Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

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

October 19, 2015

CDN Interconnection Metadata
draft-ietf-cdni-metadata-12

Abstract

The Content Delivery Networks 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.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Copyright Notice

Table of Contents

1. Introduction 4
  1.1. Terminology 5
  1.2. Supported Metadata Capabilities 5
2. Design Principles 6
3. CDNI Metadata object model 7
  3.1. HostIndex, HostMatch, HostMetadata, PathMatch, PatternMatch and PathMetadata objects 8
  3.2. Generic CDNI Metadata Objects 11
  3.3. Metadata Inheritance and Override 14
4. CDNI Metadata objects 15
  4.1. Definitions of the CDNI structural metadata objects . . . 16
    4.1.1. HostIndex 16
    4.1.2. HostMatch 17
    4.1.3. HostMetadata 18
    4.1.4. PathMatch 19
    4.1.5. PatternMatch 20
    4.1.6. PathMetadata 21
    4.1.7. GenericMetadata 22
  4.2. Definitions of the initial set of CDNI Generic Metadata objects 24
    4.2.1. SourceMetadata 24
      4.2.1.1. Source 25
    4.2.2. LocationACL Metadata 26
      4.2.2.1. LocationRule 28
      4.2.2.2. Footprint 28
    4.2.3. TimeWindowACL 29

4.2.3.1. TimeWindowRule 30
4.2.3.2. TimeWindow 31
4.2.4. ProtocolACL Metadata 32
4.2.4.1. ProtocolRule 33
4.2.5. DeliveryAuthorization Metadata 34

4.2.6. Cache 34
4.2.7. Auth 35
4.2.8. Grouping 36
4.3. CDNI Metadata Simple Data Type Descriptions 37
4.3.1. Link 37
4.3.2. Protocol 38
4.3.3. Endpoint 38
4.3.4. Time 38
4.3.5. IPv4CIDR 39
4.3.6. IPv6CIDR 39
4.3.7. ASN 39
4.3.8. CountryCode 39
5. CDNI Metadata Capabilities 40
6. CDNI Metadata interface 40
6.1. Transport 41
6.2. Retrieval of CDNI Metadata resources 42
6.3. Bootstrapping 43
6.4. Encoding 43
6.5. Extensibility 44
6.6. Metadata Enforcement 44
6.7. Metadata Conflicts 45
6.8. Versioning 45
6.9. Media Types 46
6.10. Complete CDNI Metadata Example 47
7. IANA Considerations 50
7.1. CDNI Payload Types 50
7.1.1. CDNI MI HostIndex Payload Type 51
7.1.2. CDNI MI HostMatch Payload Type 51
7.1.3. CDNI MI HostMetadata Payload Type 52
7.1.4. CDNI MI PathMatch Payload Type 52
7.1.5. CDNI MI PatternMatch Payload Type 52
7.1.6. CDNI MI PathMetadata Payload Type 52
7.1.7. CDNI MI SourceMetadata Payload Type 52
7.1.8. CDNI MI Source Payload Type 53
7.1.9. CDNI MI LocationACL Payload Type 53
7.1.10. CDNI MI LocationRule Payload Type 53

7.1.11. CDNI MI Footprint Payload Type 53
7.1.12. CDNI MI TimeWindowACL Payload Type 53
7.1.13. CDNI MI TimeWindowRule Payload Type 54
7.1.14. CDNI MI TimeWindow Payload Type 54
7.1.15. CDNI MI ProtocolACL Payload Type 54
7.1.16. CDNI MI ProtocolRule Payload Type 54
7.1.17. CDNI MI DeliveryAuthorization Payload Type 54
7.1.18. CDNI MI Cache Payload Type 55
7.1.19. CDNI MI Auth Payload Type 55
7.1.20. CDNI MI Grouping Payload Type 55
7.2. CDNI Metadata Footprint Types Registry 55
7.3. CDNI Metadata Protocol Types Registry 56

7.4. CDNI Metadata Auth Types Registry 56
8. Security Considerations 57
8.1. Authentication 57
8.2. Confidentiality 57
8.3. Integrity 58
8.4. Privacy 58
8.5. Securing the CDNI Metadata interface 58
9. Acknowledgements 59
10. Contributing Authors 59
11. References 59
11.1. Normative References 59
11.2. Informative References 60
Authors' Addresses 61

1. Introduction

Content Delivery Networks 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 is discussed in [RFC7336] along with four other interfaces that may be used to compose a CDNI solution (CDNI Control interface, CDNI Request Routing Redirection interface, CDNI Footprint & Capabilities Advertisement interface and CDNI Logging interface). [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 an uCDN in accordance with the policies defined by the uCDN.

This document focuses on the CDNI metadata interface which enables a dCDN to obtain CDNI metadata from an uCDN so that the dCDN can properly process and respond to:

- o Redirection requests received over the CDNI Request Routing Redirection interface [I-D.ietf-cdni-redirection].

- o Content requests received directly from User Agents.

