Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 1, 2012
R. Murray
B. Niven-Jenkins
Velocix (Alcatel-Lucent)
February 29, 2012

CDN Interconnect Triggers
draft-murray-cdni-triggers-00

Abstract

This document proposes a mechanism for a CDN to trigger activity in an interconnected CDN that is configured to deliver content on its behalf. The upstream CDN can use this mechanism to request that the downstream CDN pre-positions metadata or content, or that it re-validate or purge metadata or content. The upstream CDN can monitor the status of activity that it has triggered in the downstream CDN.

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 4
   1.1. Terminology 5
2. Model for CDNI Triggers 5
3. Collections of Trigger Status Resources 7
4. CDNI Trigger interface 8
   4.1. Creating Triggers 9
   4.2. Checking Status 10
      4.2.1. Polling Trigger Status Resource collections 10
      4.2.2. Polling Trigger Status Resources 10
   4.3. Deleting Triggers 10
   4.4. Expiry of Trigger Status Resources 11
5. Properties of Triggers 11
   5.1. Properties of Trigger Requests 11
      5.1.1. Content URLs 12
   5.2. Properties of Trigger Status Resources 12
   5.3. Properties of Trigger Collections 13
   5.4. Trigger Resource Simple Data Type Descriptions 14
      5.4.1. TriggerType 14
      5.4.2. TriggerStatus 14
      5.4.3. URLs 14
      5.4.4. UrlPatterns 14
      5.4.5. AbsoluteTime 15
6. JSON Encoding of Objects 15
   6.1. JSON Encoding of Embedded Types 16
      6.1.1. TriggerType 16
      6.1.2. TriggerStatus 16
      6.1.3. Metadata and Content References 16
      6.1.4. Pattern 16
      6.1.5. Relationship 17
   6.2. MIME Media Types 18
7. Examples 18
   7.1. Creating Triggers 18
      7.1.1. Preposition 19
      7.1.2. Invalidate 21
   7.2. Examining Trigger Status 22
      7.2.1. Collection of All Triggers 22
      7.2.2. Filtered Collections of Triggers 23

```
       7.2.3.  Trigger Status Resources . . . . . . . . . . . . . . . 25
       7.2.4.  Polling for Change . . . . . . . . . . . . . . . . . . 28
       7.2.5.  Cancelling or Removing a Trigger . . . . . . . . . . . 30
   8.  IANA Considerations  . . . . . . . . . . . . . . . . . . . . . 32
   9.  Security Considerations  . . . . . . . . . . . . . . . . . . . 32
   10. Acknowledgements . . . . . . . . . . . . . . . . . . . . . . . 32
   11. References . . . . . . . . . . . . . . . . . . . . . . . . . . 32
     11.1. Normative References . . . . . . . . . . . . . . . . . . . 32
     11.2. Informative References . . . . . . . . . . . . . . . . . . 32
   Authors' Addresses . . . . . . . . . . . . . . . . . . . . . . . . 33
```

1. Introduction

[I-D.ietf-cdni-problem-statement] introduces the Problem scope for CDN Interconnection (CDNI) and lists the four categories of interfaces that may be used to compose a CDNI solution (Control, Metadata, Request Routing, Logging).

[I-D.davie-cdni-framework] expands on the information provided in [I-D.ietf-cdni-problem-statement] and describes each of the interfaces and the relationships between them in more detail.

This draft concentrates on the "High" and "Medium" priority requirements for the CDNI Control Interface identified in section 4 of [I-D.ietf-cdni-requirements], reproduced here for convenience:

CNTL-1 [HIGH] The CDNI Control interface shall allow the Upstream CDN to request that the Downstream CDN (and, if cascaded CDNs are supported by the solution, that the potential cascaded Downstream CDNs) perform the following actions on an object or object set:

* Mark an object(s) and/or its CDNI metadata as "stale" and revalidate them before they are delivered again.
* Delete an object(s) and/or its CDNI metadata from the CDN surrogates and any storage. Only the object(s) and CDNI metadata that pertain to the requesting Upstream CDN are allowed to be purged.

CNTL-2 [HIGH] The CDNI Control interface shall allow the downstream CDN to report on the completion of these actions (by itself, and if cascaded CDNs are supported by the solution, by potential cascaded Downstream CDNs), in a manner appropriate for the action (e.g. synchronously or asynchronously).

CNTL-3 [HIGH] The CDNI Control interface shall support initiation and control by the Upstream CDN of pre-positioned CDNI metadata acquisition by the Downstream CDN.

CNTL-4 [MED] The CDNI Control interface should support initiation and control by the Upstream CDN of pre-positioned content acquisition by the Downstream CDN.

This document does not consider those parts of the control interface that relate to configuration, bootstrapping or authentication of CDN Interconnect interfaces.

o Section 2 outlines the model for the Trigger Interface at a high level.
o Section 3 describes collections of Trigger Resources.
o Section 4 defines the RESTful web service provided by dCDN.

- Section 5 lists properties of Trigger Requests and Status Resources.
- Section 6 defines a JSON encoding for Trigger Requests and Status Resources.
- Section 7 contains example messages.

1.1. Terminology

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

2. Model for CDNI Triggers

A trigger, sent from uCDN to dCDN, is a request for dCDN to do some work relating to data originating from uCDN.

