{
  "generic-metadata-type": "MI.LocationACL",
  "generic-metadata-value":
    {
      "locations": [
        {
          "action": "allow",
          "footprints": [
            {
              "footprint-type": "countrycode",
              "footprint-value": ["us"]
            }
          ]
        }
      ]
    }
}
```

4.2.2.1. LocationRule

A LocationRule contains or references an array of Footprint objects and the corresponding action.

Property: footprints

Description: Array of footprints to which the rule applies.

Type: Array of Footprint objects (see Section 4.2.2.2)

Mandatory-to-Specify: Yes.

Property: action

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

Type: Enumeration [allow|deny] encoded as a lowercase string

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

Example LocationRule object (which contains a Footprint object) that allows the dCDN to deliver content to clients in the USA:

```
{
  "action": "allow",
  "footprints": [
    {
      "footprint-type": "countrycode",
      "footprint-value": ["us"]
    }
  ]
}
```

4.2.2.2. Footprint

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

Property: footprint-type

Description: Registered footprint type (see Section 7.2). The footprint types specified by this document are "ipv4cidr" (IPv4CIDR; see Section 4.3.5), "ipv6cidr" (IPv6CIDR; see Section 4.3.6), "asn" (Autonomous System Number; see Section 4.3.7), and "countrycode" (Country Code; see Section 4.3.8).

Type: Lowercase string

Mandatory-to-Specify: Yes.

Property: footprint-value

Description: Array of footprint values conforming to the specification associated with the registered footprint type. Footprint values can be simple strings (e.g., IPv4CIDR, IPv6CIDR, ASN, and Country Code); however, other Footprint objects can be defined in the future, along with a more complex encoding (e.g., GPS coordinate tuples).

Type: Array of footprints

Mandatory-to-Specify: Yes.

Example Footprint object describing a footprint covering the USA:

```
{
  "footprint-type": "countrycode",
  "footprint-value": ["us"]
}
```

Example Footprint object describing a footprint covering the IP address ranges 192.0.2.0/24 and 198.51.100.0/24:

```
{
  "footprint-type": "ipv4cidr",
  "footprint-value": ["192.0.2.0/24", "198.51.100.0/24"]
}
```

Example Footprint object describing a footprint covering the IP address ranges 2001:db8::/32:

```
{
  "footprint-type": "ipv6cidr",
  "footprint-value": ["2001:db8::/32"]
}
```

Example Footprint object describing a footprint covering the autonomous system 64496:

```
{
  "footprint-type": "asn",
  "footprint-value": ["as64496"]
}
```

4.2.3. TimeWindowACL

TimeWindowACL metadata defines time-based restrictions.

A TimeWindowACL that does not include a "times" property results in an action of "allow all", meaning that delivery can be performed regardless of the time of the User Agent's request; otherwise, a CDN MUST take the action from the first window to match against the current time. If two or more windows overlap, the first window that matches against the current time determines the action a CDN MUST take. If the "times" property is included but is empty or if none of the listed windows match the current time, then the result is an action of "deny".

Although the LocationACL (see Section 4.2.2), TimeWindowACL, and ProtocolACL (see Section 4.2.4) are independent GenericMetadata objects, they can provide conflicting information to a dCDN, e.g.,

a content request that is simultaneously allowed based on the LocationACL and denied based on the TimeWindowACL. The dCDN MUST use the logical AND of all ACLs (where "allow" is true and "deny" is false) to determine whether or not a request should be allowed.

Property: times

Description: ACL that allows or denies (blocks) delivery based on the time of a User Agent's request.

Type: Array of TimeWindowRule objects (see Section 4.2.3.1)

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

Example TimeWindowACL object (which contains a TimeWindowRule object that in turn contains a TimeWindow object) that only allows the dCDN to deliver content to clients between 09:00 01/01/2000 UTC and 17:00 01/01/2000 UTC:

```
{
  "generic-metadata-type": "MI.TimeWindowACL",
  "generic-metadata-value":
    {
      "times": [
        {
          "action": "allow",
          "windows": [
            {
              "start": 946717200,
              "end": 946746000
            }
          ]
        }
      ]
    }
}
```

4.2.3.1. TimeWindowRule

A TimeWindowRule contains or references an array of TimeWindow objects and the corresponding action.

Property: windows

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

Type: Array of TimeWindow objects (see Section 4.2.3.2)

Mandatory-to-Specify: Yes.

Property: action

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

Type: Enumeration [allow|deny] encoded as a lowercase string

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

Example TimeWindowRule object (which contains a TimeWindow object) that only allows the dCDN to deliver content to clients between 09:00 01/01/2000 UTC and 17:00 01/01/2000 UTC:

```
{
  "action": "allow",
  "windows": [
    {
      "start": 946717200,
      "end": 946746000
    }
  ]
}
```

4.2.3.2. TimeWindow

A TimeWindow object describes a time range that can be applied by a TimeWindowACL, e.g., start 946717200 (i.e., 09:00 01/01/2000 UTC), end: 946746000 (i.e., 17:00 01/01/2000 UTC).

Property: start

Description: The start time of the window.

Type: Time (see Section 4.3.4)

Mandatory-to-Specify: Yes.

Property: end

Description: The end time of the window.

Type: Time (see Section 4.3.4)

Mandatory-to-Specify: Yes.

Example TimeWindow object that describes a time window from 09:00 01/01/2000 UTC to 17:00 01/01/2000 UTC:

```
{
  "start": 946717200,
  "end": 946746000
}
```

4.2.4. ProtocolACL Metadata

ProtocolACL metadata defines delivery protocol restrictions.

A ProtocolACL that does not include a protocol-acl property results in an action of "allow all", meaning that delivery can be performed regardless of the protocol in the User Agent's request; otherwise, a CDN MUST take the action from the first protocol to match against the request protocol. If two or more request protocols overlap, the first protocol that matches the request protocol determines the action a CDN MUST take. If the protocol-acl property is included but is empty or if none of the listed protocols match the request protocol, then the result is an action of "deny".

Although the LocationACL (see Section 4.2.2), TimeWindowACL (see Section 4.2.3), and ProtocolACL are independent GenericMetadata objects, they can provide conflicting information to a dCDN, e.g., a content request that is simultaneously allowed based on the ProtocolACL and denied based on the TimeWindowACL. The dCDN MUST use the logical AND of all ACLs (where "allow" is true and "deny" is false) to determine whether or not a request should be allowed.

Property: protocol-acl

Description: ACL that allows or denies (blocks) delivery based on delivery protocol.

Type: Array of ProtocolRule objects (see Section 4.2.4.1)

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

Example ProtocolACL object (which contains a ProtocolRule object) that only allows the dCDN to deliver content using HTTP/1.1:

```
{
  "generic-metadata-type": "MI.ProtocolACL",
  "generic-metadata-value":
    {
      "protocol-acl": [
        {
          "action": "allow",
          "protocols": ["http/1.1"]
        }
      ]
    }
}
```

4.2.4.1. ProtocolRule

A ProtocolRule contains or references an array of Protocol objects and the corresponding action.

Property: protocols

Description: Array of protocols to which the rule applies.

Type: Array of Protocol objects (see Section 4.3.2)

Mandatory-to-Specify: Yes.

Property: action

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

Type: Enumeration [allow|deny] encoded as a lowercase string

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

Example ProtocolRule object (which contains a Protocol object) that allows the dCDN to deliver content using HTTP/1.1:

```
{
  "action": "allow",
  "protocols": ["http/1.1"]
}
```

4.2.5. DeliveryAuthorization Metadata

Delivery authorization defines authorization methods for the delivery of content to User Agents.

Property: delivery-auth-methods

Description: Options for authorizing content requests. Delivery for a content request is authorized if any one of the authorization methods in the list is satisfied for that request.

Type: Array of Auth objects (see Section 4.2.7)

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

Example DeliveryAuthorization object (which contains an Auth object):

```
{
  "generic-metadata-type": "MI.DeliveryAuthorization",
  "generic-metadata-value":
    {
      "delivery-auth-methods": [
        {
          "auth-type": <CDNI Payload Type of this Auth object>,
          "auth-value":
          {
            <Properties of this Auth object>
          }
        }
      ]
    }
}
```

4.2.6. Cache

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

Cache keys are generated from the URI of the content request [RFC7234]. In some cases, a CDN or content provider might want certain path segments or query parameters to be excluded from the cache key generation. The Cache object provides guidance on what parts of the path and query string to include.

Property: exclude-path-pattern

Description: A pattern for matching against the URI path, i.e., against the path-absolute [RFC3986]. The pattern can contain the wildcards "*" and "?", where "*" matches any sequence of pchar [RFC3986] or "/" characters (including the empty string) and "?" matches exactly one pchar character. The three literals "$", "*", and "?" MUST be escaped as "$$", "$*", and "$?" (where "$" is the designated escape character). All other characters are treated as literals. Cache key generation MUST only include the portion of the path-absolute that matches the wildcard portions of the pattern. Note: Inconsistency between the PatternMatch pattern (Section 4.1.5) and the exclude-path-pattern can result in inefficient caching.

Type: String

Mandatory-to-Specify: No. Default is to use the full URI path-absolute to generate the cache key.

Property: include-query-strings

Description: Allows a Surrogate to specify the URI query string parameters [RFC3986] to include when comparing the requested URI against the URIs in its cache for equivalence. Matching query parameters MUST be case insensitive. If all query parameters should be ignored, then the list MUST be specified and MUST be empty. If a query parameter appears multiple times in the query string, each instance value MUST be aggregated prior to comparison. For consistent cache key generation, query parameters SHOULD be evaluated in the order specified in this array.

Type: Array of strings

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

Example Cache object that instructs the dCDN to use the full URI path and ignore all query parameters:

```
{
  "generic-metadata-type": "MI.Cache",
  "generic-metadata-value":
  {
    "include-query-strings": []
  }
}
```

Example Cache object that instructs the dCDN to exclude the "CDNX" path prefix and only include the (case-insensitive) query parameters named "mediaid" and "providerid":

```
{
  "generic-metadata-type": "MI.Cache",
  "generic-metadata-value":
  {
    "exclude-path-pattern": "/CDNX/*",
    "include-query-strings": ["mediaid", "providerid"]
  }
}
```

Example Cache object that instructs the dCDN to exclude the "CDNX" path prefix but includes all query parameters:

```
{
  "generic-metadata-type": "MI.Cache",
  "generic-metadata-value":
  {
    "exclude-path-pattern": "/CDNX/*"
  }
}
```

4.2.7. Auth

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

Note: This document does not define any Auth methods. Individual Auth methods are being defined separately (e.g., URI Signing [CDNI-URI-SIGNING]). The GenericMetadata object that contains Auth objects is defined herein for convenience and so as not to be specific to any particular Auth method.

Property: auth-type

Description: Auth type (The CDNI Payload Type [RFC7736] of the GenericMetadata object contained in the auth-value property).

Type: String

Mandatory-to-Specify: Yes.

Property: auth-value

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

Type: GenericMetadata object

Mandatory-to-Specify: Yes.

Example Auth object:

```
{
  "generic-metadata-type": "MI.Auth",
  "generic-metadata-value":
  {
    "auth-type": <CDNI Payload Type of this Auth object>,
    "auth-value":
      {
        <Properties of this Auth object>
      }
  }
}
```

4.2.8. Grouping

A Grouping object identifies a group of content to which a given asset belongs.

Property: ccid

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

Type: String

Mandatory-to-Specify: No. Default is not to apply any grouping.

Example Grouping object that specifies a Content Collection IDentifier for the content associated with the Grouping object's parent HostMetadata and PathMetadata:

```
{
  "generic-metadata-type": "MI.Grouping",
  "generic-metadata-value":
  {
    "ccid": "ABCD"
  }
}
```

4.3. CDNI Metadata Simple Data Type Descriptions

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

4.3.1. Link

A Link object can be used in place of any of the objects described above. Link objects can be used to avoid duplication if the same metadata information is repeated within the metadata tree. When a Link object replaces another object, its "href" property is set to the URI of the resource and its "type" property is set to the CDNI Payload Type of the object it is replacing.

dCDNs can detect the presence of a Link object by detecting the presence of a property named "href" within the object. This means that GenericMetadata types MUST NOT contain a property named "href" because doing so would conflict with the ability for dCDNs to detect Link objects being used to reference a GenericMetadata object.

Property: href

Description: The URI of the addressable object being referenced.

Type: String

Mandatory-to-Specify: Yes.

Property: type

Description: The CDNI Payload Type of the object being referenced.

Type: String

Mandatory-to-Specify: No. If the container specifies the type (e.g., the HostIndex object contains an array of HostMatch objects, so a Link object in the list of HostMatch objects must reference a HostMatch), then it is not necessary to explicitly specify a type.

Example Link object referencing a HostMatch object:

```
{
  "type": "MI.HostMatch",
  "href": "https://metadata.ucdn.example/hostmatch1234"
}
```

Example Link object referencing a HostMatch object, without an explicit type, inside a HostIndex object:

```
{
  "hosts": [
    {
      <Properties of embedded HostMatch object>
    },
    {
      "href": "https://metadata.ucdn.example/hostmatch1234"
    }
  ]
}
```

4.3.1.1. Link Loop Prevention

When following a link, CDNI Metadata clients SHOULD verify that the CDNI Payload Type of the object retrieved matches the expected CDNI Payload Type, as indicated by the Link object or containing property. For GenericMetadata objects, type checks will prevent self-references; however, incorrect linking can result in circular references for structural metadata objects, specifically PathMatch and PathMetadata objects (Figure 1). To prevent circular references, CDNI Metadata clients SHOULD verify that no duplicate links occur for PathMatch or PathMetadata objects.

4.3.2. Protocol

Protocol objects are used to specify protocols (from the "CDNI Metadata Protocol Types" registry; see Section 7.3) for content acquisition or delivery.

Type: String

Example:

"http/1.1"

4.3.3. Endpoint

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

All implementations MUST support IPv4 addresses encoded as specified by the "IPv4address" rule in Section 3.2.2 of [RFC3986]. IPv6 addresses MUST be encoded in one of the IPv6 address formats specified in [RFC5952], although receivers MUST support all IPv6 address formats specified in [RFC4291]. Hostnames MUST conform to

the Domain Name System (DNS) syntax defined in [RFC1034] and [RFC1123]. Internationalized Domain Names (IDNs) must first be transformed to the A-label form [RFC5890] as per [RFC5891].

Type: String

Example hostname:

"metadata.ucdn.example"

Example IPv4 address:

"192.0.2.1"

Example IPv6 address (with port number):

"[2001:db8::1]:81"

4.3.4. Time

A time value expressed in seconds since the UNIX epoch (i.e., zero hours, zero minutes, zero seconds, on January 1, 1970) Coordinated Universal Time (UTC) [POSIX].

Type: Integer

Example time representing 09:00:00 01/01/2000 UTC:

946717200

4.3.5. IPv4CIDR

An IPv4address Classless Inter-Domain Routing (CIDR) block encoded as specified by the "IPv4address" rule in Section 3.2.2 of [RFC3986] followed by a "/" followed by an unsigned integer representing the leading bits of the routing prefix (i.e., IPv4 CIDR notation). Single IP addresses can be expressed as /32.

Type: String

Example IPv4CIDR:

"192.0.2.0/24"

4.3.6. IPv6CIDR

An IPv6address CIDR block encoded in one of the IPv6 address formats specified in [RFC5952] followed by a "/" followed by an unsigned integer representing the leading bits of the routing prefix (i.e., IPv6 CIDR notation). Single IP addresses can be expressed as /128.

Type: String

Example IPv6CIDR:

"2001:db8::/32"

4.3.7. ASN

An ASN value encoded as a string consisting of the characters "as" (in lowercase) followed by the ASN [RFC6793].

Type: String

Example ASN:

"as64496"

4.3.8. Country Code

An ISO 3166-1 alpha-2 code [ISO3166-1] in lowercase.

Type: String

Example Country Code representing the USA:

"us"

5. CDNI Metadata Capabilities

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

the dCDN, any delegated requests for that content can fail, so again the uCDN is likely to want to avoid delegating those requests to that dCDN.

The CDNI Footprint & Capabilities Advertisement interface (FCI) provides a means of advertising capabilities from the dCDN to the uCDN [RFC8008]. Support for optional metadata types and values can be advertised using the FCI.

6. CDNI Metadata Interface

This section specifies an interface to enable a dCDN to retrieve CDNI Metadata objects from a uCDN.

The interface can be used by a dCDN to retrieve CDNI Metadata objects in either of two ways:

- Dynamically, as required by the dCDN to process received requests -- for example, in response to a query from a uCDN over the CDNI Request Routing Redirection interface (RI) [RFC7975] or in response to receiving a request for content from a User Agent.

- In advance of being required -- for example, in the case of pre-positioned CDNI Metadata acquisition, initiated through the "CDNI Control interface / Triggers" (CI/T) interface [RFC8007].

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

CDNI Metadata servers (i.e., servers in the uCDN) are free to assign whatever structure they desire to the URIs for CDNI Metadata objects, and CDNI Metadata clients MUST NOT make any assumptions regarding the structure of CDNI Metadata URIs or the mapping between CDNI Metadata objects and their associated URIs. Any URIs present in the examples in this document 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 [RFC7230].

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

The corresponding HTTP response returns the status of the operation in the HTTP status code and returns the current representation of the resource (if appropriate) in the response body. HTTP responses that contain a response body SHOULD include an entity-tag (ETag) to enable validation of cached versions of returned resources.

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

6.2. Retrieval of CDNI Metadata Resources

In the general case, a CDNI Metadata server makes CDNI Metadata objects available via unique URIs; thus, in order to retrieve CDNI Metadata, a CDNI Metadata client (i.e., a client in the dCDN) first makes an HTTP GET request for the URI of the HostIndex, which provides an array of hostnames for which the uCDN can delegate content delivery to the dCDN.

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 listed in the HostMatch objects). If the HostMetadata is linked (rather than embedded), the CDNI Metadata client then makes an HTTP 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 (by matching the URI path in the request against the path-pattern property items in any PathMatch objects listed in the HostMetadata object). If a PathMetadata object is found to match (and is linked rather than embedded), the CDNI Metadata client makes another HTTP GET request for the PathMetadata. Each PathMetadata object can also include

references to additional more-specific metadata. If this is the case, the CDNI Metadata client continues requesting PathMatch and PathMetadata objects recursively. The CDNI Metadata client repeats this approach of processing metadata objects and retrieving (via HTTP GETs) any linked objects until it has all the metadata objects it requires in order to process the redirection request from the uCDN or the content request from a User Agent.

In cases where a dCDN is not able to retrieve the entire set of CDNI Metadata associated with a User Agent request, or it has retrieved that metadata but it is stale according to standard HTTP caching rules and cannot be revalidated -- for example, because the uCDN is unreachable or returns an HTTP 4xx or 5xx status in response to some or all of the dCDN's CDNI Metadata requests -- the dCDN MUST NOT serve the requested content.

Where a dCDN is interconnected with multiple uCDNs, the dCDN needs to determine which uCDN's CDNI Metadata interface should be used to handle a particular User Agent request.

When HTTP redirection (e.g., HTTP 302 redirects) is being used between CDNs, it is expected that the dCDN will be able to determine the uCDN that redirected a particular request from information contained in the received request (e.g., via the URI). With knowledge of which uCDN routed the request, the dCDN can choose the correct uCDN from which to obtain the HostIndex. Note that the HostIndexes served by each uCDN can 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 uCDN that redirected a particular request (e.g., when content from a given host is redirected to a given dCDN by more than one uCDN); therefore, dCDNs will have to apply local policy when deciding which uCDN's CDNI Metadata interface to use.

6.3. Bootstrapping

