Network Working Group K. Ma
Internet-Draft Azuki Systems, Inc.
Intended status: Standards Track July 16, 2012
Expires: January 17, 2013

Content Distribution Network Interconnection (CDNI) Metadata Interface
draft-ma-cdni-metadata-03

Abstract

Content publishers (CPs) often use multiple Content Delivery Networks (CDNs) to deliver content to consumers. Though existing interactions between CPs and individual CDNs are beyond the scope of CDN interconnection (CDNI), it is important to understand the management capabilities and features available with existing non-interconnected multi-CDN deployments. Before migrating to CDNI, CPs must first assess the suitability of CDNI as a replacement for their existing non-interconnected multi-CDN deployments. CDN feature configuration and capability advertisement and enforcement is likely to occur through the CDNI metadata interface (MI). This document describes an approach to implementing the CDNI MI through the use of an extensible metadata model and a light-weight HTTP-based API.

Requirements Language

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

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

The use cases described in the CDNI use case document [I-D.ietf-cdni-use-cases] provide motivational use cases for CDN interconnection (CDNI). They describe reasons and situations where CDNI provides a benefit to CDN vendors as well as content service providers (CSPs). Additional use cases exist which describe how CDNs are used today, however, these use cases often involve specific features (e.g., customized content transformations, content security, client authentication and filtering, content acquisition optimization and redundancy, etc.) which are beyond the scope of CDNI. Though the features themselves are not relevant to CDNI, the ability to support those features or enforce policies related to those features in a generic and extensible manner should be considered when designing CDNI interfaces. The ability to support feature parity with existing deployment models (i.e., non-CDNI-based CDN federation) may help to remove barriers to CDNI adoption.

Though certain interfaces are out of scope of CDNI, e.g.:

o upstream CDN (uCDN) configuration by the CP

o uCDN content acquisition

o uCDN content delivery

o downstream CDN (dCDN) content acquisition

o end user (EU) content acquisition

o third party workflow management

o third party request routing

An awareness of these interfaces and an understanding of the restrictions which they may impose on CDNI request routing is useful for understanding the needs of the CDNI metadata interface (MI). As described in the "Dynamic CDNI Metadata Acquisition Example" section in the CDNI framework document [I-D.davie-cdni-framework], upon receiving a request routing interface (RRI) request, the MI MAY be used to retrieve metadata that is "considered" before responding to the RRI request. To that end, the MI MUST define a deterministic method for handling metadata processing. Though the definition and interpretation of any individual piece of metadata is beyond the scope of CDNI, a well-defined method for how to respond to a RRI request when any unknown metadata value is encountered MUST be supported.

This document describes a simple data model for representing CDNI metadata and a simple protocol for creating and retrieving CDNI metadata in an opaque manner. The term opaque, in this case, should be understood to mean: without understanding the underlying meaning or interpretation of the metadata being represented. The metadata model and retrieval protocol SHOULD be completely independent of the definition of individual metadata values. The metadata model and retrieval protocol MUST also define default behaviors for dealing with metadata processing errors. The document defines a list of metadata which are likely applicable to a broad range of CDNI deployments. The document also provides a separate list of metadata which are likely to be desirable to content publishers (CPs). This document is not intended to suggest that any additional interfaces or requirements are needed beyond those already specified in the CDNI requirements document [I-D.ietf-cdni-requirements], nor is this document intended to suggest that any out of scope interfaces or content publisher feature functionality should be brought into scope. The metadata examples provided are intended only to illustrate possible features that interconnected CDNs may wish to support and the extensibility of the metadata model to handle those situations.

1.1. Terminology

[Ed. insert terminology reference]

1.2. Abbreviations

- o CDN: Content Distribution Network
- o uCDN: Upstream Content Distribution Network
- o dCDN: Downstream Content Distribution Network
- o CDNI: Content Distribution Network Interconnection
- o CP: Content Publisher
- o CSP: Content Service Provider
- o EU: End User
- o NSP: Network Service Provider
- o RRI: Request Routing Interface
- o MI: Metadata Interface

o CI: Control Interface

2. CDNI Metadata Data Model

The simple data model is shown in Figure 1 below. It includes a top level Domain object which describes the site(s) to which metadata is associated. The term site, in this case, should be understood to mean a collection of related content assets accessed through a single portal or Web-site. The Domain is associated with zero or more opaque Metadata objects. Each Metadata object is associated with one or more Base Address objects. The Metadata objects are each associated with a URI extension, applicable to any of the associated Base Addresses. A combination of Base Address and URI prefix matching is used identify Metadata to allow for hierarchical associations between individual Metadata and sets of content items. Each Domain is also associated with one or more Agent objects. Agents represent entities which require access to metadata (e.g., CPs, uCDNs, dCDNs, or local operators). An Agent is associated with each Metadata entry allowing different Metadata values to be returned to different Agents.

```
         +----------+
         |          | 1
         |   Agent  +---------------+
         |          |               |
         +----+-----+               |
              | 1..*                |
              |                     |
              | 1                   | 0..*
         +----+-----+          +----+-----+          +----------+
         |          | 1   0..* |          | 1   1..* |          |
         |  Domain  +----------+ Metadata +----------+ BaseAddr |
         |          |          |          |          |          |
         +----+-----+          +----------+          +----------+
```

Figure 1: CDNI Metadata Data Model

Note: The data model described above provides the basic components required for distributing Metadata and implementing the CDNI MI. The specific semantics of individual pieces of metadata are abstracted to allow for opaque distribution of metadata. Not all of the information described need be distributed through the MI. Some information (e.g., Domains and Agents) may be necessary for the MI to function, but MAY be negotiated or implemented out-of-band. They could be configured either by the CDN as part of a non-CDNI process, or through the CDNI control interface (CI) bootstrapping process, or using the MI APIs described herein. The MI APIs may also be used by

CDNs, internally, to configure themselves. The complete data model and full set of APIs are provided as part of a holistic MI description.

The following sections describe an example implementation of the metadata scheme described above using a standard SQL database.