Specifically, this document specifies:

- o A data structure for mapping content requests and redirection requests to CDNI metadata objects (Section 3 and Section 4.1).

- o An initial set of CDNI Generic metadata objects (Section 4.2).

- o A 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:

- o Object - a collection of properties.

- o 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".

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:

o Acquisition Source: Metadata for allowing a dCDN to fetch content from a uCDN.

o Delivery Access Control: Metadata for restricting (or permitting) access to content based on any of the following factors:

  * Location

  * Time Window

  * Delivery Protocol

o Delivery Authorization: Metadata for authorizing dCDN user agent requests.

o 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, covering the initial capabilities above, is only able to support CDN interconnection for the delivery of content by a dCDN using HTTPv1.1 [RFC7230] and for a dCDN to be able to acquire content from a uCDN using either HTTPv1.1 or HTTPv1.1 over TLS [RFC2818].

Supporting CDN interconnection for the delivery of content using unencrypted HTTPv2.0 [RFC7540] (as well as for a dCDN to acquire content using unencrypted HTTPv2.0 or HTTPv2.0 over TLS) requires the registration of these protocol names in the CDNI Metadata Protocol Registry.

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

2. Design Principles

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

1. Cacheability of CDNI metadata objects.

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

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

4. Minimal duplication of CDNI metadata.

5. Leveraging of existing protocols.

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

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

Support for both HTTP and DNS redirection ensures that the CDNI metadata interface can be used for HTTP and DNS redirection and also

meets the same design principles for both HTTP and DNS based redirection schemes.

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

Leveraging existing protocols avoids reinventing common mechanisms such as data structure encoding (by leveraging I-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 an uCDN, authorize access to content, and deliver content from a dCDN. The object model relies on the assumption that these metadata properties may be aggregated based on the hostname of the content and subsequently on the resource path (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 behaviour when a dCDN surrogate or request router is processing User Agent requests for content at that Hostname or URI path. As a result of this structure, significant commonality may exist between the CDNI metadata properties specified for different Hostnames, different URI paths within a Hostname and different URI paths on different Hostnames. For example the definition of which User Agent IP addresses should be treated as being grouped together into a single network or geographic location is likely to be common for a number of different Hostnames. Another example is that although a uCDN is likely to have several different policies configured to express geo-blocking rules, it is likely that a single geo-blocking policy would be applied to multiple Hostnames delivered through the CDN.

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

be retrieved and stored by a dCDN once, even if it is referenced by the CDNI metadata of multiple Hostnames or of multiple URI paths.

Important Note: Any CDNI metadata object A that contains another CDNI metadata object B may, instead of including the second object B embedded within object A, include a Link object that contains a URI that can be dereferenced to retrieve the complete serialized representation of the second metadata object B. 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 and which CDNI metadata objects to embed and which to make available as 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 those objects.

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

Section 4 describes in detail the specific CDNI metadata objects and properties specified by this document which 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.

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

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

A HostIndex object (see Section 4.1.1) contains a list of HostMatch objects (see Section 4.1.2) that contain Hostnames (and/or IP addresses) for which content requests may 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, on receipt of a User Agent request for content, which other CDNI metadata objects it requires in order to deliver the requested content.

The HostIndex links Hostnames (and/or IP addresses) to HostMetadata objects (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 CDNI metadata within 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 properties for each host as well as PathMetadata objects (see Section 4.1.6), via PathMatch objects (see Section 4.1.4), which may override those properties for given URI paths within the host. The CDNI metadata contained in HostMetadata objects is applied to content requests for which there is not more specific metadata, i.e. for content requests that do not match any of the PathMatch objects contained by that HostMetadata object and its child PathMetadata objects.

HostMetadata can also contain PathMatch objects. PathMatch objects define patterns, contained inside PatternMatch objects (see Section 4.1.5), to match against the requested URI path, and contain PathMetadata objects which contain the GenericMetadata objects to be applied when a content request matches against the defined URI path pattern. PatternMatch objects contain the pattern strings and flags that describe the URI path that a PathMatch applies to.

PathMetadata objects override the CDNI metadata in the HostMetadata object or one or more parent PathMetadata objects with more specific CDNI metadata that applies to content requests matching the URI pattern defined in the PatternMatch object of that PathMatch object. A PathMetadata object may also contain PathMatch objects in order to recursively define more specific URI paths that require different (e.g., more specific) CDNI metadata to this one.

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

For example, if "example.com" is a content provider, a HostMatch object may include an entry for "example.com" with the URI of the associated HostMetadata object. The HostMetadata object for "example.com" describes the metadata properties which 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 CDNI metadata objects for those more specific URI paths. The PathMetadata object for "example.com/movies/*" describes CDNI metadata which apply to that URI path and could 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 which 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 (e.g., property semantics, enforcement options, etc.).