The trigger may request action on either metadata or on content, it specifies one of the following actions:

- preposition - used to instruct dCDN to fetch metadata from uCDN, or content from any origin including uCDN.
- invalidate - used to instruct dCDN to revalidate specific metadata or content before re-using it.
- purge - used to instruct dCDN to delete specific metadata or content.

The CDNI trigger interface is a RESTful web service offered by dCDN. It allows creation and deletion of triggers, and tracking of the triggered activity. When dCDN accepts a trigger it creates a resource describing status of the triggered activity, a Trigger Status Resource. The uCDN may poll Trigger Status Resources to monitor progress.

Requests to invalidate and purge metadata or content apply to all variants of that data with a given URI.

The dCDN maintains a collection of Trigger Status Resources for each uCDN, each uCDN only has access to its own collection and the location of that collection is shared when CDN interconnection is established.

To trigger activity in dCDN, uCDN will POST to the collection of Trigger Status Resources. If dCDN accepts the trigger, it creates a new Trigger Status Resource and returns its location to uCDN. To monitor progress, uCDN may GET the Trigger Status Resource. To cancel a trigger, or remove a trigger from the collection once its activity has been completed, uCDN may DELETE the Trigger Status

Resource.

In addition to the collection of all Trigger Status Resources for uCDN, uCDN shall have access to filtered views of that collection. These filtered views are defined in Section 3 and include collections of active and completed triggers. These collections provide a mechanism for polling the status of multiple jobs.

Figure 1 is an example showing the basic message flow used by the uCDN to trigger activity in dCDN, and for uCDN to discover the status of that activity. Only successful triggering is shown. Examples of the messages are given in Section 7.

```
      uCDN                                                 dCDN
       |    (1) POST http://dcdn.example.com/triggers/uCDN     |
      [ ] --------------------------------------------------> [ ]--+
       |                                                      [ ]  | (2)
       |    (3) HTTP 201 Response                             [ ]<-+
      [ ] <-------------------------------------------------- [ ]
       |     Loc: http://dcdn.example.com/triggers/uCDN/123    |
       |                                                       |
       .                           .                           .
       .                           .                           .
       .                           .                           .
       |                                                       |
       |   (4) GET http://dcdn.example.com/triggers/uCDN/123   |
      [ ] --------------------------------------------------> [ ]
       |                                                      [ ]
       |   (5) HTTP 200 Trigger Status Resource               [ ]
      [ ] <-------------------------------------------------- [ ]
       |                                                       |
       |                                                       |
```

Figure 1: Basic CDNI Message Flow for Triggers

The steps in Figure 1 are:

1. uCDN triggers action in dCDN by posting to a collection of Trigger Status Resources, "http://dcdn.example.com/triggers/uCDN". The URL of this was given to uCDN when the trigger interface was established.
2. dCDN authenticates the request, validates the trigger and if it accepts the request, creates a new Trigger Status Resource. Timing of the activity in dCDN is under dCDN's control. Progress will be reported in the newly created Trigger Status Resource which uCDN may poll.
3. dCDN responds to uCDN with an HTTP 201 response status, and the location of the Trigger Status Resource.

4. uCDN may repeatedly poll the Trigger Status Resource in dCDN.
5. dCDN responds with the Trigger Status Resource, describing progress or results of the triggered activity.

The remainder of this document describes the messages, Trigger Status Resources, and collections of Trigger Status Resources in more detail.

3. Collections of Trigger Status Resources

As described in Section 2, Trigger Status Resources exist in dCDN to report the status of activity triggered by each uCDN.

A collection of Trigger Status Resources is a resource that contains a reference to each Trigger Status Resource in that collection.

To trigger activity in dCDN, uCDN creates a new Trigger Status Resource by posting to dCDN's collection of uCDN's Trigger Status Resources. The URL of each Trigger Status Resource is generated by the dCDN when it accepts the trigger, and returned to uCDN. This immediately enables uCDN to check the status of that trigger.

The dCDN must present a different set of Trigger Status Resources to each interconnected uCDN, only Trigger Status Resources belonging to a uCDN shall be visible to it. The dCDN may, for example, achieve this by offering different collection URLs to uCDNs, or by filtering the response based on the client uCDN.

The dCDN resource representing the collection of all uCDN's Trigger Status Resources is accessible to uCDN. This collection lists all uCDN triggers that have been accepted by dCDN, and have not yet been deleted by uCDN or expired and removed by dCDN.

In order to allow uCDN to check status of multiple jobs in a single request, dCDN shall also maintain collections representing filtered views of the collection of all Trigger Status Resources. The filtered collections are:

- Pending - Trigger Status Resources for triggers that have been accepted, but not yet acted upon.
- Active - Trigger Status Resources for triggered activity that is currently being processed in dCDN.
- Complete - Trigger Status Resources representing activity that completed successfully.
- Failed - Trigger Status Resources representing activity that failed.

4. CDNI Trigger interface

This section describes an interface to enable an upstream CDN to trigger defined activities in a downstream CDN. The interface is intended to be independent of the set of activities defined now, or that may be defined in future.

The CDNI Trigger interface is built on the principles of RESTful web services. Requests are made over HTTP, and the HTTP Method defines the operation the request would like to perform. 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 Triggers interface that contain a response body SHOULD include an ETag to enable validation of cached versions of returned resources.