The URI for the HostIndex object of a given uCDN needs to be configured in the dCDN. All other objects/resources are then discoverable from the HostIndex object by following any links in the HostIndex object, and through the referenced HostMetadata and PathMetadata objects and their GenericMetadata sub-objects.

Manual configuration of the URI for the HostIndex object is outside the scope of this document.

6.4. Encoding

CDNI Metadata objects MUST be encoded as I-JSON objects [RFC7493] 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 CDNI Payload Type of the returned resource). Likewise, the values associated with each property (dictionary key) are dependent on the specific object being encoded (i.e., dependent on the CDNI Payload Type of the returned resource).

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

6.5. Extensibility

The set of GenericMetadata objects can be extended with additional (standards-based or vendor-specific) metadata objects through the specification of new GenericMetadata objects. The GenericMetadata object defined in Section 4.1.7 specifies a type field and a type-specific value field that allow any metadata to be included in either the HostMetadata or PathMetadata arrays.

As with the initial GenericMetadata types defined in Section 4.2, future GenericMetadata types MUST specify the information necessary for constructing and decoding the GenericMetadata object.

Any document that defines a new GenericMetadata type MUST:

1. Register the CDNI Payload Type [RFC7736] used to identify the new GenericMetadata type being specified.

2. Define the set of properties associated with the new GenericMetadata object. GenericMetadata MUST NOT contain a property named "href" because doing so would conflict with the ability to detect Link objects (see Section 4.3.1).

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

4. Describe the semantics of the new type, including its purpose, and provide a use case to which it applies, including an example encoded in I-JSON.

5. Describe the security and privacy consequences, for both the User Agent and the CDNs, of the new GenericMetadata object.

6. Describe any relation to, conflict with, or obsolescence of other existing CDNI Metadata objects.

Note: In the case of vendor-specific extensions, vendor-identifying CDNI Payload Type names will decrease the possibility of GenericMetadata type collisions. It is RECOMMENDED that any vendor-specific extensions use vendor-identifying CDNI Payload Type names.

6.6. Metadata Enforcement

At any given time, the set of GenericMetadata types supported by the uCDN might not match the set of GenericMetadata types supported by the dCDN.

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

Note: Ideally, uCDNs would not delegate content requests to a dCDN that 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 FCI or through out-of-band negotiation between CDN operators), metadata support can fluctuate or be inconsistent (e.g., due to miscommunication, misconfiguration, or temporary outage). Thus, the dCDN MUST always evaluate all metadata associated with redirection and content requests and reject any requests where mandatory-to-enforce metadata associated with the content cannot be enforced.

6.7. Metadata Conflicts

It is possible that new metadata definitions will obsolete or conflict with existing GenericMetadata (e.g., a future revision of the CDNI Metadata interface could redefine the Auth GenericMetadata object or a custom vendor extension could implement an alternate Auth metadata option). If multiple metadata (e.g., MI.Auth.v2, vendor1.Auth, and vendor2.Auth) all conflict with an existing GenericMetadata object (i.e., MI.Auth) and all are marked as

mandatory-to-enforce, it could be ambiguous as to which metadata should be applied, especially in the case of overlapping functionality.

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 GenericMetadata types. It is the responsibility of the CSP and the CDN operators to ensure that metadata assigned to a given piece of content do not conflict.

Note: Because metadata is inherently ordered in HostMetadata and PathMetadata arrays, as well as in the PathMatch hierarchy, multiple conflicting metadata types MAY be used; however, metadata hierarchies SHOULD ensure that independent PathMatch root objects are used to prevent ambiguous or conflicting metadata definitions.

6.8. Versioning

The version of CDNI Metadata objects is conveyed inside the CDNI Payload Type that is included in either (1) the HTTP Content-Type header (for example, "Content-Type: application/cdni; ptype=MI.HostIndex" when retrieved via a link) or (2) in the link type (Section 4.3.1), generic-metadata-type (Section 4.1.7), or auth-type (Section 4.2.7) properties in the JSON payload. The CDNI Payload Type uniquely identifies the specification defining that object, including any relation to, conflicts with, or obsolescence of other metadata. There is no explicit version mapping requirement; however, for ease of understanding, metadata creators SHOULD make new versions of metadata easily visible via the CDNI Payload Type, e.g., by appending a version string. Note: A version string is optional on the first version (e.g., MI.HostIndex) but could be added for subsequent versions (MI.HostIndex.v2, MI.HostIndex.v3, etc.).

Except when referenced by a Link object, nested metadata objects (i.e., structural metadata below the HostIndex; and Source, LocationRule, TimeWindowRule, ProtocolRule, Footprint, and TimeWindow objects) can be serialized into a JSON payload without explicit CDNI Payload Type information. The type is inferred from the outer structural metadata, GenericMetadata, or Auth object CDNI Payload Type. To avoid ambiguity when revising nestable metadata objects, any outer metadata object(s) MUST be reversioned and allocated new CDNI Payload Type(s) at the same time. For example, the MI.HostIndex object defined in this document contains an array of MI.HostMatch objects, each of which in turn contains a MI.HostMetadata object. If a new MI.HostMetadata.v2 object were required, the outer MI.HostIndex and MI.HostMatch objects would need to be revised, e.g., to

MI.HostIndex.v2 and MI.HostMatch.v2, respectively. Similarly, if a new MI.TimeWindowRule.v2 object were required, the outer MI.TimeWindowACL object would need to be revised, e.g., to MI.TimeWindowACL.v2; however, the MI.TimeWindowRule.v2 object could still contain MI.TimeWindow objects, if so specified.

HTTP requests sent to a metadata server SHOULD include an Accept header with the CDNI Payload Type of the expected object. Metadata clients can specify multiple CDNI Payload Types in the Accept header; for example, if a metadata client is capable of processing two different versions of the same type of object (defined by different CDNI Payload Types), it might decide to include both in the Accept header.

6.9. Media Types

All CDNI Metadata objects use the media type "application/cdni". The CDNI Payload Type for each object then contains the object name of that object as defined by this document, prefixed with "MI.". Table 4 lists the CDNI Payload Types for the metadata objects (resources) specified in this document.

Data Object	CDNI Payload Type
HostIndex	MI.HostIndex
HostMatch	MI.HostMatch
HostMetadata	MI.HostMetadata
PathMatch	MI.PathMatch
PatternMatch	MI.PatternMatch
PathMetadata	MI.PathMetadata
SourceMetadata	MI.SourceMetadata
Source	MI.Source
LocationACL	MI.LocationACL
LocationRule	MI.LocationRule
Footprint	MI.Footprint
TimeWindowACL	MI.TimeWindowACL
TimeWindowRule	MI.TimeWindowRule
TimeWindow	MI.TimeWindow
ProtocolACL	MI.ProtocolACL
ProtocolRule	MI.ProtocolRule
DeliveryAuthorization	MI.DeliveryAuthorization
Cache	MI.Cache
Auth	MI.Auth
Grouping	MI.Grouping

Table 4: CDNI Payload Types for CDNI Metadata Objects

6.10. Complete CDNI Metadata Example

A dCDN requests the HostIndex and receives the following object with a CDNI Payload Type of "MI.HostIndex":