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

Although a CDN MUST NOT serve content to a User Agent if a "mandatory-to-enforce" property cannot be enforced, it may be "safe-to-redistribute" that metadata to another CDN without modification. For example, in the cascaded CDN case, a transit CDN (tCDN) may pass through "mandatory-to-enforce" metadata to a dCDN. For metadata which does not require customization or translation (i.e., metadata that is "safe-to-redistribute"), the data representation received off the wire MAY be stored and redistributed without being natively understood or supported by the transit CDN. However, for metadata

which requires translation, transparent redistribution of the uCDN metadata values might not be appropriate. Certain metadata may be safely, though possibly not optimally, redistributed unmodified. For example, source acquisition address may not be optimal if transparently redistributed, but might still work.

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

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

Table 2 describes the action to be taken by a transit CDN (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 either by transforming not StR 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 either by transforming not StR 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" (MtE) 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 incomprehensible.
False	False	False	Can serve.
False	True	False	Can serve but MUST NOT

			interpret/apply any metadata marked 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 may contain multiple PathMetadata objects (via PathMatch objects). Each PathMetadata object may 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 type LocationACL and TimeWindowACL, while a PathMetadata object which applies to "example.com/movies/*" defines an alternate GenericMetadata object of type TimeWindowACL, then:

- o 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

- o 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 a list 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 the inheritance tree and identify which specific GenericMetadata objects apply to a given User Agent content request.

Section 4.2 provides the definitions for a base set of core metadata objects which 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 sub-objects (i.e., GenericMetadata sub-objects contained within the GenericMetadata object that refers to that GenericMetadata sub-object). 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 may be defined in the future in separate documents.

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

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. See Section 6.4 for more details of 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" by this document 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, e.g., 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

Each of the sub-sections 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 a list 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 which applies to the request.

Property: hosts

Description: List 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: List 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.v1",
      "href": "http://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: String (Hostname or IP address) to match against the requested host. 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 when converted to lowercase in the content request MUST be identical to the value of the host property when converted to lowercase. IPv4 addresses MUST be 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].

Type: String

Mandatory-to-Specify: Yes.

Property: host-metadata

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

Type: HostMetadata

Mandatory-to-Specify: Yes.

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.v1",
    "href": "http://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: List of host related metadata.

Type: List of GenericMetadata objects

Mandatory-to-Specify: Yes.

Property: paths

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

Type: List of PathMatch objects

Mandatory-to-Specify: No.

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


```
{
  "metadata": [
    {
      <Properties of 1st embedded GenericMetadata object>
    },
    {
      <Properties of 2nd embedded GenericMetadata object>
    },

 ...

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

4.1.4. PathMatch

A PathMatch object contains a pattern within a PatternMatch object to match against a resource's URI path and contains a PathMetadata object to apply if the resource's URI path matches the pattern within the PatternMatch object.

Property: path-pattern

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

Type: PatternMatch

Mandatory-to-Specify: Yes.

Property: path-metadata

Description: CDNI metadata to apply when delivering content that matches 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.v1",
      "href": "http://metadata.ucdn.example/host1234/pathDCE"
  }
}
```


4.1.5. PatternMatch

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

Property: pattern

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

Type: String

Mandatory-to-Specify: Yes.

Property: case-sensitive

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

Type: Boolean

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

Property: ignore-query-string

Description: List of query parameters which should be ignored when searching for a pattern match. Matching against query parameters to ignore MUST be case-insensitive. If all query parameters should be ignored then the list MUST be empty.

Type: List of String

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

Example PatternMatch object that matches the case-sensitive URI path pattern "/movies/*". All query parameters will be ignored when matching URIs requested from surrogates by content clients against this path pattern:

```
{
  "pattern": "/movies/*",
```

```
  "case-sensitive": true,
  "ignore-query-string": []
}
```

Example PatternMatch object that matches the case-sensitive URI path pattern "/movies/*". The query parameter "sessionid" will be ignored when matching URIs requested from surrogates by content clients against this path pattern:

```
{
  "pattern": "/movies/*",
  "case-sensitive": true,
  "ignore-query-string": ["sessionid"]
}
```

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) and possibly child PathMatch objects.

Note that if DNS-based redirection is employed, then a dCDN will be unable to evaulate any metadata at the PathMetadata level or below against the content redirection request at request routing time because only the hostname of the content request is available at request routing time. dCDNs SHOULD still process any metadata at the PathMetadata level or below before responding to the redirection request in order to detect if any unsupported metadata is specifed. If any metadata is included and marked as "mandatory-to-enforce" which is not supported by the dCDN then the dCDN SHOULD NOT redirect the the content redirection request to itself in order to avoid receiving content requests that it is not able to satisfy/serve.

Property: metadata

Description: List of path related metadata.

Type: List of GenericMetadata objects

Mandatory-to-Specify: Yes.

Property: paths

Description: Path specific rules. First match applies.

Type: List of PathMatch objects

Mandatory-to-Specify: No.

