

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: June 5, 2014

R. Murray  
B. Niven-Jenkins  
Velocix (Alcatel-Lucent)  
December 2, 2013

**CDNI Control Interface / Triggers**  
**draft-ietf-cdni-control-triggers-02**

Abstract

This document describes the part of the CDN Interconnect Control Interface that allows 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

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on June 5, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal

## Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	<a href="#">Introduction . . . . .</a>	<a href="#">4</a>
<a href="#">1.1.</a>	<a href="#">Terminology . . . . .</a>	<a href="#">5</a>
<a href="#">2.</a>	<a href="#">Model for CDNI Triggers . . . . .</a>	<a href="#">5</a>
<a href="#">2.1.</a>	<a href="#">Timing of Triggered Activity . . . . .</a>	<a href="#">7</a>
<a href="#">2.2.</a>	<a href="#">Trigger Results . . . . .</a>	<a href="#">7</a>
<a href="#">3.</a>	<a href="#">Collections of Trigger Status Resources . . . . .</a>	<a href="#">7</a>
<a href="#">4.</a>	<a href="#">CDNI Trigger interface . . . . .</a>	<a href="#">8</a>
<a href="#">4.1.</a>	<a href="#">Creating Triggers . . . . .</a>	<a href="#">9</a>
<a href="#">4.2.</a>	<a href="#">Checking Status . . . . .</a>	<a href="#">10</a>
<a href="#">4.2.1.</a>	<a href="#">Polling Trigger Status Resource collections . . . . .</a>	<a href="#">11</a>
<a href="#">4.2.2.</a>	<a href="#">Polling Trigger Status Resources . . . . .</a>	<a href="#">11</a>
<a href="#">4.3.</a>	<a href="#">Deleting Triggers . . . . .</a>	<a href="#">11</a>
<a href="#">4.4.</a>	<a href="#">Expiry of Trigger Status Resources . . . . .</a>	<a href="#">12</a>
<a href="#">4.5.</a>	<a href="#">Error Handling . . . . .</a>	<a href="#">12</a>
<a href="#">5.</a>	<a href="#">Properties of Triggers . . . . .</a>	<a href="#">13</a>
<a href="#">5.1.</a>	<a href="#">Properties of Trigger Requests . . . . .</a>	<a href="#">13</a>
<a href="#">5.1.1.</a>	<a href="#">Content URLs . . . . .</a>	<a href="#">14</a>
<a href="#">5.2.</a>	<a href="#">Properties of Trigger Status Resources . . . . .</a>	<a href="#">14</a>
<a href="#">5.3.</a>	<a href="#">Properties of ErrorDesc . . . . .</a>	<a href="#">15</a>
<a href="#">5.4.</a>	<a href="#">Properties of Trigger Collections . . . . .</a>	<a href="#">15</a>
<a href="#">5.5.</a>	<a href="#">Trigger Resource Simple Data Type Descriptions . . . . .</a>	<a href="#">16</a>
<a href="#">5.5.1.</a>	<a href="#">TriggerType . . . . .</a>	<a href="#">16</a>
<a href="#">5.5.2.</a>	<a href="#">TriggerStatus . . . . .</a>	<a href="#">16</a>
<a href="#">5.5.3.</a>	<a href="#">URLs . . . . .</a>	<a href="#">16</a>
<a href="#">5.5.4.</a>	<a href="#">AbsoluteTime . . . . .</a>	<a href="#">17</a>
<a href="#">5.5.5.</a>	<a href="#">ErrorCode . . . . .</a>	<a href="#">17</a>
<a href="#">6.</a>	<a href="#">JSON Encoding of Objects . . . . .</a>	<a href="#">17</a>
<a href="#">6.1.</a>	<a href="#">JSON Encoding of Embedded Types . . . . .</a>	<a href="#">18</a>
<a href="#">6.1.1.</a>	<a href="#">TriggerType . . . . .</a>	<a href="#">18</a>
<a href="#">6.1.2.</a>	<a href="#">TriggerStatus . . . . .</a>	<a href="#">18</a>
<a href="#">6.1.3.</a>	<a href="#">PatternMatch . . . . .</a>	<a href="#">18</a>
<a href="#">6.1.4.</a>	<a href="#">ErrorDesc . . . . .</a>	<a href="#">19</a>
<a href="#">6.1.5.</a>	<a href="#">ErrorCode . . . . .</a>	<a href="#">19</a>
<a href="#">6.1.6.</a>	<a href="#">Relationship . . . . .</a>	<a href="#">19</a>
<a href="#">6.1.7.</a>	<a href="#">Link Object . . . . .</a>	<a href="#">20</a>
<a href="#">6.2.</a>	<a href="#">MIME Media Types . . . . .</a>	<a href="#">20</a>