Servers implementing the CDNI Trigger interface MUST support the HTTP GET, HEAD, POST and DELETE methods. The only representation specified in this document is JSON.

Trigger Requests are POSTed to a URI in dCDN. If the trigger is accepted by dCDN, it creates a Trigger Status Resource and returns its URI to dCDN in an HTTP 201 response. The triggered activity can then be monitored by uCDN using that resource and the collections described in Section 3.

The URI that Trigger Requests should be POSTed to needs to be either discovered by or configured in the upstream CDN. Performing a GET on that URI retrieves a collection of the URIs of all Trigger Status Resources. The URI of each Trigger Status Resource is also returned to uCDN when it is created. This means all Trigger Status Resources can be discovered, so CDNI Trigger servers are free to assign whatever structure they desire to the URIs for CDNI Trigger resources. CDNI Trigger clients MUST NOT make any assumptions regarding the structure of CDNI Trigger URIs or the mapping between CDNI Trigger objects and their associated URIs. Therefore any URIs present in the examples below are purely illustrative and are not intended to impose a definitive structure on CDNI Trigger interface implementations.

The CDNI Trigger interface builds on top of HTTP, so CDNI Trigger servers may make use of any HTTP feature when implementing the CDNI Trigger interface. For example, a CDNI Trigger server may make use of HTTP's caching mechanisms to indicate that the returned response/ representation has not been modified since it was last returned, reducing the processing needed to determine whether the status of triggered activity has changed.

This specification is neutral with regard to the transport below the HTTP layer. For example, is anticipated that decisions on use of HTTPS for other CDNI interfaces will be adopted for Triggers.

Discovery of the CDNI Triggers Interface is outside the scope of this document. It is anticipated that a common mechanism for discovery of all CDNI interfaces will be defined.

4.1. Creating Triggers

To create a new trigger, uCDN makes an HTTP POST to the unfiltered collection of its triggers. The request body of that POST is a Trigger Request.

dCDN validates and authenticates that request, if it is malformed or uCDN does not have sufficient access rights it MAY reject the request immediately. In this case, it SHALL respond with an appropriate 4xx HTTP error code and no resource shall be created on dCDN.

If the request is accepted, uCDN SHALL create a new Trigger Status Resource. The HTTP response to dCDN SHALL have status code 201 and the URI of the Trigger Status Resource in the Location header field. The HTTP response MAY include the content of the newly created Trigger Status Resource, this is recommended particularly in cases where the trigger has completed immediately.

Once a Trigger Status Resource has been created dCDN MUST NOT re-use its location, even after that resource has been removed through deletion or expiry.

The "request" property of the Trigger Status Resource SHALL contain the information posted in the body of the Trigger Request. Note that this need not be a byte-for-byte copy. For example, in the JSON representation the dCDN may re-serialise the information differently.

If the trigger is queued by dCDN for later action, the "status" property of the Trigger Status Resource SHALL be "pending". Once trigger processing has started the "status" SHALL be "active".

A trigger may result in no activity in dCDN if, for example, it is an invalidate or purge request for data the dCDN has not acquired, or a prepopulate request for data it has already acquired. In this case, the "status" of the Trigger Status Resource shall be "complete" and the Trigger Status Resource shall be added to the dCDN collection of Complete Triggers.

Once created, Trigger Status Resources may be deleted by uCDN but not modified. The dCDN MUST reject PUT and POST requests from uCDN to

Trigger Status Resources using HTTP status code 403.

4.2. Checking Status

The uCDN has two ways to check progress of activity it has triggered in dCDN, described in the following sections.

Because the triggers protocol is based on HTTP, Entity Tags may be used by the uCDN as cache validators, as defined in section 3.11 of [RFC2616], to cheaply check for change in status of a resource or collection of resources.

The dCDN should set the cache control headers for responses to GETs for Trigger Status Resources and Collections to indicate the frequency it would like uCDN to poll at.

4.2.1. Polling Trigger Status Resource collections

uCDN can fetch the collection of its Trigger Status Resources, or filtered views of that collection.

This makes it possible to poll status of all triggered activity in a single request. If dCDN moves a Trigger Status Resource from the Active to the Completed collection, uCDN may chose to fetch the result of that activity.

When polling in this way, uCDN may choose to use HTTP Entity Tags to monitor for change, rather than repeatedly fetching the whole collection.

4.2.2. Polling Trigger Status Resources

uCDN has a reference (URI provided by the dCDN) for each Trigger Status Resource it has created, it may fetch that resource at any time.

This may be used to retrieve progress information, and to fetch the result of triggered activity.

4.3. Deleting Triggers

The uCDN MAY delete Trigger Status Resources at any time, using the HTTP DELETE method.

Once deleted, the references to a Trigger Status Resource MUST be removed from all Trigger Status Resource collections.

If a "pending" Trigger Status Resource is deleted, dCDN SHOULD NOT

start processing of that activity. Deleting a "pending" trigger does not however guarantee that it is not started because, once it has triggered activity, uCDN cannot control the timing of that activity. Processing may, for example, start after the DELETE is sent by uCDN and before the DELETE is processed by dCDN.

If an "active" Trigger Status Resource is deleted, dCDN MAY stop processing the triggered activity. However, as with deletion of a "pending" trigger, dCDN does not guarantee this.

Deletion of a "complete" or "failed" Trigger Status Resource requires no processing in dCDN other than deletion of the resource.