2.1. Domain Table

The Domain object contains basic information related to the site being described. The example shown contains a primary key index and a unique name for the site. An OPTIONAL site description (e.g., a textual description of the site and its content) and site provider (e.g., the name of the CP or CSP which owns the content) information is also included.

```
CREATE TABLE "domain" ("domain_id" serial primary key,
                       "name" character varying(255) NOT NULL,
                       "provider" character varying(255),
                       "description" character varying(4095));
CREATE UNIQUE INDEX index_domain ON domain (name);
```

The Domain is the central object for binding Metadata. The example Domain shown below highlights the descriptive nature of the Domain object:

```
   domain_id: 1
   name: acme
   provider: acme rocket-powered products, inc
   description: fine purveyors of high quality anvils, rubber bands,
                bird seed, and rocket-powered footwear.
```

2.2. Base Address Table

The Base Address object contains basic hostname and base URI information related to the site being accessed. The example shown requires a primary key index, a string containing the hostname and base URI, and a foreign key reference to the Metadata to which this Base Address is associated. A uniqueness constraint is imposed on baseaddr/metadata_id pairs to prevent duplicate Base Address entries for a given Metadata.

```
CREATE TABLE "baseaddr" ("baseaddr_id" serial primary key,
                         "baseaddr" character varying(255) NOT NULL,
                         "metadata_id" integer NOT NULL);
CREATE UNIQUE INDEX index_baseaddr ON baseaddr (baseaddr,
                                                metadata_id);
```

Base Address Table Definition

The Base Address objects allows multiple hostname and base URI pairs to be associated with each Metadata object denoting the list of Base Addresses through which content within the Domain may be accessed. There are many cases where different Base Addresses are used to access the same content, e.g.:

- internal vs. external addresses: content may be accessible via both internal 10-net IP addresses and their associated DNS addresses and base URIs, as well as publicly routable external IP addresses and their associated DNS addresses and base URIs, where all of the addresses point to the same content servers and the base URIs are mapped to the same base directories,

- service white-labeling: multiple CSPs may provide access to the same content through different branded services where each branded service has its own DNS address and/or base URI, but all of the services point to the same content, or

- analytics partitioning: redirects from other sites may use different DNS addresses and/or base URIs, so that they may be easily accounted for, while still pointing at the same content.

The example Base Addresses shown below represent two DNS addresses through which content may be accessed as well as an internal IP address which may be used for staging:

```
baseaddr_id: 1
baseaddr: wile.e.coyote.acme.com
metadata_id: 1

baseaddr_id: 2
baseaddr: road.runner.acme.com
metadata_id: 1

baseaddr_id: 3
baseaddr: 10.10.10.10/meemeep
metadata_id: 1
```

Note: The exact schema described above may result in heavy duplication of Base Addresses. It is presented as an example for its simplicity, however, it may be optimized by using other table joining implementation schemes.

2.2.1. Hierarchical Base Addresses

In order to support hierarchical Base Addresses, the wildcard '*.' SHOULD be allowed as the first part of DNS-type Base Addresses. The wildcard does not make sense at the beginning of IP Address-type Base Addresses. Though a wildcard at the end of IP Address-type Base Addresses would make more sense, support for IP Address-type Base Addresses is OPTIONAL. The wildcard signifies the applicability of the associated Metadata value to all Base Addresses which match the address suffix.

The following two Base Addresses condense the previous example by allowing all acme.com DNS addresses:

```
   baseaddr_id: 1
   baseaddr: *.acme.com
   metadata_id: 1

   baseaddr_id: 2
   baseaddr: 10.10.10.10/meemeep
   metadata_id: 1
```

Note: There is no explicit enforcement that Base Addresses associated with a given piece of Metadata not overlap, however, for performance reasons, Base Addresses associated with a given piece of Metadata SHOULD NOT be allowed to overlap.

2.3. Agent Table

The Agent object contains basic information for authenticating entities which require access to Metadata. The example shown contains a primary key index, a string containing the username, an OPTIONAL string containing the password (possibly hashed or encrypted), a boolean value for differentiating between full read/ write access (e.g., for uCDNs) and read only access (e.g., for dCDNs), and a foreign key reference to the Domain to which this Agent is associated. A uniqueness constraint is imposed on username/ domain_id pairs to prevent duplicate Agent entries for a given Domain.

```
CREATE TABLE "agent" ("agent_id" serial primary key,
                      "username" character varying(255) NOT NULL,
                      "password" character varying(255),
                      "read_only" boolean DEFAULT false NOT NULL,
                      "domain_id" integer NOT NULL);
CREATE UNIQUE INDEX index_agent ON agent (username, domain_id);
```

Agent Table Definition

Note: The password field is included to support the HTTP authentication described in the API sections, however, if alternate authentication schemes are used, the password may not be necessary.

The Agent objects manage Metadata access rights. The Agent functionality as described attempts to address two issues:

- o security concerns: where unauthorized injection or deletion of Metadata may alter the functionality of a content service and MUST be prevented, as described in the Security Considerations section, and

- o customization requirements: where retrieval of certain metadata may require different responses depending on the Agent who is accessing the Metadata (e.g., with multiple and/or cascaded dCDNs).

Note: Though both of the above issues could be addressed through means outside of the MI, or through a means common to all of the CDNI interfaces, the Agent serves the purpose of addressing these needs within the context of the MI, in lieu of a consensus alternative and as per the CDNI framework document [I-D.davie-cdni-framework].