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.

```
{
  "metadata": [
    {
      <Properties of 1st embedded GenericMetadata object>
    },
    {
      <Properties of 2nd embedded GenericMetadata object>
    },

 ...

    {
      <Properties of Nth embedded GenericMetadata 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 of the object contained in the generic-metadata-value property.

Mandatory-to-Specify: Yes.

Property: generic-metadata-value

Description: CDNI metadata object.

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

Mandatory-to-Specify: Yes.

Property: mandatory-to-enforce

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

Type: Boolean

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

Property: safe-to-redistribute

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

Type: Boolean

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

```
{
  "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 Generic Metadata objects

The objects defined below are intended to be used in the GenericMetadata object generic-metadata-value field as defined in Section 4.1.7 and their generic-metadata-type property MUST be set to the appropriate 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 an uCDN Surrogate or an Origin Server to obtain the content to be served. The sources are not necessarily the actual Origin Servers operated by the CSP but might be a set of Surrogates in the uCDN.

Property: sources

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

Type: List of Source objects (see Section 4.2.1.1)

Mandatory-to-Specify: No. Default is to use static configuration, out-of-band from the 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.v1"
  "generic-metadata-value":
    {
      "sources": [
        {
          "endpoints": [
            "a.service123.ucdn.example",
            "b.service123.ucdn.example"
            ],
          "protocol": "http1.1"
        },
        {
          "endpoints": ["origin.service123.example"],
          "protocol": "http1.1"
        }
      ]
    }
}
```


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 so a uCDN can specify a list of sources in preference order and for each source/preference rank a uCDN can specify a list 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 in preference order, for example a pool of servers behind a load balancer.

Type: List 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": "http1.1"
}
```

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 which does not include a locations property results in an action of allow, meaning that delivery can be performed regardless of the User Agent's location. The action from the first footprint to match against the User Agent's location is the action a CDN MUST take. If two or more footprints overlap, the first footprint that matches against the User Agent's location determines the action a CDN MUST take. If the locations property is included but is empty, or if none of the listed footprints matches the User Agent's location, then the result is an action of deny.

Although the LocationACL, TimeWindowACL (see Section 4.2.3), and ProtocolACL (see Section 4.2.4) are independent GenericMetadata objects, they may provide conflicting information to a dCDN, e.g., a content request which 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: Access control list which allows or denies (blocks) delivery based on the User Agent's location.

Type: List of LocationRule objects (see Section 4.2.2.1)

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

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

```
{
  "generic-metadata-type": "MI.LocationACL.v1"
  "generic-metadata-value":
    {
    }
}
```

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


```
{
  "generic-metadata-type": "MI.LocationACL.v1"
  "generic-metadata-value":
    {
      "locations": [
        {
          "action": "allow",
          "footprints": [
            {
              "footprint-type": "countrycode",
              "footprint-value": ["us"]
            }
          ]
        }
      ]
    }
}
```


4.2.2.1. LocationRule

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

Property: footprints

Description: List of footprints to which the rule applies.

Type: List of Footprint objects (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 may be applied to, e.g., an IPv4 address range or a geographic location.

Property: footprint-type

Description: Registered footprint type. 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: List of footprint values conforming to the specification associated with the registered footprint type. Footprint values may be simple strings (e.g., IPv4CIDR, IPv5CIDR, ASN, and CountryCode), however, other Footprint objects may be defined in the future, along with a more complex encoding (e.g., GPS coordinate tuples).

Type: List 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"]
}
```

4.2.3. TimeWindowACL

TimeWindowACL metadata defines time-based restrictions.

A TimeWindowACL which does not include a times property results in an action of allow, meaning that delivery can be performed regardless of the time of the User Agent's request. The action from the first window to match against the current time is the action a CDN MUST take. 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 matches the current time, then the result is an action of deny.

Although the LocationACL, TimeWindowACL, and ProtocolACL are independent GenericMetadata objects, they may provide conflicting information to a dCDN, e.g., a content request which 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: Access control list which allows or denies (blocks) delivery based on the time of a User Agent's request.

Type: List of TimeWindowRule objects (see Section 4.2.3.1)

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

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


```
{
  "generic-metadata-type": "MI.TimeWindowACL.v1"
  "generic-metadata-value":
    {
      "times": [
        {
          "action": "allow",
          "windows": [
            {
              "start": 946717200,
              "end": 946746000
            }
          ]
        }
      ]
    }
}
```


4.2.3.1. TimeWindowRule

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

Property: windows

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

Type: List of TimeWindow objects (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 09:00AM 01/01/2000 UTC and 17:00AM 01/01/2000 UTC:


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


4.2.3.2. TimeWindow

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

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

4.2.4. ProtocolACL Metadata

ProtocolACL metadata defines delivery protocol restrictions.

A ProtocolACL which does not include a protocol-acl property results in an action of allow, meaning that delivery can be performed regardless of the protocol of the User Agent's request. The action from the first protocol to match against the request protocol is the action a CDN MUST take. If two or more request protocols overlap, the first protocol that matches thre 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 protocol matches the request protocol, then the result is an action of deny.

Although the LocationACL, TimeWindowACL, and ProtocolACL are independent GenericMetadata objects, they may provide conflicting information to a dCDN, e.g., a content request which 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: Description: Access control list which allows or denies (blocks) delivery based on delivery protocol.

Type: List of ProtocolRule objects (see Section 4.2.4.1)

Mandatory-to-Specify: No. Default is 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.v1"
  "generic-metadata-value":
    {
      "protocol-acl": [
        {
          "action": "allow",
          "protocols": ["http1.1"]
        }
      ]
    }
}
```


4.2.4.1. ProtocolRule

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

Property: protocols

Description: List of protocols to which the rule applies.