<a href="#">7.</a>	<a href="#">Examples . . . . .</a>	<a href="#">20</a>
<a href="#">7.1.</a>	<a href="#">Creating Triggers . . . . .</a>	<a href="#">21</a>
<a href="#">7.1.1.</a>	<a href="#">Preposition . . . . .</a>	<a href="#">21</a>
<a href="#">7.1.2.</a>	<a href="#">Invalidate . . . . .</a>	<a href="#">22</a>
<a href="#">7.2.</a>	<a href="#">Examining Trigger Status . . . . .</a>	<a href="#">23</a>
<a href="#">7.2.1.</a>	<a href="#">Collection of All Triggers . . . . .</a>	<a href="#">24</a>
<a href="#">7.2.2.</a>	<a href="#">Filtered Collections of Triggers . . . . .</a>	<a href="#">25</a>
<a href="#">7.2.3.</a>	<a href="#">Trigger Status Resources . . . . .</a>	<a href="#">26</a>
<a href="#">7.2.4.</a>	<a href="#">Polling for Change . . . . .</a>	<a href="#">28</a>
<a href="#">7.2.5.</a>	<a href="#">Cancelling or Removing a Trigger . . . . .</a>	<a href="#">31</a>
<a href="#">7.2.6.</a>	<a href="#">Error Reporting . . . . .</a>	<a href="#">33</a>
<a href="#">8.</a>	<a href="#">IANA Considerations . . . . .</a>	<a href="#">34</a>
<a href="#">8.1.</a>	<a href="#">CI/T MIME Media Types . . . . .</a>	<a href="#">35</a>
<a href="#">9.</a>	<a href="#">Security Considerations . . . . .</a>	<a href="#">35</a>
<a href="#">10.</a>	<a href="#">Acknowledgements . . . . .</a>	<a href="#">35</a>
<a href="#">11.</a>	<a href="#">References . . . . .</a>	<a href="#">35</a>
<a href="#">11.1.</a>	<a href="#">Normative References . . . . .</a>	<a href="#">35</a>
<a href="#">11.2.</a>	<a href="#">Informative References . . . . .</a>	<a href="#">36</a>
	<a href="#">Authors' Addresses . . . . .</a>	<a href="#">36</a>



## 1. Introduction

[RFC6707] 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.ietf-cdni-framework] expands on the information provided in [RFC6707] and describes each of the interfaces and the relationships between them in more detail.

This document describes the "CI/T" interface, "CDNI Control interface / Triggers". It does not consider those parts of the control interface that relate to configuration, bootstrapping or authentication of CDN Interconnect interfaces. Requirements for CI/T are the "High" and "Medium" priority requirements for the CI identified in section 4 of [I-D.ietf-cdni-requirements], reproduced here for convenience:

CI-1 [HIGH] The CDNI Control interface shall allow the Upstream CDN to request that the Downstream CDN, including downstream cascaded CDNs, delete an object or set of objects 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.

CI-2 [MED] The CDNI Control interface should allow for multiple content items identified by a Content Collection ID to be purged using a single Content Purge action.

CI-3 [MED] The CDNI Control interface should allow the Upstream CDN to request that the Downstream CDN, including downstream cascaded CDNs, mark an object or set of objects and/or its CDNI metadata as "stale" and revalidate them before they are delivered again.

CI-4 [HIGH] The CDNI Control interface shall allow the Downstream CDN to report on the completion of these actions (by itself, and including downstream cascaded CDNs, in a manner appropriate for the action (e.g. synchronously or asynchronously). The confirmation receipt should include a success or failure indication. The failure indication along with the reason are used if the Downstream CDN cannot delete the content in its storage.

CI-5 [MED] The CDNI Control interface should support initiation and control by the Upstream CDN of pre-positioned CDNI metadata acquisition by the Downstream CDN.

CI-6 [MED] The CDNI Control interface should support initiation and control by the Upstream CDN of pre-positioned content acquisition by the Downstream CDN.



- o [Section 2](#) outlines the model for the CI/T Interface at a high level.
- o [Section 3](#) describes collections of Trigger Resources.
- o [Section 4](#) defines the RESTful web service provided by dCDN.
- o [Section 5](#) lists properties of Trigger Requests and Status Resources.
- o [Section 6](#) defines a JSON encoding for Trigger Requests and Status Resources.
- o [Section 7](#) contains example messages.