The example Agents shown below represent a uCDN Agent with write privileges and two dCDN Agents with read-only permissions:

```
agent_id: 1
username: ucdn
password: xxx
read_only: false
domain_id: 1

agent_id: 2
username: dcdn1
password: yyy
read_only: true
domain_id: 1

agent_id: 3
username: dcdn2
password: zzz
read_only: true
domain_id: 1
```

2.4. Metadata Table

The Metadata object contains the actual individual pieces of metadata for the site being described. The example shown contains a primary key index, a string containing the URI(s) to which the metadata applies, a name/value pair of strings which represent the name and value of the Metadata, respectively, as well as a boolean value stating whether or not the given Metadata must be enforced. An OPTIONAL priority value is included for creating order lists of values for a given named Metadata. An OPTIONAL ttl value and timeout field are included to support metadata invalidation. The table also contains a foreign key reference to the Domain to which this Metadata is associated and a foreign key reference to the Agent to whom this Metadata is intended. A compound uniqueness constraint is also applied to each uri/name/priority/domain_id/agent_id tuple to prevent a given Metadata from being ambiguously applied multiple times to the same URI in a given Domain for a given Agent.

```
  CREATE TABLE "metadata" ("metadata_id" serial primary key,
                           "uri" character varying(4095) NOT NULL,
                           "name" character varying(511) NOT NULL,
                           "value" character varying(65535) NOT NULL,
                           "must_enforce" boolean DEFAULT true NOT NULL,
                           "priority" integer DEFAULT 0 NOT NULL,
                           "ttl" integer DEFAULT 0 NOT NULL,
                           "timeout" timestamp without time zone,
                           "domain_id" integer NOT NULL,
                           "agent_id" integer NOT NULL);
  CREATE UNIQUE INDEX index_metadata ON metadata (uri, name, priority,
                                                  domain_id, agent_id);
```

The name/value pair is represented as simple opaque strings. The MI does not require and understanding of the semantics or inherent meaning of Metadata names or values to distribute the Metadata. Though, each piece of Metadata MUST have a defined set of semantics in order to be enforced, distributing the Metadata and determining whether or not the Metadata is supported does not require any understanding of the Metadata semantics, but rather, only an ability to identify supported Metadata by their name is REQUIRED. Metadata names SHOULD be properly defined and registered, and any implied functionality SHOULD be agreed upon and documented. A base set of CDNI Metadata is provided in the Metadata Definitions Section.

The intent of the must_enforce boolean is to identify Metadata that MUST be enforced by all CDNs. If a CDN is unable to understand or is unable to comply with the Metadata, it MUST NOT deliver the content being requested. For dCDNs, the must_enforce flag defines how to respond to MI and RRI requests when unknown or unsupported Metadata

is encountered. If Metadata is marked as must_enforce, then the dCDN MUST NOT accept any RRI request if it is unable to enforce that piece of Metadata (e.g., the named Metadata is not supported, the Metadata value is invalid, or the Metadata value is not supported). If the MI request resulted from a "recursive" RRI request, then the dCDN MUST return an error to the uCDN. If the MI request resulted from an "iterative" RRI request, then the dCDN MUST respond with a 403 Forbidden status code to the EU and report the failure to the uCDN.

In the case of cascaded CDN deployments, though a given CDN may not be able to enforce a given piece of Metadata, other CDNs further down stream may be able to enforce that Metadata. When a Metadata rejection occurs, the CDN SHOULD still store the Metadata so that it can be provided to other dCDNs.

The OPTIONAL priority value is provided to allow configuration of ordered Metadata lists. When specifying multiple values for a given named Metadata, each value MUST be specified with a unique priority value. The explicit priority value enforces a deterministic ordering across MI implementations.

The OPTIONAL ttl value is provided to allow configuration of a Metadata TTLs. If the ttl is specified, it MUST be specified in seconds and the timeout field SHOULD be populated by the local MI processor and used internally, to prevent the need for clock synchronization between MI processors.

The association of each Metadata to an Agent allows different Agents to retrieve different Metadata values for a given URI in the given Domain. This is intended to allow CDNs to separate upstream Metadata from downstream Metadata (e.g., a uCDN content acquisition URL may point to a CP origin, however, the content acquisition URL that the dCDN retrieves from the uCDN may point at a surrogate in the uCDN; likewise the content acquisition URLs for different dCDNs may point at different surrogates in the uCDN). Though this information could be hidden within a CDN's implementation, the security aspects related to deterministically associating an authenticated Agent with the proper metadata should be considered as part of the MI. Explicitly representing this in the data model reduces ambiguity in Metadata retrieval.

2.4.1. Hierarchical Metadata

In order to support hierarchical metadata, '/*' SHOULD be allowed as the last part of the URI hierarchy, signifying the application of this Metadata value to all URIs which match this URI prefix. If multiple Metadata are defined with overlapping prefixes, the URI with the longest prefix match MUST be used. The uniqueness constraint on

the uri/name/priority/domain_id tuple allows for unambiguous resolution of Metadata priority.

Note: The wildcard is only supported at the end of the URI string to provide a well-defined ordering for URI prefixes (i.e., longest prefix matching). Use of generalized regular expression matching requires ordering rules to ensure deterministically coherent results across multiple MI implementations. It is assumed that the URI path extensions (beyond the base paths provided in the Base Address) for content will be the same across CDNs. Any CDN specific URL rewrites MUST only affect the Base Address portion of the URL as defined in the Base Address.

Note: It is often desirable to separate specific types of files which may live in the same directory (e.g., .m3u8 vs .ts). Wildcard support in the URI support for file extension differentiation, i.e., '/*[.extension]', is OPTIONAL.

Given the following four Metadata objects, the value of color is defined five times, for three different URIs, all within the same Domain, but for different Agents:

```
metadata_id: 1
uri: /*
name: color
value: blue
must_enforce: false
priority: 0
ttl: 0
domain_id: 1
agent_id: 2

metadata_id: 2
uri: /*
name: color
value: gold
must_enforce: false
priority: 0
ttl: 0
domain_id: 1
agent_id: 1

metadata_id: 3
uri: /*
name: color
value: blue
must_enforce: false
priority: 1
```

```
      ttl: 0
      domain_id: 1
      agent_id: 1

      metadata_id: 4
      uri: /grass/*
      name: color
      value: brown
      must_enforce: false
      priority: 0
      ttl: 0
      domain_id: 1
      agent_id: 2

      metadata_id: 5
      uri: /grass/on/the/other/side/*
      name: color
      value: green
      must_enforce: false
      priority: 0
      ttl: 0
      domain_id: 1
      agent_id: 2
```

The default value for the color metadata (signified by the all encompassing URI "/*") is blue for the dCDN Agent and gold for the uCDN Agent, though the default color may be blue for the uCDN as well (as signified by the lower priority alternate color value). Alternate colors are associated with requests from the dCDN Agent for URIs that begin with "/grass". By default "/grass" has a color of brown, except when requesting "/grass/on/the/other/side/" which is green.