```
{
  "hosts": [
    {
      "host": "video.example.com",
      "host-metadata": {
        "type": "MI.HostMetadata",
        "href": "https://metadata.ucdn.example/host1234"
      }
    },
    {
      "host": "images.example.com",
      "host-metadata": {
        "type": "MI.HostMetadata",
        "href": "https://metadata.ucdn.example/host5678"
      }
    }
  ]
}
```

If the incoming request has a Host header with "video.example.com", then the dCDN would fetch the HostMetadata object from "https://metadata.ucdn.example/host1234" expecting a CDNI Payload Type of "MI.HostMetadata":

```
{
  "metadata": [
    {
      "generic-metadata-type": "MI.SourceMetadata",
      "generic-metadata-value": {
        "sources": [
          {
            "endpoint": ["acq1.ucdn.example"],
            "protocol": "http/1.1"
          },
          {
            "endpoint": ["acq2.ucdn.example"],
            "protocol": "http/1.1"
          }
        ]
      }
    },
```

```
    {
      "generic-metadata-type": "MI.LocationACL",
      "generic-metadata-value": {
        "locations": [
          {
            "footprints": [
              {
                "footprint-type": "ipv4cidr",
                "footprint-value": ["192.0.2.0/24"]
              },
              {
                "footprint-type": "ipv6cidr",
                "footprint-value": ["2001:db8::/32"]
              },
              {
                "footprint-type": "countrycode",
                "footprint-value": ["us"]
              },
              {
                "footprint-type": "asn",
                "footprint-value": ["as64496"]
              }
            ],
            "action": "deny"
          }
        ]
      }
    },
    {
      "generic-metadata-type": "MI.ProtocolACL",
      "generic-metadata-value": {
        "protocol-acl": [
          {
            "protocols": [
              "http/1.1"
            ],
            "action": "allow"
          }
        ]
      }
    }
  ],
```

```
    "paths": [
      {
        "path-pattern": {
          "pattern": "/videos/trailers/*"
        },
        "path-metadata": {
          "type": "MI.PathMetadata",
          "href": "https://metadata.ucdn.example/host1234/pathABC"
        }
      },
      {
        "path-pattern": {
          "pattern": "/videos/movies/*"
        },
        "path-metadata": {
          "type": "MI.PathMetadata",
          "href": "https://metadata.ucdn.example/host1234/pathDEF"
        }
      }
    ]
  }
```

Suppose that the path of the requested resource matches the "/videos/movies/*" pattern; the next metadata requested would be for "https://metadata.ucdn.example/host1234/pathDEF" with an expected CDNI Payload Type of "MI.PathMetadata":

```
  {
    "metadata": [],
    "paths": [
      {
        "path-pattern": {
          "pattern": "/videos/movies/hd/*"
        },
        "path-metadata": {
          "type": "MI.PathMetadata",
          "href":
            "https://metadata.ucdn.example/host1234/pathDEF/path123"
        }
      }
    ]
  }
```

Finally, if the path of the requested resource also matches the "/videos/movies/hd/*" pattern, the dCDN would also fetch the following object from "https://metadata.ucdn.example/host1234/pathDEF/path123" with a CDNI Payload Type of "MI.PathMetadata":