### **[1.1.](#) Terminology**

This document reuses the terminology defined in [[RFC6707](#)].

## **[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 content, the following actions can be requested:

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

The CI/T 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](#).

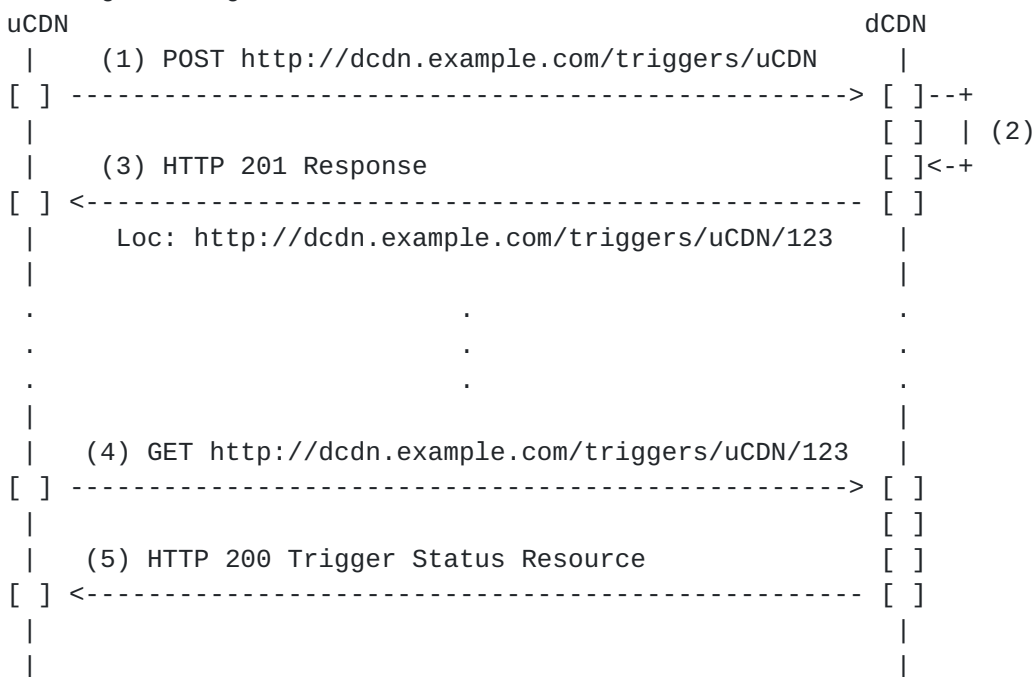


Figure 1: Basic CDNI Message Flow for Triggers

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.
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.

### **[2.1.](#) Timing of Triggered Activity**

Timing of triggered activity is under dCDN control, including its start-time and pacing of the activity in the network.

Invalidate and purge triggers MUST be applied to all data acquired before the trigger was created in dCDN. The dCDN MAY apply the triggers to data acquired after trigger creation.

If uCDN wishes to invalidate or purge content, then immediately preposition replacement content at the same URLs, it must ensure the dCDN has completed the invalidate/purge before initiating the prepositioning. If it fails to do that and the requests overlap, and dCDN passes the triggers on to a further dCDN in a cascade, that CDN may preposition content that has not yet been invalidated/purged in its uCDN.

### **[2.2.](#) Trigger Results**

Each Trigger Request may operate on multiple data items. The trigger shall be reported as "complete" only if all actions can be completed successfully, otherwise it shall be reported as "failed". The reasons for failure and URLs or Patterns affected shall be enumerated in the Trigger Status Resource. For more detail, see section [Section 4.5](#).

If a dCDN is also acting as uCDN in a cascade, it MUST forward triggers to any downstream CDNs that may have data affected by the trigger. The trigger MUST NOT be reported as complete in a CDN until it is complete in all of its downstream CDNs. A trigger MAY be reported as failed as soon as it fails in a CDN or in any of its downstream CDNs.

## **[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:

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

CI/T 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 CI/T that contain a response body SHOULD include an ETag to enable validation of cached versions of returned resources.



Servers implementing CI/T 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 CI/T servers are free to assign whatever structure they desire to the URIs for CI/T resources. CI/T clients MUST NOT make any assumptions regarding the structure of CI/T URIs or the mapping between CI/T 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 CI/T interface implementations.

The CI/T interface builds on top of HTTP, so CI/T servers may make use of any HTTP feature when implementing the CI/T interface. For example, a CI/T 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.