3. CDNI Metadata Bootstrapping

It is assumed that a well-known hostname to which MI requests should be sent is configured through the CDNI bootstrap data. Bootstrap information is sent through the CDNI CI, as described in the CDNI requirements document [I-D.ietf-cdni-requirements]. The MI APIs described herein are intended to be serviced by the MI running on that host.

Domain and Agent configurations must exist prior to Metadata creation/retrieval. Domains and Agents MAY be created as a part of an off-line business negotiation process or as a part of the CDNI bootstrapping process. Domain and Agent API descriptions are included in Appendix A and Appendix B, respectively. When the Domain

and Agent APIs described are used, access to the APIs SHOULD be secured using SSL with client authentication as described in the Security Considerations section.

Two sets of Agent configurations are also REQUIRED:

- o Upstream Agent Configuration: Agent credentials for all external agents who require access to the local CDN MI, e.g. for dCDNs to retrieve Metadata or for uCDNs to trigger Metadata.

- o Downstream Agent Configuration: Agent credentials for the local CDN to use when accessing uCDN MIs for retrieving Metadata or triggering Metadata responses. Separate credentials may be required for each uCDN and Domain combination from which content redirections may originate.

4. CDNI Metadata Management

The Metadata creation, modification, retrieval and removal protocols are defined in the following sections. All use a simple HTTP-based approach. The protocol, in general, SHOULD be data format agnostic. The examples shown herein use an XML representation for MI requests/ responses, however, other well-defined representations (e.g., JSON) are also acceptable. The examples shown illustrate functionality required to support the data model described in Section 2, however, any protocol which allows for the forced retrieval, invalidation, and removal of Metadata could also be acceptable.

Metadata creation/update is distinguished from retrieval by the HTTP method. Metadata creation/update MUST use the POST method. Metadata retrieval MUST use the GET method. Metadata MUST be removed if the value field is empty (i.e., updating the value to be an empty string MUST force removal of the entire Metadata entry and all associated Base Address entries).

A trigger API is also specified to initiate retrieval of Metadata. The uCDN may issue a trigger to the dCDN to force (re)acquisition of Metadata by the dCDN. The trigger API MUST use the POST method.

In addition to being secured using SSL with client authentication as described in the Security Considerations section, the MI SHOULD also employ an additional Agent authentication mechanism to filter requests and results. In the examples shown below, HTTP basic authentication is used for Agent authentication, though other methods (e.g., HTTP digest authentication or URL hashing) could also be used.

4.1. Metadata API

The Metadata for a Domain is created/modified/retrieved using the "/CDNI/MI/metadata" API. The metadata API REQUIRES a single query string argument "domain" which specifies the name of the Domain to which the Metadata being created/modified/retrieved belongs. Three additional OPTIONAL arguments MAY also be provided when retrieving metadata: "name" which specifies the name of the Metadata field to create/modify/retrieve, "uri" which specifies the URI for which the Metadata must apply, and/or "agent" which specifies the agent(s) to which the Metadata is associated, as a comma separated list. The "agent" option MUST only be allowed for agents with full read/write permissions.

A simple XML representation of the information provided to the metadata creation/update API or returned from the metadata retrieval API is shown below:

```
   <metadatas>
     <metadata>
       <uri></uri>
       <name></name>
       <values>
         <set>
           <value></value>
           <priority></priority>
         </set>
         ...
       </values>
       <must_enforce></must_enforce>
       <ttl></ttl>
       <agent></agent>
       <baseaddrs>
         <baseaddr></baseaddr>
         ...
       </baseaddrs>
     </metadata>
     ...
   </metadatas>
```

Metadata retrieval for a Domain may be triggered using the "/CDNI/MI/trigger" API. The trigger API provides the information required to issue a metadata API retrieval request (i.e., the "domain", "name, and "uri" query string arguments). The metadata API REQUIRES a single query string argument "action" which specifies what type of action is being triggered.

The following actions MUST be supported:

o refresh: The dCDN MUST retrieve and update all Metadata specified
   in the trigger.

The following actions are considered OPTIONAL:

o preposition: The dCDN SHOULD retrieve and update all Metadata
   specified in the trigger.

A simple XML representation of the information provided to the
trigger API is shown below:

```
<triggers>
  <trigger>
    <host></host>
    <domain></domain>
    <name></name>
    <uri></uri>
  </trigger>
  ...
</triggers>
```

4.1.1. Metadata Creation

The following example creates three new Metadata "color" for the
"dcdn" Agent in the "acme" Domain, issued by the "ucdn" Agent to the
uCDN MI:

```
POST /CDNI/MI/metadata?domain=acme HTTP/1.1
Host: ucdn.mi.cdni.example.com
Accept: */*
Authorization: Basic dWNkbjp4eHg=
Content-Length: 1053
Content-Type: application/x-www-form-urlencoded

<metadatas>
  <metadata>
    <uri>/grass/*</uri>
    <name>color</name>
    <values>
      <set>
        <value>brown</value>
        <priority>0</priority>
      </set>
    </values>
    <must_enforce>false</must_enforce>
    <ttl></ttl>
    <agent>dcdn</agent>
    <baseaddrs>
```

```
            <baseaddr>*.acme.com</baseaddr>
          </baseaddrs>
        </metadata>
        <metadata>
          <uri>/grass/on/the/other/side/*</uri>
          <name>color</name>
          <values>
            <set>
              <value>green</value>
              <priority>0</priority>
            </set>
          </values>
          <must_enforce>true</must_enforce>
          <ttl></ttl>
          <agent>dcdn</agent>
          <baseaddrs>
            <baseaddr>*.acme.com</baseaddr>
          </baseaddrs>
        </metadata>
        <metadata>
          <uri>/glasses/*</uri>
          <name>color</name>
          <values>
            <set>
              <value>violet</value>
              <priority>0</priority>
            </set>
          </values>
          <must_enforce>false</must_enforce>
          <ttl></ttl>
          <agent>ucdn</agent>
          <baseaddrs>
            <baseaddr>*.acme.com</baseaddr>
          </baseaddrs>
        </metadata>
      </metadatas>
```