Type: List of Protocols (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 ProtocolRule object) that includes the protocol HTTP/1.1:

```
{
  "action": "allow",
  "protocols": ["http1.1"]
}
```


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 of the authorization methods in the list is satisfied for that request.

Type: List 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.v1"
  "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.

Property: ignore-query-string

Description: Allows a Surrogate to ignore URI query string parameters when comparing the requested URI against the URIs in its cache for equivalence. Matching against query parameters to ignore MUST be case-insensitive. Each query parameter to ignore is specified in the list. If all query parameters

should be ignored, then the list MUST be specified and MUST be empty.

Type: List of String

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

Example Cache object that instructs the dCDN to ignore all query parameters:

```
   {
     "generic-metadata-type":
       "MI.Cache.v1"
     "generic-metadata-value":
```

```
    {
      "ignore-query-string": []
    }
  }
```

Example Cache object that instructs the dCDN to ignore the (case-insensitive) query parameters named "sessionid" and "random":

```
  {
    "generic-metadata-type":
      "MI.Cache.v1"
    "generic-metadata-value":
    {
      "ignore-query-string": ["sessionid", "random"]
    }
  }
```

4.2.7. Auth

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

Property: auth-type

Description: Registered Auth type (Section 7.4).

Type: String

Mandatory-to-Specify: Yes.

Property: auth-value

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

Type: GenericMetadata Object

Mandatory-to-Specify: Yes.

Example Auth object:

```
{
  "generic-metadata-type":
    "MI.Auth.v1"
  "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 large group of content to which a given asset belongs.

Property: ccid

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

Type: String

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

Example Grouping object that specifies a Content Collection Identifier for the content associated with the Grouping object's parent HostMetdata or PathMetadata:

```
{
  "generic-metadata-type":
    "MI.Grouping.v1"
  "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 may be used in place of any of the objects or properties 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 instead of another metadata 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 type of the object being referenced.

Type: String

Mandatory-to-Specify: No

Example Link object referencing a HostMetadata object:

```
{
  "type": "MI.HostMetadata.v1",
  "href": "http://metadata.ucdn.example/host1234"
}
```

4.3.2. Protocol

Protocol objects are used to specify registered protocols for content acquisition or delivery (see Section 7.3).

Type: String

Example:

"http1.1"

4.3.3. Endpoint

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

Note: All implementations MUST support IPv4 addresses encoded as specified by the 'IPv4address' rule in Section 3.2.2 of [RFC3986]. 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].

Type: String

Example Hostname:

"http://metadata.ucdn.example/host1234"

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 Unix epoch in the UTC timezone.

Type: Integer

Example Time representing 09:00AM 01/01/2000 UTC:

946717200

4.3.5. IPv4CIDR

An IPv4address 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 IPv4 CIDR:

"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 IPv6 CIDR:

"2001:db8::/32"

4.3.7. ASN

An Autonomous System Number encoded as a string consisting of the characters "as" (in lowercase) followed by the Autonomous System number.

Type: String

Example ASN:

"as64496"

4.3.8. CountryCode

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 uCDN to dCDN. For optional metadata, it may be useful for the uCDN to know if the dCDN supports the underlying functionality described by the metadata, prior to delegating any content requests to the dCDN. 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 which may be assigned optional values, it may be useful for the uCDN to know which values a dCDN supports, prior to delegating any content requests to that dCDN. If the optional value assigned to a given piece of content's metadata is not supported by the dCDN, any delegated requests for that content may fail, so again the uCDN is likely to want to avoid delegating those requests to that dCDN.

The CDNI Footprint and Capabilities Interface (FCI) [RFC7336] provides a means of advertising capabilities from dCDN to uCDN. Support for optional metadata and support for optional metadata values may be advertised using the FCI.

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 either:

o Dynamically as required by the dCDN to process received requests.

For example in response to a query from an uCDN over the CDNI Request Routing Redirection interface (RI) [I-D.ietf-cdni-redirection] or in response to receiving a request for content from a User Agent. Or;

o In advance of being required. For example in the case of pre-positioned CDNI metadata acquisition.

The CDNI metadata interface is built on the principles of 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 Section 3 and Section 4).

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

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

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

The corresponding HTTP Response returns the status of the operation in the HTTP Status Code and returns the current representation of the resource (if appropriate) in the Response Body. HTTP Responses from servers implementing the CDNI metadata interface that contain a response body SHOULD include an ETag to enable validation of cached versions of returned resources.