4.4. Expiry of Trigger Status Resources

The dCDN MAY choose to automatically delete Trigger Status Resources some time after they become completed or failed. In this case, dCDN will remove the resource and respond to subsequent requests for it with HTTP status 404 or 410.

If dCDN performs this housekeeping, it MUST have reported the length of time after which completed Trigger Status Resources become stale via a property of the collection of all Trigger Status Resources. It is recommended that Trigger Status Resources are automatically deleted 24 hours after they become completed or failed.

To ensure it has access to the status of its completed and failed triggers, it is recommended that uCDN's polling interval is half the time after which records for completed activity will be considered stale.

5. Properties of Triggers

5.1. Properties of Trigger Requests

Properties of Trigger Requests are defined in the following subsections.

```
Property: type
   Description: This property defines the type of the trigger:
   Type: TriggerType
   Mandatory: Yes

Property: metadata.urls
   Description: The uCDN URL for the metadata the trigger applies
   to.
```

```
         Type: URLs
         Mandatory: No, but at least one of 'metadata.*' or 'content.*'
         MUST be present and non-empty.
      Property: content.urls
         Description: URLs of content data the trigger applies to, see
         Section 5.1.1.
         Type: URLs
         Mandatory: No, but at least one of 'metadata.*' or 'content.*'
         MUST be present and non-empty.
      Property: metadata.patterns
         Description: The metadata the trigger applies to.
         Type: UrlPatterns
         Mandatory: No, but at least one of 'metadata.*' or 'content.*'
         MUST be present and non-empty, and metadata.patterns MUST NOT
         be present if the TriggerType is Preposition.
      Property: content.patterns
         Description: The content data the trigger applies to.
         Type: UrlPatterns
         Mandatory: No, but at least one of 'metadata.*' or 'content.*'
         MUST be present and non-empty, and content.patterns MUST NOT be
         present if the TriggerType is Preposition.
```

5.1.1. Content URLs

Once interfaces for metadata and request routing interfaces have been agreed, it will be possible to define a way to make reference to content. That form of reference will be used in Trigger Requests.

Some possible options for content references are:

- o Canonical URL - a reference to the content that is shared by all Interconnected CDNs. A potential problem is that the URL visible to dCDN may have been modified by uCDN during request redirection.
- o Origin URL - the URL from which dCDN acquired the content. May be different in each CDN as each dCDN may use its uCDN as the origin.
- o Metadata URL - some portion of the metadata served to uCDN by dCDN will describe content, it would be possible to refer to content fetched as a result of that metadata description.

If the Content URL is modifed by uCDN, it is uCDN's responsibility to translate and pass-on Trigger Requests relating to that content using appropriately modified Content URLs.

5.2. Properties of Trigger Status Resources

```
      Property: trigger
         Description: The properties of trigger request that created
         this record.
```

```
         Type: TriggerRequest
         Mandatory: Yes

      Property: ctime
         Description: Time at which the request was received by dCDN.
         Time is local to dCDN, there is no requirement to synchronise
         clocks between interconnected CDNs.
         Type: AbsoluteTime
         Mandatory: Yes

      Property: mtime
         Description: Time at which the resource was last modified.
         Time is local to dCDN, there is no requirement to synchronise
         clocks between interconnected CDNs.
         Type: AbsoluteTime
         Mandatory: Yes
      Property: etime
         Description: Estimate of the time at which dCDN expects to
         complete the activity.  Time is local to dCDN, there is no
         requirement to synchronise clocks between interconnected CDNs.
         Type: AbsoluteTime
         Mandatory: No

      Property: status
         Description: Current status of the triggered activity.
         Type: TriggerStatus
         Mandatory: Yes

      Property: error
         Description: Error indication.
         Type: (To be decided - a set of standard error conditions needs
         to be defined.  The namespace for these errors codes should
         allow vendor-defined error codes for extension of the protocol.
         This may allow, for example, for the definition of more
         specific error codes when two CDNs supplied by the same vendor
         are interconnected.)
         Mandatory: No, and only allowed when "status" is "Failed".
```

5.3. Properties of Trigger Collections

```
      Property: links
         Description: References to Trigger Status Resources in the
         collection.
         Type: List of Relationships.
         Mandatory: Yes
      Property: staleresourcetime
```

Description: The length of time for which dCDN guarantees to keep a completed Trigger Status Resource. After this time, dCDN MAY delete the resource and all references to it from collections.
Type: Integer, time in seconds.
Mandatory: Yes, in the collection of all Trigger Status Resources if dCDN deletes stale entries. If the property is present in the filtered collections, it MUST have the same value as in the collection of all Trigger Status Resources.

5.4. Trigger Resource Simple Data Type Descriptions

This section describes the simpler data types that are used for properties of Trigger Status resources.

5.4.1. TriggerType

This type defines the type of action being triggered, permitted actions are:

- o Preposition - a request for dCDN to acquire metadata or content.
- o Invalidate - a request for dCDN to invalidate metadata or content. After servicing this request the dCDN will not use the specified data without first re-validating it using, for example, an "If-None-Match" HTTP request. The dCDN need not erase the associated data.
- o Purge - a request for dCDN to erase metadata or content. After servicing the request, the specified data must not be held on dCDN.