4.1.2. Metadata Update

The following example updates the "color" Metadata for the "/glasses/*" portion of the "acme" Domain and "dcdn" Agent, issued by the "ucdn" Agent to the uCDN MI:

```
      POST /CDNI/MI/metadata?domain=acme HTTP/1.1
      Host: ucdn.mi.cdni.example.com
      Accept: */*
      Authorization: Basic dWNkbjp4eHg=
      Content-Length: 361
      Content-Type: application/x-www-form-urlencoded

      <metadatas>
        <metadata>
          <uri>/glasses/*</uri>
          <name>color</name>
          <values>
            <set>
              <value>rose</value>
              <priority>0</priority>
            </set>
            <set>
              <value>violet</value>
              <priority>2</priority>
            </set>
          </values>
          <must_enforce>true</must_enforce>
          <ttl></ttl>
          <agent>ucdn</agent>
          <baseaddrs>
            <baseaddr>*.acme.com</baseaddr>
          </baseaddrs>
        </metadata>
      </metadatas>
```

4.1.3. Metadata Refresh Trigger

The following example triggers the refresh of all "color" Metadata for the "acme" Domain. The trigger is issued by the "ucdn" Agent to the dCDN MI and is intended to force the "dcdn" Agent to retrieve Metadata from the uCDN MI.

```
      POST /CDNI/MI/trigger?action=refresh HTTP/1.1
      Host: dcdn.mi.cdni.example.com
      Accept: */*
      Authorization: Basic dWNkbjp4eHg=
      Content-Length: 155
      Content-Type: application/x-www-form-urlencoded

      <triggers>
        <trigger>
          <host>ucdn.mi.cdni.example.com</host>
          <domain>acme</domain>
          <name>color</name>
          <uri></uri>
        </trigger>
      </triggers>
```

The following example triggers the refresh of all Metadata for the URI "/grass/on/this/side", in the "acme" Domain. The trigger is issued by the "ucdn" Agent to the dCDN MI and is intended to force the "dcdn" Agent to retrieve Metadata from the uCDN MI.

```
      POST /CDNI/MI/trigger?action=refresh HTTP/1.1
      Host: dcdn.mi.cdni.example.com
      Accept: */*
      Authorization: Basic dWNkbjp4eHg=
      Content-Length: 169
      Content-Type: application/x-www-form-urlencoded

      <triggers>
        <trigger>
          <host>ucdn.mi.cdni.example.com</host>
          <domain>acme</domain>
          <name></name>
          <uri>/grass/on/this/side</uri>
        </trigger>
      </triggers>
```

4.1.4. **Metadata Retrieval**

The following example retrieves all "color" Metadata for the "acme" Domain. The request was issued by the "dcdn" Agent to the uCDN MI, and the results are filtered for the "dcdn" Agent:

```
      GET /CDNI/MI/metadata?domain=acme&name=color HTTP/1.1
      Host: ucdn.mi.cdni.example.com
      Accept: */*
      Authorization: Basic ZGNkbjp5eXk=

      HTTP/1.1 200 OK
      Content-Length: 714
      Connection: close
      Content-Type: text/xml

      <metadatas>
        <metadata>
          <uri>/grass/*</uri>
          <name>color</name>
          <values>
            <set>
              <value>brown</value>
              <priority>0</priority>
            </set>
          </values>
          <must_enforce>false</must_enforce>
          <ttl></ttl>
          <agent>dcdn</agent>
          <baseaddrs>
            <baseaddr>*.acme.com</baseaddr>
          </baseaddrs>
        </metadata>
        <metadata>
          <uri>/grass/on/the/other/side/*</uri>
          <name>color</name>
          <values>
            <set>
              <value>green</value>
              <priority>0</priority>
            </set>
          </values>
          <must_enforce>true</must_enforce>
          <ttl></ttl>
          <agent>dcdn</agent>
          <baseaddrs>
            <baseaddr>*.acme.com</baseaddr>
          </baseaddrs>
        </metadata>
      </metadatas>
```

The following example retrieves the Metadata for the URI "/grass/on/this/side" in the "acme" Domain. The request was issued by and the results are filtered for the "dcdn" Agent:

```
      GET /CDNI/MI/metadata?domain=acme&uri=/grass/on/this/side HTTP/1.1
      Host: ucdn.mi.cdni.example.com
      Accept: */*
      Authorization: Basic ZGNkbjp5eXk=

      HTTP/1.1 200 OK
      Content-Length: 361
      Connection: close
      Content-Type: text/xml

      <metadatas>
        <metadata>
          <uri>/grass/*</uri>
          <name>color</name>
          <values>
            <set>
              <value>brown</value>
              <priority>0</priority>
            </set>
          </values>
          <must_enforce>false</must_enforce>
          <ttl></ttl>
          <agent>dcdn</agent>
          <baseaddrs>
            <baseaddr>*.acme.com</baseaddr>
          </baseaddrs>
        </metadata>
      </metadatas>
```

4.1.5. **Metadata Removal**

The following example removes the violet "color" Metadata value for the URI "/glasses/*" and the "ucdn" Agent in the "acme" Domain by setting the value to an empty string, issued by the "ucdn" Agent to the uCDN MI:

```
      POST /CDNI/MI/metadata?domain=acme HTTP/1.1
      Host: ucdn.mi.cdni.example.com
      Accept: */*
      Authorization: Basic dWNkbjp4eHg=
      Content-Length: 225
      Content-Type: application/x-www-form-urlencoded

      <metadatas>
        <metadata>
          <uri>/glasses/*</uri>
          <name>color</name>
          <values>
            <set>
              <value/>
              <priority>2</priority>
            </set>
          </values>
          <agent>ucdn</agent>
        </metadata>
      </metadatas>
```

4.1.6. **Metadata Errors**

For any update, retrieval, or trigger request with malformed XML, the MI SHOULD respond with a 400 Bad Request status code. Ancillary unknown tags MAY be ignored.

For any trigger requests with an unsupported action, the MI SHOULD respond with a 403 Forbidden status code.

For any update or retrieval request for a uri/name/domain_id tuple which does not exist, the MI SHOULD respond with a 404 Not Found status code.