```
{
  "metadata": [
    {
      "generic-metadata-type": "MI.TimeWindowACL",
      "generic-metadata-value": {
        "times": [
          "windows": [
            {
              "start": "1213948800",
              "end": "1478047392"
            }
          ],
          "action": "allow"
        ]
      }
    }
  ]
}
```

The final set of metadata that applies to the requested resource includes a SourceMetadata, a LocationACL, a ProtocolACL, and a TimeWindowACL.

7. IANA Considerations

7.1. CDNI Payload Types

This document requests the registration of the following entries under the "CDNI Payload Types" registry hosted by IANA:

Payload Type	Specification
MI.HostIndex	RFC 8006
MI.HostMatch	RFC 8006
MI.HostMetadata	RFC 8006
MI.PathMatch	RFC 8006
MI.PatternMatch	RFC 8006
MI.PathMetadata	RFC 8006
MI.SourceMetadata	RFC 8006
MI.Source	RFC 8006
MI.LocationACL	RFC 8006
MI.LocationRule	RFC 8006
MI.Footprint	RFC 8006
MI.TimeWindowACL	RFC 8006
MI.TimeWindowRule	RFC 8006
MI.TimeWindow	RFC 8006
MI.ProtocolACL	RFC 8006
MI.ProtocolRule	RFC 8006
MI.DeliveryAuthorization	RFC 8006
MI.Cache	RFC 8006
MI.Auth	RFC 8006
MI.Grouping	RFC 8006

7.1.1. CDNI MI HostIndex Payload Type

Purpose: The purpose of this Payload Type is to distinguish HostIndex MI objects (and any associated capability advertisement)

Interface: MI/FCI

Encoding: see Section 4.1.1

7.1.2. CDNI MI HostMatch Payload Type

Purpose: The purpose of this Payload Type is to distinguish HostMatch MI objects (and any associated capability advertisement)

Interface: MI/FCI

Encoding: see Section 4.1.2

7.1.3. CDNI MI HostMetadata Payload Type

Purpose: The purpose of this Payload Type is to distinguish HostMetadata MI objects (and any associated capability advertisement)

Interface: MI/FCI

Encoding: see Section 4.1.3

7.1.4. CDNI MI PathMatch Payload Type

Purpose: The purpose of this Payload Type is to distinguish PathMatch MI objects (and any associated capability advertisement)

Interface: MI/FCI

Encoding: see Section 4.1.4

7.1.5. CDNI MI PatternMatch Payload Type

Purpose: The purpose of this Payload Type is to distinguish PatternMatch MI objects (and any associated capability advertisement)

Interface: MI/FCI

Encoding: see Section 4.1.5

7.1.6. CDNI MI PathMetadata Payload Type

Purpose: The purpose of this Payload Type is to distinguish PathMetadata MI objects (and any associated capability advertisement)

Interface: MI/FCI

Encoding: see Section 4.1.6

7.1.7. CDNI MI SourceMetadata Payload Type

Purpose: The purpose of this Payload Type is to distinguish SourceMetadata MI objects (and any associated capability advertisement)

Interface: MI/FCI

Encoding: see Section 4.2.1

7.1.8. CDNI MI Source Payload Type

Purpose: The purpose of this Payload Type is to distinguish Source MI objects (and any associated capability advertisement)

Interface: MI/FCI

Encoding: see Section 4.2.1.1

7.1.9. CDNI MI LocationACL Payload Type

Purpose: The purpose of this Payload Type is to distinguish LocationACL MI objects (and any associated capability advertisement)

Interface: MI/FCI

Encoding: see Section 4.2.2

7.1.10. CDNI MI LocationRule Payload Type

Purpose: The purpose of this Payload Type is to distinguish LocationRule MI objects (and any associated capability advertisement)

Interface: MI/FCI

Encoding: see Section 4.2.2.1

7.1.11. CDNI MI Footprint Payload Type

Purpose: The purpose of this Payload Type is to distinguish Footprint MI objects (and any associated capability advertisement)

Interface: MI/FCI

Encoding: see Section 4.2.2.2

7.1.12. CDNI MI TimeWindowACL Payload Type

Purpose: The purpose of this Payload Type is to distinguish TimeWindowACL MI objects (and any associated capability advertisement)

Interface: MI/FCI

Encoding: see Section 4.2.3

7.1.13. CDNI MI TimeWindowRule Payload Type

Purpose: The purpose of this Payload Type is to distinguish TimeWindowRule MI objects (and any associated capability advertisement)

Interface: MI/FCI

Encoding: see Section 4.2.3.1

7.1.14. CDNI MI TimeWindow Payload Type

Purpose: The purpose of this Payload Type is to distinguish TimeWindow MI objects (and any associated capability advertisement)

Interface: MI/FCI

Encoding: see Section 4.2.3.2

7.1.15. CDNI MI ProtocolACL Payload Type

Purpose: The purpose of this Payload Type is to distinguish ProtocolACL MI objects (and any associated capability advertisement)

Interface: MI/FCI

Encoding: see Section 4.2.4

7.1.16. CDNI MI ProtocolRule Payload Type

Purpose: The purpose of this Payload Type is to distinguish ProtocolRule MI objects (and any associated capability advertisement)

Interface: MI/FCI

Encoding: see Section 4.2.4.1

7.1.17. CDNI MI DeliveryAuthorization Payload Type

Purpose: The purpose of this Payload Type is to distinguish DeliveryAuthorization MI objects (and any associated capability advertisement)

Interface: MI/FCI

Encoding: see Section 4.2.5

7.1.18. CDNI MI Cache Payload Type

Purpose: The purpose of this Payload Type is to distinguish Cache MI objects (and any associated capability advertisement)

Interface: MI/FCI

Encoding: see Section 4.2.6

7.1.19. CDNI MI Auth Payload Type

Purpose: The purpose of this Payload Type is to distinguish Auth MI objects (and any associated capability advertisement)

Interface: MI/FCI

Encoding: see Section 4.2.7

7.1.20. CDNI MI Grouping Payload Type

Purpose: The purpose of this Payload Type is to distinguish Grouping MI objects (and any associated capability advertisement)

Interface: MI/FCI

Encoding: see Section 4.2.8

7.2. "CDNI Metadata Footprint Types" Registry

IANA has created a new "CDNI Metadata Footprint Types" subregistry in the "Content Delivery Network Interconnection (CDNI) Parameters" registry. The "CDNI Metadata Footprint Types" namespace defines the valid Footprint object type values used by the Footprint object described in Section 4.2.2.2. Additions to the "CDNI Metadata Footprint Types" namespace conform to the Specification Required policy as defined in [RFC5226]. The Designated Expert will verify that new type definitions do not duplicate existing type definitions

and prevent gratuitous additions to the namespace. New registrations are required to provide a clear description of how to interpret new footprint types.

The following table defines the initial values for the "CDNI Metadata Footprint Types" registry:

Footprint Type	Description	Specification
ipv4cidr	IPv4 CIDR address block	RFC 8006
ipv6cidr	IPv6 CIDR address block	RFC 8006
asn	Autonomous System Number (ASN)	RFC 8006
countrycode	ISO 3166-1 alpha-2 code	RFC 8006

7.3. "CDNI Metadata Protocol Types" Registry

IANA has created a new "CDNI Metadata Protocol Types" subregistry in the "Content Delivery Network Interconnection (CDNI) Parameters" registry. The "CDNI Metadata Protocol Types" namespace defines the valid Protocol object values (Section 4.3.2) used by the SourceMetadata and ProtocolACL objects. Additions to the Protocol namespace conform to the Specification Required policy as defined in [RFC5226], where the specification defines the Protocol Type and the protocol to which it is associated. The Designated Expert will verify that new protocol definitions do not duplicate existing protocol definitions and prevent gratuitous additions to the namespace.