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

The dCDN must ensure that activity triggered by uCDN only affects metadata or content originating from that uCDN. Since only one CDN can be authoritative for a given item of metadata or content, this requirement means there cannot be any "loops" in trigger requests between CDNs.

#### **[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.

If dCDN is not able to track triggered activity, it MAY indicate that it has undertaken to complete the activity but will not report completion or any further errors. To do this, it must set the trigger status to "complete", with an estimated completion time in the future ("etime" greater than "mtime").

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 check for change in status of a resource or collection of resources without re-fetching the whole resource or collection.

The dCDN should use the cache control headers for responses to GETs for Trigger Status Resources and Collections to indicate the frequency at which it recommends uCDN should poll for change.

#### **[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. Subsequent requests for the resource shall be handled as required by HTTP, and so will receive responses with status 404 or 410.

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.

#### **4.5. Error Handling**

A CI/T server may reject a trigger request using HTTP status codes, for example 400 if the request is malformed or 401 if the client does not have permission to create triggers or it is trying to act on another CDN's data.

If any part of the trigger request fails the trigger shall be reported as "failed" once its activity is complete, or if no further errors will be reported. The "errors" property in the Trigger Status Resource will be used to enumerate which actions failed and the reasons for failure, and may be present while the trigger is still "pending" or "active" if the trigger is still running for some URLs or Patterns in the trigger request.

Once a request has been accepted, processing errors are reported in the Trigger Status Resource using a list of "ErrorDesc". Each ErrorDesc is used to report errors against one or more of the URLs or Patterns in the trigger request.

If a surrogate affected by a trigger is offline in dCDN, or dCDN is unable to pass a trigger request on to any of its affected dCDNs; dCDN should report an error if the request is abandoned, otherwise it



must keep the trigger in state "pending" or "active" until it's acted upon or uCDN chooses to cancel it. Or, if the request is queued and dCDN will not report further status, dCDN may report the trigger as "complete" with an "etime" in the future.

Note that an "invalidate" trigger may be reported as "complete" when surrogates that may have the data are offline, if those surrogates will not use the affected data without first revalidating it when they are back online. This does not apply to "preposition" or "purge" triggers.

## **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: content.ccid

Description: The Content Collection IDentifier of data the trigger applies to.

Type: List of strings

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: List of PatternMatch

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: List of PatternMatch

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**

To refer to content in dCDN, uCDN must present URLs in the same form clients will use to access content in that dCDN, after transformation to remove any surrogate-specific parts of a 302-redirect URL form. By definition, it is always possible to locate content based on URLs in this form.

If content URLs are transformed by an intermediate CDN in a cascade, that intermediate CDN must transform URLs in trigger requests it passes to its dCDN.

When processing trigger requests, CDNs SHOULD ignore the URL scheme (http or https) in comparing URLs. For example, for an invalidate or purge trigger, content may be invalidated or purged regardless of the protocol clients use to request it.

#### **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: errors

Description: List of ErrorDesc.

Mandatory: No.

### **5.3. Properties of ErrorDesc**

An ErrorDesc object is used to report failure for URLs and patterns in a trigger request.

Property: error

Type: ErrorCode.

Mandatory: Yes.

Property: metadata.urls, content.urls, metadata.patterns, content.patterns

Description: Metadata and content references copied from the trigger request. Only those URLs and patterns to which the error applies shall be included in each property, but those URLs and patterns shall be exactly as they appear in the request, dCDN must not generalise the URLs. (For example, if uCDN requests prepositioning of URLs

"http://ucdn.example.com/a" and "http://ucdn.example.com/b", dCDN may not generalise its error report to Pattern

"http://ucdn.example.com/\*").

Mandatory: At least one of these properties is mandatory in each ErrorDesc.

Property: description

Description: A String containing a human-readable description of the error.

Mandatory: No.

### **5.4. 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.

## **5.5. Trigger Resource Simple Data Type Descriptions**

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

### **5.5.1. TriggerType**

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

- o Proposition - 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.

### **5.5.2. 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, or the trigger has been accepted and no further status update will be made.
- o Failed - the triggered activity could not be completed.

### **5.5.3. URLs**

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



#### **5.5.4. AbsoluteTime**

Times are expressed in seconds since the UNIX epoch.

#### **5.5.5. ErrorCode**

This type is used by dCDN to report failures in trigger processing.