For any request which lacks a valid Agent authorization, the MI MUST respond with a 401 Unauthorized status code. This includes Agents with valid credentials, but who are marked as read_only and have requested Metadata associated with an alternate Agent through the specification of an "agent" query string parameter.

For any request which results in Metadata with an expired TTL, and for which an update cannot be retrieved from an upstream MI, the MI MUST respond to with a 500 Internal Server status code.

4.1.7. **Metadata Prepositioning**

The metadata creation/modification/removal APIs discussed above SHOULD only be used by uCDNs to manage Metadata in the local CDN.

Though the metadata creation/modification/removal APIs could be used to preposition metadata in dCDNs, the trigger API allows the uCDN to force refresh of the dCDN Metadata without directly posting Metadata to the dCDN. This allows the dCDNs to manage retrieval of Metadata using lazy updates.

dCDNs SHOULD NOT modify metadata dictated by a uCDN. dCDNs SHOULD only be assigned Agents with read_only access and SHOULD NOT have access to uCDN Domain or Agent APIs (restricted through the use of different SSL client authentication certificates, as described in the Security Considerations section).

5. Metadata Definitions

This section defines a base set of Metadata which SHOULD be supported by all CDNI implementations.

5.1. Origin Server

Content which is not pre-positioned must be acquired by the CDN from an origin server. The origin server Metadata specifies the base URL to which the content request URI may be appended in order to acquire the content. The origin server Metadata is defined as having the name "origin_server", with valid values containing a comma separated list of base URLs, and the must_enforce flag set to false:

```
name: origin_server
value: <url>
must_enforce: false
```

In some cases, multiple non-load balanced origin servers may be available for content acquisition. The origin server Metadata SHOULD support an unprioritized comma separate list of base URL values.

Note: The origin list Metadata is not a must_enforce, since, if the content cannot be acquired, there is no threat of unauthorized content distribution. Other Metadata or content pre-positioning may negate the need for origin server Metadata.

5.2. Activation Time

Content may be pre-positioned in anticipation of demand, however, the content license may have restrictions on delivery timeframe. The activation time Metadata specifies the first time at which the content may be delivered. The activation time Metadata is defined as having the name "activation_time", with valid timestamp values that MUST conform to RFC3339 [RFC3339], and the must_enforce flag set to

true:

```
   name: activation_time
   value: <timestamp>
   must_enforce: true
```

If the activation time Metadata is set and the current time is less than the specified activation time, the CDN MUST respond to requests for that content with a 403 Forbidden status code (or equivalent for the given non-HTTP request protocol).

5.3. Deactivation Time

Content may be pre-positioned in anticipation of demand, however, the content license may have restrictions on delivery timeframe. The deactivation time Metadata specifies the last time at which the content may be delivered. The deactivation time Metadata is defined as having the name "deactivation_time", with valid timestamp values that MUST conform to RFC3339 [RFC3339], and the must_enforce flag set to true:

```
   name: deactivation_time
   value: <timestamp>
   must_enforce: true
```

If the deactivation time Metadata is set and the current time is greater than the specified activation time, the CDN MUST respond to requests for that content with a 403 Forbidden status code (or equivalent for the given non-HTTP request protocol).

5.4. Administrative Disable

It is sometimes necessary to temporarily disable the distribution of certain media (e.g., inappropriate content, irregular access patterns, etc.) within a set accessibility period (i.e., the activation/deactivation time range). The administrative disable Metadata instructs the CDN not to deliver the specified content under any circumstances. The administrative disable Metadata is defined as having the name "admin_disable", with two valid values "true" and "false", and the must_enforce flag set to true:

```
   name: admin_disable
   value: [true | false]
   must_enforce: true
```

If the administrative disable Metadata is set to "true", the CDN MUST respond to requests for that content with a 403 Forbidden status code (or equivalent for the given non-HTTP request protocol).

5.5. Delegation Depth

CSPs may wish to prevent cascading CDNs to enforce licensing restrictions. The delegation depth Metadata instructs the CDN to only delegate requests for the specified content if the delegation depth is greater than zero. If the depth is less than or equal to zero, a uCDN should not delegate requests for the specified content to any dCDNs under any circumstances. When distributing the delegation depth Metadata the uCDN MUST decrement the value of delegation depth by at least one if the current value is greater than zero. The uCDN MAY choose not to decrement the value if the value is already less than or equal to zero. The uCDN MAY decrement by more than one in order to get to zero. The delegation depth Metadata is defined as having the name "delegate_depth", with an integer value and the must_enforce flag set to true:

```
name: delegate_depth
value: <integer>
must_enforce: true
```

If the delegation depth Metadata is less than or equal to 0, the CDN MUST either service the content requests itself or respond to requests for that content with a 504 Server Busy status code (or equivalent for the given non-HTTP request protocol).

5.6. Footprint Filter

CSPs often purchase rights to content which are only valid when accessed from certain locations (e.g., within a given country or through a given access network). The footprint filter Metadata provides a list of valid source IP subnets from which content requests may be accepted. The footprint filter Metadata is defined as having the name "footprint", with valid values containing a comma separated list of IP subnet definitions, and the must_enforce flag set to true:

```
name: footprint
value: <ip_subnet> [, <ip_subnet>]...
must_enforce: true
```

If the footprint filter Metadata is set and the source address of a requesting client does not match any of the IP subnets listed, the CDN MUST respond to the content request with a 403 Forbidden status code (or equivalent for the given non-HTTP request protocol).

5.7. HTTP Header Filter

CSPs often desire the ability to filter requests based on the existence of specific HTTP header fields and values (e.g., User-Agent headers for device detection or custom headers inserted by client-side applications). The HTTP header filter Metadata provides a list of HTTP header names and values which MUST be verified. The HTTP header filter Metadata is defined as having the name "http_filter_headers", with valid values containing a comma separated list of HTTP header names and regular expression matching criteria definitions, and the must_enforce flag set to true:

```
name: http_filter_headers
value: <name>:<regex> [, <name>:<regex>]...
must_enforce: true
```

If the HTTP header filter Metadata is set and the HTTP headers of the content request do not match all of the filters specified, the CDN MUST respond to the content request with a 403 Forbidden status code (or equivalent for the given non-HTTP request protocol).