The following table defines the initial Protocol values corresponding to the HTTP and HTTPS protocols:

Protocol Type	Description	Type Specification	Protocol Specifications
http/1.1	Hypertext Transfer Protocol -- HTTP/1.1	RFC 8006	RFC 7230
https/1.1	HTTP/1.1 over TLS	RFC 8006	RFC 7230, RFC 2818

8. Security Considerations

8.1. Authentication and Integrity

A malicious metadata server, proxy server, or attacker impersonating an authentic uCDN CDNI Metadata interface without being detected could provide false metadata to a dCDN that either:

- Denies service for one or more pieces of content to one or more User Agents;
- Directs dCDNs to contact malicious origin servers instead of the actual origin servers, so that malware or slanderous alternate content may be substituted for legitimate content; or
- Removes delivery restrictions (e.g., LocationACL, TimeWindowACL, ProtocolACL, or Auth metadata), allowing access to content that would otherwise be denied and thus possibly violating license restrictions and incurring unwarranted delivery costs.

Unauthorized access to metadata could also enable a malicious metadata client to continuously issue metadata requests in order to overload a uCDN's metadata server or servers.

Unauthorized access to metadata could further result in leakage of private information. A malicious metadata client could request metadata in order to gain access to origin servers, as well as information pertaining to content restrictions.

An implementation of the CDNI Metadata interface MUST use mutual authentication and message authentication codes to prevent unauthorized access to, and undetected modification of, metadata (see Section 8.3).

8.2. Confidentiality and Privacy

Unauthorized viewing of metadata could result in leakage of private information. Content provider origin and policy information is conveyed through the CDNI Metadata interface. A third party could intercept metadata transactions in order to gain access to origin servers, as well as information pertaining to content restrictions and usage patterns.

Note: The distribution of metadata by a uCDN to dCDNs could introduce privacy concerns for some content providers, e.g., dCDNs accepting content requests for a content provider's content might be able to obtain additional information and usage patterns relating to the users of a content provider's services. Content providers with concerns about divulging information to dCDNs can instruct their uCDN partners not to use CDNI when delivering their content.

An implementation of the CDNI Metadata interface MUST use strong encryption to prevent unauthorized interception or monitoring of metadata (see Section 8.3).

8.3. Securing the CDNI Metadata Interface

An implementation of the CDNI Metadata interface MUST support TLS transport as per [RFC2818] and [RFC7230].

TLS MUST be used by the server side (uCDN) and the client side (dCDN) of the CDNI Metadata interface, including authentication of the remote end, unless alternate methods are used for ensuring the security of the information in the CDNI Metadata interface requests and responses (such as setting up an IPsec tunnel between the two CDNs or using a physically secured internal network between two CDNs that are owned by the same corporate entity).

The use of TLS for transport of the CDNI Metadata interface messages allows the dCDN and uCDN to authenticate each other.

Once the dCDN and uCDN have mutually authenticated each other, TLS allows:

- The dCDN and uCDN to authorize each other (to ensure that they are transmitting/receiving CDNI Metadata requests and responses from an authorized CDN);
- CDNI Metadata interface requests and responses to be transmitted with confidentiality; and
- The integrity of the CDNI Metadata interface requests and responses to be protected during the exchange.

When TLS is used, the general TLS usage guidance in [RFC7525] MUST be followed.

9. References

9.1. Normative References

[ISO3166-1] The International Organization for Standardization, "Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes", ISO 3166-1:2013, 2013.

[POSIX] Institute of Electrical and Electronics Engineers, "Information Technology Portable Operating System Interface (POSIX) Part 1: System Application Program Interface (API) [C Language]", IEEE P1003.1, 1990.

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

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

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <http://www.rfc-editor.org/info/rfc4291>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <http://www.rfc-editor.org/info/rfc5890>.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <http://www.rfc-editor.org/info/rfc5891>.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <http://www.rfc-editor.org/info/rfc5952>.

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

[RFC7230] Fielding, R., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

[RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <http://www.rfc-editor.org/info/rfc7493>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <http://www.rfc-editor.org/info/rfc7525>.

9.2. Informative References

[CDNI-URI-SIGNING] van Brandenburg, R., Leung, K., Sorber, P., and M. Miller, "URI Signing for CDN Interconnection (CDNI)", Work in Progress, draft-ietf-cdni-uri-signing-10, October 2016.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <http://www.rfc-editor.org/info/rfc2818>.

[RFC6793] Vohra, Q. and E. Chen, "BGP Support for Four-Octet Autonomous System (AS) Number Space", RFC 6793, DOI 10.17487/RFC6793, December 2012, <http://www.rfc-editor.org/info/rfc6793>.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <http://www.rfc-editor.org/info/rfc7234>.

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

[RFC7337] Leung, K., Ed., and Y. Lee, Ed., "Content Distribution Network Interconnection (CDNI) Requirements", RFC 7337, DOI 10.17487/RFC7337, August 2014, <http://www.rfc-editor.org/info/rfc7337>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <http://www.rfc-editor.org/info/rfc7540>.

[RFC7736] Ma, K., "Content Delivery Network Interconnection (CDNI) Media Type Registration", RFC 7736, DOI 10.17487/RFC7736, December 2015, <http://www.rfc-editor.org/info/rfc7736>.

[RFC7975] Niven-Jenkins, B., Ed., and R. van Brandenburg, Ed., "Request Routing Redirection Interface for Content Delivery Network (CDN) Interconnection", RFC 7975, DOI 10.17487/RFC7975, October 2016, <http://www.rfc-editor.org/info/rfc7975>.

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

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

Acknowledgments

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

Contributors

The authors would also like to thank Grant Watson and Kent Leung for their contributions to this document.

Authors' Addresses

Ben Niven-Jenkins
Nokia
3 Ely Road
Milton, Cambridge CB24 6DD
United Kingdom

Email: ben.niven-jenkins@nokia.com

Rob Murray
Nokia
3 Ely Road
Milton, Cambridge CB24 6DD
United Kingdom

Email: rob.murray@nokia.com

Matt Caulfield
Cisco Systems
1414 Massachusetts Avenue
Boxborough, MA 01719
United States of America

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

Kevin J. Ma
Ericsson
43 Nagog Park
Acton, MA 01720
United States of America

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