- o EMETA - dCDN was unable to acquire metadata required to fulfil the request.
- o ECONTENT - dCDN was unable to acquire content (preposition triggers only).
- o EPERM - uCDN does not have permission to trigger the requested activity (for example, the data is owned by another CDN).
- o EREJECT - dCDN is not willing to fulfil the request (for example, a preposition request for content at a time when dCDN would not accept Request Routing requests from uCDN).
- o ECDN - An internal error in dCDN or one of its downstream CDNs.

### **6. JSON Encoding of Objects**

This encoding is based on that described in [[I-D.ietf-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 encoding for a CI/T 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).

The "trigger" property of the top level JSON object lists the requested action.

Key: trigger

Description: An object specifying the trigger type and a set of data to act upon.

Type: A JSON object.





Mandatory: Yes.

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

In addition to the properties of an 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: Array 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

### **[6.1.2.](#) TriggerStatus**

Key: status

Description: One of "pending", "active", "failed", "complete"

Type: string

### **[6.1.3.](#) PatternMatch**

A PatternMatch is encoded as a JSON Object containing a string to match and flags describing the type of match.

Key: pattern

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



Type: String  
Mandatory: Yes  
Key: case-sensitive  
Description: Flag indicating whether or not case-sensitive matching should be used.  
Type: Boolean  
Mandatory: No, default is case-insensitive match.  
Key: match-query-string  
Description: Flag indicating whether or not the query string should be included in the pattern match.  
Type: Boolean  
Mandatory: No, default is not to include query.  
Example of case-sensitive prefix match against  
"http://www.example.com/trailers/":  
{  
 "pattern": "http://www.example.com/trailers/\*",  
 "case-sensitive": true  
}

#### **6.1.4. ErrorDesc**

ErrorDesc shall be encoded as a JSON object with the following keys:

Key: error  
Type: ErrorCode  
Mandatory: Yes  
Keys: metadata.urls, content.urls  
Type: Array of strings  
Mandatory: At least one of metadata.\* or content.\* must be present.  
Keys: metadata.patterns, content.patterns  
Type: Array of PatternMatch  
Mandatory: At least one of metadata.\* or content.\* must be present.  
Key: description  
Type: String  
Mandatory: No.

#### **6.1.5. ErrorCode**

One of the strings "EMETA", "ECONTENT", "EPERM", "EREJECT" or "ECDN".

#### **6.1.6. Relationship**

The key "\_links" in a dictionary object may be used to define relationships to other resources. Keys of the "\_links" dictionary are link relation types, the value for each relation type can either be a Link Object or an array of Link Objects.



The relation type "self" SHOULD be included, with the target being the containing resource.

#### 6.1.7. Link Object

A Link Object is a JSON dictionary containing the following keys:

- o "href" - With a value containing the URI of the of the addressable object being referenced. The "href" must be specified.
- o "type" - The MIME Media Type of the referenced object. It is optional to specify "type". See [Section 6.2](#) for the MIME Media Types of objects specified in this document.

#### 6.2. MIME Media Types

Table 1 lists the MIME Media Type for the trigger request, and each trigger object (resource) that is retrievable through the CI/T interface.

Data Object	MIME Media Type
TriggerRequest	application/cdni.ci.TriggerRequest+json
TriggerStatus	application/cdni.ci.TriggerStatus+json
TriggerCollection	application/cdni.ci.TriggerCollection+json

Table 1: MIME Media Types for CDNI Trigger resources

### 7. Examples

The following sections provide examples of different CI/T 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 "metadata.patterns" and "content.patterns" are not allowed in a preposition trigger request.

REQUEST:

```
POST /triggers HTTP/1.1
User-Agent: example-user-agent/0.1
Host: dcdn.example.com
Accept: */*
Content-Type: application/cdni.ci.TriggerRequest+json
Content-Length: 315

{
  "trigger" : {
    "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: Mon, 11 Nov 2013 03:28:27 GMT
Content-Length: 472
Content-Type: application/cdni.ci.TriggerStatus+json
Location: http://dcdn.example.com/triggers/0
```





Server: example-server/0.1

```
{
  "ctime": 1384140507,
  "etime": 1384140515,
  "mtime": 1384140507,
  "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/cdni.ci.TriggerRequest+json
Content-Length: 352

{
  "trigger" : {
    "type": "invalidate",

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

    "content.urls" : [ "http://www.example.com/a/index.html" ],
    "content.patterns" : [
```



```
    { "pattern" : "http://www.example.com/a/b/*",  
      "case-sensitive" : true  
    }  
  ]  
}  
}
```