5.4.2. TriggerStatus

This type describes the current status of a Trigger, possible values are:

- o Pending - the trigger has not yet been acted upon.
- o Active - the trigger is currently being acted upon.
- o Complete - the triggered activity completed successfully.
- o Failed - the triggered activity could not be completed.

5.4.3. URLs

This type describes a set of references to metadata or content, it is simply a list of absolute URLs.

5.4.4. UrlPatterns

This type describes a reference to a set of metadata or content. It is a set of 'pattern.string' properties each of which has an optional

'pattern.flags'.

The intention is to align this with the pattern matching capabilities of the CDNI metadata interface, once defined. This current definition is based on that in [I-D.jenkins-cdni-metadata].

   Property: pattern.string
      Description: String to match against the URL of metadata or content, i.e. a [RFC3986] path-absolute.
      Type: Pattern
      Mandatory: Yes
   Property: pattern.flags
      Description: Flags to control the pattern match.
      Type: PatternFlags
      Mandatory: No (default Case-sensitive infix matching)

5.4.5. AbsoluteTime

Times are expressed in seconds since the UNIX epoch.

6. JSON Encoding of Objects

This encoding is based on that described in [I-D.jenkins-cdni-metadata], but has been reproduced here while metadata work is in progress. Once that work is complete, the authors would look to align with the structure of the metadata draft and make reference to common definitions as appropriate.

The "base" encoding for a CDNI Trigger object is a JSON object containing a dictionary of (key,value) pairs where the keys are the property names and the values are the associated property values.

The keys of the dictionary are the names of the properties associated with the object and are therefore dependent on the specific object being encoded (i.e. dependent on the MIME Media Type of the returned resource). Likewise, the values associated with each key are dependent on the specific object being encoded (i.e. dependent on the MIME Media Type of the returned resource).

Dictionary keys in JSON are case sensitive and therefore any dictionary key defined by this document (for example the names of CDNI Triggers object properties) MUST always be represented in lowercase.

In addition to the properties of the object, the following additional keys may be present.

```
      Key: base
         Description: Provides a prefix for any relative URLs in the
         object.  This is similar to the XML base tag [XML-BASE].  If
         absent, all URLs in the remainder of the document must be
         absolute URLs.
         Type: URI
         Mandatory: No

      Key: links
         Description: The relationships of this object to other
         addressable objects.
         Type: List of Relationships.
         Mandatory: Yes
```

6.1. JSON Encoding of Embedded Types

6.1.1. TriggerType

```
      Key: type
         Description: One of "preposition", "invalidate" or "purge".
         Type: string
         Mandatory: Yes
```

6.1.2. TriggerStatus

```
      Key: status
         Description: One of "pending", "active", "failed", "complete"
         Type: string
         Mandatory: Yes
```

6.1.3. Metadata and Content References

```
      Keys: metadata.urls, content.urls
         Description: A list of URLs of the addressable objects being
         referenced.
         Type: URLs
         Mandatory: No
      Keys: metadata.patterns, content.patterns
         Description: A list of patterns to match against URLs of
         objects being referenced.
         Type: list of Patterns
         Mandatory: No
```

6.1.4. Pattern

JSON: A dictionary with two keys, "pattern.string" and "pattern.flags":

```
      Key: pattern.string
         Description: The string to match URLs against.
         Type: string
         Mandatory: Yes
      Key: pattern.flags
         Description: A number calculated by adding together the values
         associated with each flag that is set.

         +  1 - Case-insensitive
         +  2 - Prefix
         +  4 - Suffix
         Type: integer
         Mandatory: Yes
```

Example of case-insensitive prefix match against "http://www.example.com/trailers/":

```
   {
       "pattern.string": "http://www.example.com/trailers",
       "pattern.flags": 3
   }
```

6.1.5. Relationship

JSON: A dictionary with the following keys:

- o href - The URI of the of the addressable object being referenced.
- o rel - The Relationship between the referring object and the object it is referencing.
- o type - The MIME Media Type of the referenced object. See Section 6.2 for the MIME Media Types of objects specified in this document.
- o title - An optional title for the Relationship/link.

Note: The above structure follows the pattern of atom:link in [RFC4287].

Example Relationship to a CDNI Trigger Resource within a CDNI Trigger Collection:

```
   {
     "href": "http://triggers.cdni.example.com/trigger/12345",
     "rel": "Trigger",
     "type": "application/vnd.cdni.control.trigger.status+json"
   }
```

The format of relationship values is expected to align with other CDNI interfaces. For example, rather than use simple names (like "Trigger" in this case), there may be a namespace that allows well-known and proprietary values to co-exist.

6.2. MIME Media Types

Table 1 lists the MIME Media Type for each trigger object (resource) that is retrievable through the CDNI Trigger interface.

Note: A prefix of "vnd.cdni" is used, however it is expected that a more appropriate prefix will be used if the CDNI WG accepts this document.

Data Object	MIME Media Type
TriggerStatus	application/ vnd.cdni.control.trigger.status+json
TriggerCollection	application/ vnd.cdni.control.trigger.collection+json

Table 1: MIME Media Types for CDNI Trigger resources

7. Examples

The following sections provide examples of different CDNI Trigger objects encoded as JSON.

No authentication is shown in the following illustrative examples, it is anticipated that authentication mechanisms will be aligned with other CDNI Interfaces as and when those mechanisms are defined.