5.8. HTTP Header Logging

CSP client applications often include proprietary headers in their content requests (e.g., for user tracking or analaytics collection) which may be needed for business reasons (e.g., billing) or may be useful for debugging purposes. The HTTP header logging Metadata provides a list of HTTP header names whose values MUST be extracted and logged with the normal per-request information passed through the CDNI logging interface. The HTTP header logging Metadata is defined as having the name "http_logging_headers", with valid values containing a comma separated list of HTTP header names, and the must_enforce flag flag optionally set to true (depending on the application):

```
name: http_logging_headers
value: <name> [, <name>]...
must_enforce: [true | false]
```

If the HTTP header logging Metadata is set and the content request contains HTTP headers which match any of the header names listed, the CDN MUST extract all matching headers and add them to the per-request log message.

5.9. Protocol Filter

Though content is typically only accessible using specific a protocol (e.g., HTTP, RTMP, or RTSP), a CSP may wish to explicitly allow/

disallow access to certain content for a given protocol. The protocol filter Metadata provides a list of allowed protocols via which content may be delivered. The protocol filter Metadata is defined as having the name "protocol", with valid values containing a comma separate list of protocol strings, and the must_enforce flag set to true:

```
name: protocols
value: <protocol> [, <protocol>]...
must_enforce: true
```

If the protocol filter Metadata is set and the request protocol does not match any protocol in the list, the CDN MUST respond to the content request with a 403 Forbidden status code (or equivalent for the given non-HTTP request protocol).

5.10. SSL Required

CSPs which require delivery privacy may require dCDNs to support the same SSL configurations which were applied to the uCDN. The SSL required Metadata expresses the requirement to enforce SSL on content request connections and provides the necessary key and certificate information required for server authentication. The SSL required Metadata is defined as having the name "ssl_required", with valid values containing two URLs (comma separated) which point to the key and certificate, respectively, and the must_enforce flag set to true:

```
name: ssl_required
value: <key_url>,<cert_url>
must_enforce: true
```

If the SSL required Metadata is set and the request is not received over an SSL channel, the CDN MUST respond to the content request with a 403 Forbidden status code (or equivalent for the given non-HTTP request protocol).

Note: Retrieval of server key and certificate information SHOULD be performed in a secure manner. Retrieval could be implemented through the CDNI MI, however, this is not required.

5.11. SSL Client Authentication Required

CSPs which require client authentication may require dCDNs to support a SSL client authentication configuration which was applied to the uCDN. The SSL client authentication required Metadata expresses the requirement to enforce SSL client authentication on content requests and provides the necessary certificate authority (CA) information for authenticating clients. The SSL client authentication required

Metadata is defined as having the name "ssl_auth_required", with valid values containing a single URL which points to the CA certificate to be used in client verification, and the must_enforce flag set to true:

```
name: ssl_auth_required
value: <ca_url>
must_enforce: true
```

If the SSL client authentication required Metadata is set and the client certificate cannot be verified using the CA certificate, the CDN MUST respond with a handshake_failure alert.

5.12. URL Hash

TBD.

[Ed. Note: There are many proprietary URL hashing techniques in use today with varying timestamp formats, query string parameter names, hashing algorithm combinations, etc. A generic definition of URL hashing algorithm parameters, capable of supporting all algorithms would be best. An alternative of defining specific algorithms and assigning each and enumerated identifier would also work.]

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

There are a number of security concerns associated with the MI as Metadata may be used to influence CDNI request routing. Metadata may describe content acquisition parameters or content security restrictions. Altering Metadata or inhibiting Metadata discovery may impact content distribution. Some MI concerns include:

- intercepting and discarding Metadata requests to prevent content acquisition may be used as a denial of service attack,

- altering content acquisition Metadata to prevent content acquisition may be used as a denial of service attack, and

- spoofing content security Metadata to disable delivery restrictions may be used to circumvent rights management.

To combat these concerns, unauthorized access to the MI MUST be

prevented. The use of SSL with client authentication SHOULD be used for all MI APIs. Deployments in controlled environments where physical security and IP address white-listing is employed MAY choose not to use SSL. Different client authentication certificates SHOULD be used to protect access to Domain and Agent APIs, as well as uCDN access to the Metadata API, differently from dCDN access to the Metadata API. Deployments where uCDNs and dCDNs are mutually trusted entities (e.g., when uCDNs and dCDNs are controlled by the same corporate organization) MAY choose to use a single client authentication certificate.

8. Acknowledgements

The authors would like to thank Daniel Biagini, Susan He, Francois Le Faucheur, Kent Leung, Ben Niven-Jenkins, Gilles Bertrand, and Raj Nair for their helpful reviews and comments.

9. Appendix A: Domain API

Domain creation, modification, retrieval, and removal protocols are defined in the following sections. All use a simple HTTP-based approach. The protocol, in general, SHOULD be data format agnostic. The examples shown herein use an XML representation for MI requests/ responses, however, other well-defined representations (e.g., JSON) are also acceptable. The examples shown illustrate the functionality required to support the data model described in Section 2, however, any protocol which allows for the creation, modification, retrieval, and removal of Domains could also be acceptable.

Domain creation/update is distinguished from domain retrieval and removal by the HTTP method. Domain creation/update MUST use the POST method. Domain retrieval MUST use the GET method. Domain removal MUST use the DELETE method.

All Agents and Metadata MUST be associated with a Domain. A Domain is created/modified/retrieved/removed using the "/CDNI/MI/domain" API. The domain API REQUIRES a single query string argument "domain" which specifies the name of the Domain to be created/modified/ retrieved.