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

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

6.2. Retrieval of CDNI Metadata resources

In the general case a CDNI metadata server makes CDNI metadata objects available via a unique URIs and therefore in order to retrieve CDNI metadata, a CDNI metadata client first makes a HTTP GET request for the URI of the HostIndex which provides the CDNI metadata client with a list of Hostnames for which the uCDN may 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 a GET request for the URI specified in the href property of the Link object which points to the HostMetadata object itself.

In order to retrieve the most specific metadata for a particular request, the CDNI metadata client inspects the HostMetadata for references to more specific PathMetadata objects (by matching the URI path in the request against the path-patterns in the PathMatch). If any PathMetadata match the request (and are linked rather than embedded), the CDNI metadata client makes another GET request for the PathMetadata. Each PathMetadata object may also include references to yet more specific metadata. If this is the case, the CDNI metadata client continues requesting 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 a redirection request from an uCDN or a 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, for example because the uCDN is uncontactable 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 unless the dCDN has stale versions of all the required metadata and the stale-if-error Cache-Control extension [RFC5861] was included in all previous responses that are required but cannot currently be retrieved. The dCDN can continue to serve other content for which it can retrieve (or for which it has fresh responses cached) all the required metadata even if some non-applicable part of the metadata tree is missing.

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

When application level redirection (e.g., HTTP 302 redirects) is being used between CDNs, it is expected that the 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 HostIndex served by each uCDN may be unique.

In the case of DNS redirection there is not always sufficient information carried in the DNS request from User Agents to determine the 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) and therefore dCDNs may have to apply local policy when deciding which uCDN's metadata to apply.

6.3. Bootstrapping

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

If the URI for the HostIndex object is not manually configured in the dCDN then the HostIndex URI could be discovered. A mechanism allowing the dCDN to discover the URI of the HostIndex 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 represented in lowercase.

6.5. Extensibility

The set of GenericMetadata objects may 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 allows any metadata to be included in either the HostMetadata or PathMetadata lists.

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 which defines a new GenericMetadata type SHOULD:

1. Specify the CDNI Payload Type used to identify the new GenericMetadata type being specified.

2. Define the set of properties associated with the new type contained within the GenericMetadata object. 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.

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 example of a use case to which it applies including an example encoded in I-JSON.

Note: In the case of vendor specific extensions, identification within the type name defined for a GenericMetadata object, of the organization that defined the new GenericMetadata object decreases the possibility of GenericMetadata type collisions.

6.6. Metadata Enforcement

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

In the 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 which does not support the "mandatory-to-enforce" metadata associated with the content being requested. However, even if the uCDN has a priori knowledge of the metadata supported by the dCDN (e.g., via the CDNI capabilities interface or through out-of-band negotiation between CDN operators) metadata support may fluctuate or be inconsistent (e.g., due to mis-communication, mis-configuration, or temporary outage). Thus, the dCDN MUST always evaluate all metadata associated with redirection requests 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 may obsolete or conflict with existing GenericMetadata (e.g., a future revision of the CDNI metadata interface may redefine the Auth GenericMetadata object or a custom vendor extension may 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 (e.g., MI.Auth.v1) and all are marked as "mandatory-to-enforce", it may be ambiguous which metadata should be applied, especially if the functionality of the metadata overlap.

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

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

6.8. Versioning

The version of CDNI metadata objects is conveyed inside the CDNI Payload Type that is included in the HTTP Content-Type header. Upon responding to a request for an object, a CDNI metadata server MUST include a Content-Type header with the CDNI Payload Type containing the version number of the object. HTTP requests sent to a metadata server SHOULD include an Accept header with the CDNI Payload Type (which includes the version) 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 may decide to include both in the Accept header. The version of each object defined by this document is version 1. For example: "Content-Type: application/cdni; ptype=MI.HostIndex.v1".

GenericMetadata objects include a "type" property which specifies the CDNI Payload Type of the GenericMetadata value. This CDNI Payload Type should also include a version. Any document which defines a new GenericMetadata type MUST specify the version number which it describes. For example: "MI.Location.v1".

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 Paylod Type for the metadata objects (resources) that are specified in this document.