Discovery of the triggers interface is out of scope of this document. In an implementation, all URLs are under control of dCDN and the uCDN must not attempt to ascribe any meaning to individual elements of the path. In examples in this section, the following URLs are used as the location of the collections of triggers:

o Collection of all Triggers belonging to one uCDN:
      http://dcdn.example.com/triggers
o Filtered collections:
      Pending: http://dcdn.example.com/triggers/pending
      Active: http://dcdn.example.com/triggers/active
      Complete: http://dcdn.example.com/triggers/complete
      Failed: http://dcdn.example.com/triggers/failed

7.1. Creating Triggers

Examples of uCDN triggering activity in dCDN:

7.1.1. Preposition

An example of a preposition request, a POST to the "AllTriggers" collection.

Note that a preposition request must not include any "metadata.patterns" or "content.patterns":

REQUEST:

```
POST /triggers HTTP/1.1
User-Agent: example-user-agent/0.1
Host: dcdn.example.com
Accept: */*
Content-Type: application/vnd.cdni.control.trigger.request+json
Content-Length: 277

{
  "type": "preposition",

  "metadata.urls" : [ "http://metadata.example.com/a/b/c" ],
  "content.urls" : [
      "http://www.example.com/a/b/c/1",
      "http://www.example.com/a/b/c/2",
      "http://www.example.com/a/b/c/3",
      "http://www.example.com/a/b/c/4"
    ]
}
```

RESPONSE:

```
HTTP/1.1 201 Created
Date: Wed, 29 Feb 2012 19:17:47 GMT
Content-Length: 472
Content-Type: application/vnd.cdni.control.trigger.status+json
Location: http://dcdn.example.com/triggers/0
Server: example-server/0.1

{
    "ctime": 1330543067,
    "etime": 1330543073,
    "mtime": 1330543067,
    "status": "pending",
    "trigger": {
        "content.urls": [
            "http://www.example.com/a/b/c/1",
            "http://www.example.com/a/b/c/2",
            "http://www.example.com/a/b/c/3",
            "http://www.example.com/a/b/c/4"
        ],
        "metadata.urls": [
            "http://metadata.example.com/a/b/c"
        ],
        "type": "preposition"
    }
}
```

7.1.2. Invalidate

An example of an invalidate request, another POST to the "AllTriggers" collection. This instructs dCDN to re-validate the content at "http://www.example.com/a/index.html", as well as any metadata and content whose URLs are prefixed by "http://metadata.example.com/a/b/" and "http://www.example.com/a/b/" respectively, using case-insensitive matching.

REQUEST:

```
  POST /triggers HTTP/1.1
  User-Agent: example-user-agent/0.1
  Host: dcdn.example.com
  Accept: */*
  Content-Type: application/vnd.cdni.control.trigger.request+json
  Content-Length: 337

  {
    "type": "invalidate",

    "metadata.patterns" : [
        { "pattern.string" : "http://metadata.example.com/a/b/",
          "pattern.flags" : 3 }
      ],

    "content.urls" : [ "http://www.example.com/a/index.html" ],
    "content.patterns" : [
        { "pattern.string" : "http://www.example.com/a/b/",
          "pattern.flags" : 3 }
      ]
  }
```

RESPONSE:

```
  HTTP/1.1 201 Created
  Date: Wed, 29 Feb 2012 19:17:47 GMT
  Content-Length: 596
  Content-Type: application/vnd.cdni.control.trigger.status+json
  Location: http://dcdn.example.com/triggers/1
  Server: example-server/0.1

  {
      "ctime": 1330543067,
      "etime": 1330543073,
      "mtime": 1330543067,
      "status": "pending",
      "trigger": {
          "content.patterns": [
```

```
                {
                    "pattern.flags": 3,
                    "pattern.string": "http://www.example.com/a/b/"
                }
            ],
            "content.urls": [
                "http://www.example.com/a/index.html"
            ],
            "metadata.patterns": [
                {
                    "pattern.flags": 3,
                    "pattern.string": "http://metadata.example.com/a/b/"
                }
            ],
            "type": "invalidate"
        }
    }
```

7.2. Examining Trigger Status

Once triggers have been created, uCDN can check their status as shown in these examples.

7.2.1. Collection of All Triggers

The uCDN can fetch the set of all the triggers it has created and which have not yet been deleted or removed as expired. After creation of the "preposition" and "invalidate" triggers shown above, this collection might look as follows:

```
REQUEST:

  GET /triggers HTTP/1.1
  User-Agent: example-user-agent/0.1
  Host: dcdn.example.com
  Accept: */*


RESPONSE:

  HTTP/1.1 200 OK
  Content-Length: 418
  Expires: Wed, 29 Feb 2012 19:18:47 GMT
  Server: example-server/0.1
  ETag: "-1621644187315998047"
  Cache-Control: max-age=60
  Date: Wed, 29 Feb 2012 19:17:47 GMT
  Content-Type: application/vnd.cdni.control.trigger.collection+json

  {
      "links": [
          {
              "href": "http://dcdn.example.com/triggers/0",
              "rel": "Trigger",
              "type": "application/vnd.cdni.control.trigger.status+json"
          },
          {
              "href": "http://dcdn.example.com/triggers/1",
              "rel": "Trigger",
              "type": "application/vnd.cdni.control.trigger.status+json"
          }
      ],
      "staleresourcetime": 86400
  }
```