A simple XML representation of the information provided to the domain creation/update API or returned from the domain retrieval API is shown below:

```
      <domain>
        <provider></provider>
        <description></description>
      </domain>
```

9.1. Domain Creation

The following example creates a new Domain "acme":

```
      POST /CDNI/MI/domain?domain=acme HTTP/1.1
      Host: host.mi.cdni.example.com
      Accept: */*
      Content-Length: 81
      Content-Type: application/x-www-form-urlencoded

      <domain>
        <provider>acme</provider>
        <description>acme</description>
      </domain>
```

9.2. Domain Update

The following example updates the "acme" Domain:

```
    POST /CDNI/MI/domain?domain=acme HTTP/1.1
    Host: host.mi.cdni.example.com
    Accept: */*
    Content-Length: 209
    Content-Type: application/x-www-form-urlencoded

    <domain>
      <provider>acme rocket-powered products, inc</provider>
      <description>fine purveyors of high quality anvils, rubber bands,
                   bird seed, and rocket-powered footwear.</description>
    </domain>
```

9.3. Domain Retrieval

The following example retrieves the updated "acme" Domain
information:

```
     GET /CDNI/MI/domain?domain=acme HTTP/1.1
     Host: host.mi.cdni.example.com
     Accept: */*

     HTTP/1.1 200 OK
     Content-Length: 209
     Connection: close
     Content-Type: text/xml

     <domain>
       <provider>acme rocket-powered products, inc</provider>
       <description>fine purveyors of high quality anvils, rubber bands,
                    bird seed, and rocket powered footwear</description>
     </domain>
```

The MI MAY support bulk retrieval of Domains through the use of a comma separated list of Domain names in the domain query string parameter.

9.4. Domain Removal

The following example removes the "acme" Domain:

```
      DELETE /CDNI/MI/domain?domain=acme HTTP/1.1
      Host: host.mi.cdni.example.com
      Accept: */*
```

9.5. Domain Errors

Any update or retrieval request with malformed XML SHOULD respond with a 400 Bad Request status code. Ancillary unknown tags MAY be ignored.

Any update or retrieval request for a Domain which does not exist SHOULD respond with a 404 Not Found status code.

10. Appendix B: Agent API

Agent creation, modification, retrieval, and removal protocols are defined in the following sections. All use a simple HTTP-based approach. The protocol, in general, SHOULD be data format agnostic. The examples shown herein use an XML representation for MI requests/ responses, however, other well-defined representations (e.g., JSON) are also acceptable. The examples shown illustrate the functionality required to support the data model described in Section 2, however, any protocol which allows for the creation, modification, retrieval,

and removal of Agents could also be acceptable.

Agent creation/update is distinguished from Agent retrieval and removal by the HTTP method. Agent creation/update MUST use the POST method. Agent retrieval MUST use the GET method. Agent removal MUST use the DELETE method and specify the Agent name(s) in the query string.

All Metadata MUST be associated with an Agent. An Agent is created/ modified/retrieved/removed using the "/CDNI/MI/agent" API. The agent API REQUIRES a single query string argument "domain" which specifies the name of the Domain to which the Agent has access. In the case of DELETEs, the agent API also REQUIRES a query string argument "agent" which specifies the name(s) of the Agent(s) to remove, as a comma separated list.

A simple XML representation of the information provided to the agent creation/update API or returned from the agent retrieval API is shown below:

```
<agents>
  <agent>
    <username></username>
    <password></password>
    <read_only></read_only>
  </agent>
  ...
</agents>
```

10.1. Agent Creation

The following example creates three new Agents "ucdn", "dcdn1", and "dcdn2" for the "acme" Domain:

```
      POST /CDNI/MI/agent?domain=acme HTTP/1.1
      Host: host.mi.cdni.example.com
      Accept: */*
      Content-Length: 362
      Content-Type: application/x-www-form-urlencoded

      <agents>
        <agent>
          <username>ucdn</username>
          <password>xxx</password>
          <read_only>false</read_only>
        </agent>
        <agent>
          <username>dcdn1</username>
          <password>aaa</password>
          <read_only>false</read_only>
        </agent>
        <agent>
          <username>dcdn2</username>
          <password>bbb</password>
          <read_only>false</read_only>
        </agent>
      </agents>
```

10.2. Agent Update

The following example updates the "dcdn1" and "dcdn2" Agents in the "acme" Domain:

```
      POST /CDNI/MI/agent?domain=acme HTTP/1.1
      Host: host.mi.cdni.example.com
      Accept: */*
      Content-Length: 245
      Content-Type: application/x-www-form-urlencoded

      <agents>
        <agent>
          <username>dcdn1</username>
          <password>yyy</password>
          <read_only>true</read_only>
        </agent>
        <agent>
          <username>dcdn2</username>
          <password>zzz</password>
          <read_only>true</read_only>
        </agent>
      </agents>
```

10.3. Agent Retrieval

The following example retrieves the updated Agent information for the "acme" Domain:

```
GET /CDNI/MI/agent?domain=acme HTTP/1.1
Host: host.mi.cdni.example.com
Accept: */*

HTTP/1.1 200 OK
Content-Length: 360
Connection: close
Content-Type: text/xml

<agents>
  <agent>
    <username>ucdn</username>
    <password>xxx</password>
    <read_only>false</read_only>
  </agent>
  <agent>
    <username>dcdn1</username>
    <password>yyy</password>
    <read_only>true</read_only>
  </agent>
  <agent>
    <username>dcdn2</username>
    <password>zzz</password>
    <read_only>true</read_only>
  </agent>
</agents>
```

10.4. Agent Removal

The following example removes the "dcdn1" Agent from the "acme" Domain:

```
DELETE /CDNI/MI/agent?domain=acme&agent=dcdn1 HTTP/1.1
Host: host.mi.cdni.example.com
Accept: */*
```

10.5. Agent Errors

Any update or retrieval request with malformed XML SHOULD respond with a 400 Bad Request status code. Ancillary unknown tags MAY be ignored.

Any update or retrieval requests for an Agent which does not exist SHOULD respond with a 404 Not Found status code.

11. References

11.1. Normative References

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

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.

11.2. Informative References

[I-D.davie-cdni-framework]
Davie, B., Ed. and L. Peterson, Ed., "Framework for CDN Interconnection draft-davie-cdni-framework-01", October 2011.

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

[I-D.ietf-cdni-use-cases]
Bertrand, G., Stephan, E., Watson, G., Burbridge, T., Eardley, P., and K. Ma, "Use Cases for Content Delivery Network Interconnection draft-ietf-cdni-use-cases-04", March 2012.

Author's Address

Kevin J. Ma
Azuki Systems, Inc.
43 Nagog Park
Acton, MA 01720
USA

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