Data Object	CDNI Payload Type
HostIndex	MI.HostIndex.v1
HostMatch	MI.HostMatch.v1
HostMetadata	MI.HostMetadata.v1
PathMatch	MI.PathMatch.v1
PatternMatch	MI.PatternMatch.v1
PathMetadata	MI.PathMetadata.v1
GenericMetadata	MI.SourceMetadata.v1
Source	MI.Source.v1
LocationACL	MI.LocationACL.v1
LocationRule	MI.LocationRule.v1
Footprint	MI.Footprint.v1
TimeWindowACL	MI.TimeWindowACL.v1
TimeWindowRule	MI.TimeWindowRule.v1
TimeWindow	MI.TineWindow.v1
ProtocolACL	MI.ProtocolACL.v1
ProtocolRule	MI.ProtocolRule.v1
DeliveryAuthorization	MI.DeliveryAuthorization.v1

Cache	MI.Cache.v1
Auth	MI.Auth.v1
Grouping	MI.Grouping.v1

Table 4: CDNI Payload Types for CDNI Metadata objects

6.10. Complete CDNI Metadata Example

A dCDN may request the HostIndex and receive the following object with a CDNI payload type of "MI.HostIndex.v1":


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


If the incoming request has a Host header with "video.example.com" then the dCDN would fetch the next metadata object from "http://metadata.ucdn.example/host1234" expecting a CDNI payload type of "MI.HostMetadata.v1":


```
{
  "metadata": [
    {
      "generic-metadata-type":
```

```
          "MI.SourceMetadata.v1",
        "generic-metadata-value": {
          "sources": [
            {
              "endpoint": "acq1.ucdn.example",
              "protocol": "http1.1"
            },
            {
              "endpoint": "acq2.ucdn.example",
              "protocol": "http1.1"
            }
          ]
        }
      },
      {
```



```
        "generic-metadata-type":
          "MI.LocationACL.v1",
        "generic-metadata-value": {
          "locations": [
            {
              "footprints": [
                {
                  "footprint-type": "IPv4CIDR",
                  "footprint-value": "192.0.2.0/24"
                }
              ],
              "action": "deny"
            }
          ]
        }
      },
      {
        "generic-metadata-type":
          "MI.ProtocolACL.v1",
        "generic-metadata-value": {
          "protocol-acl": [
            {
              "protocols": [
                "http1.1"
              ],
              "action": "allow"
```

```
            }
          ]
        }
      }
    ],
    "paths": [
      {
        "path-pattern": {
          "pattern": "/video/trailers/*"
        },
        "path-metadata": {
          "type": "MI.PathMetadata.v1",
          "href": "http://metadata.ucdn.example/host1234/pathABC"
        }
      },
      {
        "path-pattern": {
          "pattern": "/video/movies/*"
        },
        "path-metadata": {
          "type": "MI.PathMetadata.v1",
          "href": "http://metadata.ucdn.example/host1234/pathDCE"
```



```
        }
      }
    ]
  }
```

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

```
{
  "metadata": [],
  "paths": [
    {
      "path-pattern": {
        "pattern": "/videos/movies/hd/*"
      },
      "path-metadata": {
        "type": "MI.PathMetadata.v1",
```

```
          "href":
            "http://metadata.ucdn.example/host1234/pathABC/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 "http://metadata.ucdn.example/host1234/movies/hd" with CDNI payload type "MI.PathMetadata.v1":