7.2.2. Filtered Collections of Triggers

The filtered collections are also available to uCDN. Before dCDN starts processing the two triggers shown above, both will appear in the collection of Pending Triggers, for example:

REQUEST:

```
  GET /triggers/pending HTTP/1.1
  User-Agent: example-user-agent/0.1
  Host: dcdn.example.com
  Accept: */*
```

RESPONSE:

```
  HTTP/1.1 200 OK
  Content-Length: 418
  Expires: Wed, 29 Feb 2012 19:18:47 GMT
  Server: example-server/0.1
  ETag: "8328231265607503653"
  Cache-Control: max-age=60
  Date: Wed, 29 Feb 2012 19:17:47 GMT
  Content-Type: application/vnd.cdni.control.trigger.collection+json

  {
      "links": [
          {
              "href": "http://dcdn.example.com/triggers/0",
              "rel": "Trigger",
              "type": "application/vnd.cdni.control.trigger.status+json"
          },
          {
              "href": "http://dcdn.example.com/triggers/1",
              "rel": "Trigger",
              "type": "application/vnd.cdni.control.trigger.status+json"
          }
      ],
      "staleresourcetime": 86400
  }
```

At this point, if no other triggers had been created, the other filtered views of the triggers would be empty. For example:

REQUEST:

```
  GET /triggers/complete HTTP/1.1
  User-Agent: example-user-agent/0.1
  Host: dcdn.example.com
  Accept: */*
```

RESPONSE:

```
  HTTP/1.1 200 OK
  Content-Length: 53
  Expires: Wed, 29 Feb 2012 19:18:47 GMT
  Server: example-server/0.1
  ETag: "-654105208640281650"
  Cache-Control: max-age=60
  Date: Wed, 29 Feb 2012 19:17:47 GMT
  Content-Type: application/vnd.cdni.control.trigger.collection+json

  {
      "links": [],
      "staleresourcetime": 86400
  }
```

7.2.3. Trigger Status Resources

The Trigger Status Resources can also be examined for detail about individual triggers. For example, for the "preposition" and "invalidate" triggers from previous examples:

REQUEST:

```
  GET /triggers/0 HTTP/1.1
  User-Agent: example-user-agent/0.1
  Host: dcdn.example.com
  Accept: */*
```

RESPONSE:

```
  HTTP/1.1 200 OK
  Content-Length: 472
  Expires: Wed, 29 Feb 2012 19:18:47 GMT
  Server: example-server/0.1
  ETag: "-3658577240367174361"
  Cache-Control: max-age=60
  Date: Wed, 29 Feb 2012 19:17:47 GMT
  Content-Type: application/vnd.cdni.control.trigger.status+json

  {
      "ctime": 1330543067,
      "etime": 1330543073,
      "mtime": 1330543067,
      "status": "pending",
      "trigger": {
          "content.urls": [
              "http://www.example.com/a/b/c/1",
              "http://www.example.com/a/b/c/2",
              "http://www.example.com/a/b/c/3",
              "http://www.example.com/a/b/c/4"
          ],
          "metadata.urls": [
              "http://metadata.example.com/a/b/c"
          ],
          "type": "preposition"
      }
  }
```

REQUEST:

```
  GET /triggers/1 HTTP/1.1
  User-Agent: example-user-agent/0.1
  Host: dcdn.example.com
  Accept: */*
```

RESPONSE:

```
  HTTP/1.1 200 OK
  Content-Length: 596
  Expires: Wed, 29 Feb 2012 19:18:48 GMT
  Server: example-server/0.1
  ETag: "-6033881976191270З1"
  Cache-Control: max-age=60
  Date: Wed, 29 Feb 2012 19:17:48 GMT
  Content-Type: application/vnd.cdni.control.trigger.status+json

  {
      "ctime": 1330543067,
      "etime": 1330543073,
      "mtime": 1330543067,
      "status": "pending",
      "trigger": {
          "content.patterns": [
              {
                  "pattern.flags": 3,
                  "pattern.string": "http://www.example.com/a/b/"
              }
          ],
          "content.urls": [
              "http://www.example.com/a/index.html"
          ],
          "metadata.patterns": [
              {
                  "pattern.flags": 3,
                  "pattern.string": "http://metadata.example.com/a/b/"
              }
          ],
          "type": "invalidate"
      }
  }
```

7.2.4. Polling for Change

The uCDN may use the Entity Tags of collections or resources when polling for change in status, as shown in the following examples:
REQUEST:

```
  GET /triggers/pending HTTP/1.1
  User-Agent: example-user-agent/0.1
  Host: dcdn.example.com
  Accept: */*
  If-None-Match: "8328231265607503653"
```

RESPONSE:

```
  HTTP/1.1 304 Not Modified
  Content-Length: 0
  Expires: Wed, 29 Feb 2012 19:18:47 GMT
  Server: example-server/0.1
  ETag: "8328231265607503653"
  Cache-Control: max-age=60
  Date: Wed, 29 Feb 2012 19:17:47 GMT
  Content-Type: application/vnd.cdni.control.trigger.collection+json
```

REQUEST:

```
  GET /triggers/0 HTTP/1.1
  User-Agent: example-user-agent/0.1
  Host: dcdn.example.com
  Accept: */*
  If-None-Match: "-3658577240367174361"
```