```
  {
    "metadata": [
      {
        "generic-metadata-type":
          "MI.TimeWindowACL.v1",
        "generic-metadata-value": {
          "times": [
            "windows": [
              {
                "start": "1213948800",
                "end": "1327393200"
              }
```

            ],
            "action": "allow"
          ]
        }
      }
    ]
  }

7. IANA Considerations

7.1. CDNI Payload Types

This document requests the registration of the following CDNI Payload Types under the IANA CDNI Payload Type registry:

Payload Type	Specification
MI.HostIndex.v1	RFCthis
MI.HostMatch.v1	RFCthis

MI.HostMetadata.v1	RFCthis
MI.PathMatch.v1	RFCthis
MI.PatternMatch.v1	RFCthis
MI.PathMetadata.v1	RFCthis
MI.SourceMetadata.v1	RFCthis
MI.Source.v1	RFCthis
MI.LocationACL.v1	RFCthis
MI.LocationRule.v1	RFCthis
MI.Footprint.v1	RFCthis
MI.TimeWindowACL.v1	RFCthis
MI.TimeWindowRule.v1	RFCthis
MI.TimeWindow.v1	RFCthis
MI.ProtocolACL.v1	RFCthis
MI.ProtocolRule.v1	RFCthis
MI.DeliveryAuthorization.v1	RFCthis
MI.Cache.v1	RFCthis
MI.Auth.v1	RFCthis
MI.Grouping.v1	RFCthis

[RFC Editor: Please replace RFCthis with the published RFC number for this document.]

7.1.1. CDNI MI HostIndex Payload Type

Purpose: The purpose of this payload type is to distinguish HostIndex MI objects (and any associated capabilitiy 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 capabilitiy 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 capabilitiy 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 capabilitiy 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 capabilitiy 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 capabilitiy 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 capabilitiy 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 capabilitiy 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 capabilitiy 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 capabilitiy 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 capabilitiy 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 capabilitiy 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 capabilitiy 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 capabilitiy 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 capabilitiy 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 capabilitiy 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 capabilitiy 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 capabilitiy 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 capabilitiy 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 capabilitiy advertisement)

Interface: MI/FCI

Encoding: see Section 4.2.8

7.2. CDNI Metadata Footprint Types Registry

The IANA is requested to create a new "CDNI Metadata Footprint Types" registry in the "Content Delivery Networks Interconnection (CDNI) Parameters" category. The "CDNI Metadata Footprint Types" namespace defines the valid Footprint object type values used by the Footprint object in Section 4.2.2.2. Additions to the Footprint type namespace conform to the "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.

The following table defines the initial Footprint Registry values:

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

[RFC Editor: Please replace RFCthis with the published RFC number for this document.]

7.3. CDNI Metadata Protocol Types Registry

The IANA is requested to create a new "CDNI Metadata Protocol Types" registry in the "Content Delivery Networks Interconnection (CDNI) Parameters" category. The "CDNI Metadata Protocol Types" namespace defines the valid Protocol object values in Section 4.3.2, used by the SourceMetadata and ProtocolACL objects. Additions to the Protocol namespace conform to the "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 Specification
http1.1	Hypertext Transfer Protocol -- HTTP/1.1	RFCthis	RFC7230
https1.1	HTTP/1.1 Over TLS	RFCthis	RFC2818

[RFC Editor: Please replace RFCthis with the published RFC number for this document.]

7.4. CDNI Metadata Auth Types Registry

The IANA is requested to create a new "CDNI Metadata Auth Types" registry in the "Content Delivery Networks Interconnection (CDNI) Parameters" category. The "CDNI Metadata Auth Type" namespace defines the valid Auth object types used by the Auth object in Section 4.2.7. Additions to the Auth Type 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.

The registry will initially be unpopulated:

Auth Type	Description	Specification

8. Security Considerations

8.1. Authentication

Unauthorized access to metadata could result in denial of service. A malicious metadata server, proxy server or an attacker performing a "man in the middle" attack could provide malicious metadata to a dCDN that either:

- o Denies service for one or more pieces of content to one or more User Agents; or
- o Directs dCDNs to contact malicious origin servers instead of the actual origin servers.

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

Unauthorized access to metadata could 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 SHOULD use mutual authentication to prevent unauthorized access to metadata.

8.2. Confidentiality

Unauthorized viewing of metadata could result in leakage of private information. A third party could intercept metadata transactions in order to gain access to origin servers, as well as information pertaining to content restrictions.

An implementation of the CDNI metadata interface SHOULD use strong encryption to prevent unauthorized interception of metadata.

8.3. Integrity

Unauthorized modification of metadata could result in denial of service. A malicious metadata server, proxy server or an attacker performing a "man in the middle" attack" could modify metadata destined to a dCDN in order to deny service for one or more pieces of content to one or more user agents. A malicious metadata server, proxy server or an attacker performing a "Man in the middle" attack could modify metadata so that dCDNs are directed to contact to malicious origin servers instead of the actual origin servers.

An implementation of the CDNI metadata interface SHOULD use strong encryption and mutual authentication to prevent unauthorized modification of metadata.

8.4. Privacy

Content provider origin and policy information is conveyed through the CDNI metadata interface. The distribution of this information to another CDN may introduce potential privacy concerns for some content providers, for example because dCDNs accepting content requests for a content provider's content may be able to obtain additional information & usage patterns relating to the users of a content provider's services. Content providers with such concerns can instruct their CDN partners not to use CDN interconnects when delivering that content provider's content.

8.5. Securing the CDNI Metadata interface

An implementation of the CDNI metadata interface MUST support TLS transport as per [RFC2818] and [RFC7230]. The use of TLS for transport of the CDNI metadata interface messages allows:

o The dCDN and uCDN to authenticate each other.

and, once they have mutually authenticated each other, it allows:

o The dCDN and uCDN to authorize each other (to ensure they are transmitting/receiving CDNI metadata requests & responses from an authorized CDN).

o CDNI metadata interface requests and responses to be transmitted with confidentiality.

o The integrity of the CDNI metadata interface requests and responses to be protected during the exchange.

In an environment where any such protection is required, TLS MUST be used (including authentication of the remote end) by the server-side (uCDN) and the client-side (dCDN) of the CDNI metadata interface unless alternate methods are used for ensuring the confidentiality 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).

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

9. Acknowledgements

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.

10. Contributing Authors

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

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

Email: gwatson@velocix.com

Kent Leung
Cisco Systems
3625 Cisco Way
San Jose, 95134
USA

Email: kleung@cisco.com

11. References

11.1. Normative References

[ISO3166-1]
"https://www.iso.org/obp/ui/#search".

[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>.

[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>.

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

[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>.

[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>.

[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>.

11.2. Informative References

[I-D.ietf-cdni-redirection] Niven-Jenkins, B. and R. Brandenburg, "Request Routing Redirection Interface for CDN Interconnection", draft-ietf-cdni-redirection-13 (work in progress), October 2015.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000,

<http://www.rfc-editor.org/info/rfc2818>.

[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>.

[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>.

[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>.

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

[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>.

Authors' Addresses

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

Email: ben@velocix.com

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

Email: rmurray@velocix.com

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

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

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

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