RESPONSE:

```
  HTTP/1.1 304 Not Modified
  Content-Length: 0
  Expires: Wed, 29 Feb 2012 19:18:47 GMT
  Server: example-server/0.1
  ETag: "-3658577240367174361"
  Cache-Control: max-age=60
  Date: Wed, 29 Feb 2012 19:17:47 GMT
  Content-Type: application/vnd.cdni.control.trigger.status+json
```

When the triggered activity is complete, the contents of the filtered collections will be updated, along with their Entity Tags. For

example, when the two example triggers are complete, the collections of pending and complete triggers may look like:
REQUEST:

```
  GET /triggers/pending HTTP/1.1
  User-Agent: example-user-agent/0.1
  Host: dcdn.example.com
  Accept: */*
```

RESPONSE:

```
  HTTP/1.1 200 OK
  Content-Length: 53
  Expires: Wed, 29 Feb 2012 19:18:56 GMT
  Server: example-server/0.1
  ETag: "-7056231826368088123"
  Cache-Control: max-age=60
  Date: Wed, 29 Feb 2012 19:17:56 GMT
  Content-Type: application/vnd.cdni.control.trigger.collection+json

  {
      "links": [],
      "staleresourcetime": 86400
  }
```

```
REQUEST:

  GET /triggers/complete HTTP/1.1
  User-Agent: example-user-agent/0.1
  Host: dcdn.example.com
  Accept: */*


RESPONSE:

  HTTP/1.1 200 OK
  Content-Length: 418
  Expires: Wed, 29 Feb 2012 19:18:56 GMT
  Server: example-server/0.1
  ETag: "1388228818267892536"
  Cache-Control: max-age=60
  Date: Wed, 29 Feb 2012 19:17:56 GMT
  Content-Type: application/vnd.cdni.control.trigger.collection+json

  {
      "links": [
          {
              "href": "http://dcdn.example.com/triggers/0",
              "rel": "Trigger",
              "type": "application/vnd.cdni.control.trigger.status+json"
          },
          {
              "href": "http://dcdn.example.com/triggers/1",
              "rel": "Trigger",
              "type": "application/vnd.cdni.control.trigger.status+json"
          }
      ],
      "staleresourcetime": 86400
  }
```

7.2.5. Cancelling or Removing a Trigger

To request dCDN to cancel a Trigger, uCDN may delete the Trigger Resource. It may also delete completed and failed triggers to reduce the size of the collections. For example, to remove the "preposition" request from earlier examples:

```
   REQUEST:

     DELETE /triggers/0 HTTP/1.1
     User-Agent: example-user-agent/0.1
     Host: dcdn.example.com
     Accept: */*


   RESPONSE:

     HTTP/1.1 204 No Content
     Date: Wed, 29 Feb 2012 19:17:56 GMT
     Content-Length: 0
     Content-Type: text/html; charset=UTF-8
     Server: example-server/0.1
```

This would, for example, cause the collection of completed triggers shown in the example above to be updated to:

```
REQUEST:

  GET /triggers/complete HTTP/1.1
  User-Agent: example-user-agent/0.1
  Host: dcdn.example.com
  Accept: */*


RESPONSE:

  HTTP/1.1 200 OK
  Content-Length: 237
  Expires: Wed, 29 Feb 2012 19:18:57 GMT
  Server: example-server/0.1
  ETag: "-8850203857096517156"
  Cache-Control: max-age=60
  Date: Wed, 29 Feb 2012 19:17:57 GMT
  Content-Type: application/vnd.cdni.control.trigger.collection+json

  {
      "links": [
          {
              "href": "http://dcdn.example.com/triggers/1",
              "rel": "Trigger",
              "type": "application/vnd.cdni.control.trigger.status+json"
          }
      ],
      "staleresourcetime": 86400
  }
```

8. IANA Considerations

TBD.

9. Security Considerations

The dCDN must ensure that each uCDN only has access to its own Trigger Status Resources.

It is anticipated that a common authentication mechanism will be used by this and other CDNI Interconnect interfaces, the mechanism must exist but is not identified in this document.

The dCDN must ensure that activity triggered by uCDN only affects metadata or content originating from that uCDN.

10. Acknowledgements

TBD.

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.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

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

11.2. Informative References

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

[I-D.ietf-cdni-problem-statement]
Niven-Jenkins, B., Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem

Statement", draft-ietf-cdni-problem-statement-03 (work in progress), January 2012.

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

[I-D.jenkins-cdni-metadata]
Niven-Jenkins, B., Ferguson, D., and G. Watson, "CDN Interconnect Metadata", draft-jenkins-cdni-metadata-00 (work in progress), September 2011.

[RFC4287] Nottingham, M., Ed. and R. Sayre, Ed., "The Atom Syndication Format", RFC 4287, December 2005.

[XML-BASE]
Marsh, J., Ed. and R. Tobin, Ed., "XML Base (Second Edition) - http://www.w3.org/TR/xmlbase/", January 2009.

Authors' Addresses

Rob Murray
Velocix (Alcatel-Lucent)
326 Cambridge Science Park
Milton Road, Cambridge CB4 0WG
UK

Email: rmurray@velocix.com

Ben Niven-Jenkins
Velocix (Alcatel-Lucent)
326 Cambridge Science Park
Milton Road, Cambridge CB4 0WG
UK

Email: ben@velocix.com