**RESPONSE:**

```
HTTP/1.1 201 Created  
Date: Mon, 11 Nov 2013 03:28:28 GMT  
Content-Length: 551  
Content-Type: application/cdni.ci.TriggerStatus+json  
Location: http://dcdn.example.com/triggers/1  
Server: example-server/0.1
```

```
{  
  "ctime": 1384140508,  
  "etime": 1384140516,  
  "mtime": 1384140508,  
  "status": "pending",  
  "trigger": {  
    "content.patterns": [  
      {  
        "case-sensitive": true,  
        "pattern": "http://www.example.com/a/b/*"  
      }  
    ],  
    "content.urls": [  
      "http://www.example.com/a/index.html"  
    ],  
    "metadata.patterns": [  
      {  
        "pattern": "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: 489
Expires: Mon, 11 Nov 2013 03:29:28 GMT
Server: example-server/0.1
ETag: "8477575226503289820"
Cache-Control: max-age=60
Date: Mon, 11 Nov 2013 03:28:28 GMT
Content-Type: application/cdni.ci.TriggerCollection+json
```

```
{
  "_links": {
    "Trigger": [
      {
        "href": "http://dcdn.example.com/triggers/0",
        "type": "application/cdni.ci.TriggerStatus+json"
      },
      {
        "href": "http://dcdn.example.com/triggers/1",
        "type": "application/cdni.ci.TriggerStatus+json"
      }
    ],
    "self": {
      "href": "http://dcdn.example.com/triggers"
    }
  },
  "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: 497
Expires: Mon, 11 Nov 2013 03:29:28 GMT
Server: example-server/0.1
ETag: "-4197252672546627852"
Cache-Control: max-age=60
Date: Mon, 11 Nov 2013 03:28:28 GMT
Content-Type: application/cdni.ci.TriggerCollection+json
```

```
{
  "_links": {
    "Trigger": [
      {
        "href": "http://dcdn.example.com/triggers/0",
        "type": "application/cdni.ci.TriggerStatus+json"
      },
      {
        "href": "http://dcdn.example.com/triggers/1",
        "type": "application/cdni.ci.TriggerStatus+json"
      }
    ],
    "self": {
      "href": "http://dcdn.example.com/triggers/pending"
    }
  },
  "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: 151
Expires: Mon, 11 Nov 2013 03:29:28 GMT
Server: example-server/0.1
ETag: "-3759884165278932652"
Cache-Control: max-age=60
Date: Mon, 11 Nov 2013 03:28:28 GMT
Content-Type: application/cdni.ci.TriggerCollection+json
```

```
{
  "_links": {
    "self": {
      "href": "http://dcdn.example.com/triggers/complete"
    }
  },
  "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: Mon, 11 Nov 2013 03:29:28 GMT
Server: example-server/0.1
ETag: "4936922742974586536"
Cache-Control: max-age=60
Date: Mon, 11 Nov 2013 03:28:28 GMT
Content-Type: application/cdni.ci.TriggerStatus+json
```

```
{
  "ctime": 1384140507,
  "etime": 1384140515,
  "mtime": 1384140507,
  "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: 551
Expires: Mon, 11 Nov 2013 03:29:28 GMT
Server: example-server/0.1
ETag: "-4441420523993853535"
Cache-Control: max-age=60
Date: Mon, 11 Nov 2013 03:28:28 GMT
Content-Type: application/cdni.ci.TriggerStatus+json
```

```
{
  "ctime": 1384140508,
  "etime": 1384140516,
  "mtime": 1384140508,
  "status": "pending",
  "trigger": {
    "content.patterns": [
      {
        "case-sensitive": true,
        "pattern": "http://www.example.com/a/b/*"
      }
    ],
    "content.urls": [
      "http://www.example.com/a/index.html"
    ],
    "metadata.patterns": [
      {
        "pattern": "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: "-4197252672546627852"
```

## RESPONSE:

```
HTTP/1.1 304 Not Modified
Content-Length: 0
Expires: Mon, 11 Nov 2013 03:29:28 GMT
Server: example-server/0.1
ETag: "-4197252672546627852"
Cache-Control: max-age=60
Date: Mon, 11 Nov 2013 03:28:28 GMT
Content-Type: application/cdni.ci.TriggerCollection+json
```

## REQUEST:

```
GET /triggers/0 HTTP/1.1
User-Agent: example-user-agent/0.1
Host: dcdn.example.com
Accept: */*
If-None-Match: "4936922742974586536"
```

## RESPONSE:

```
HTTP/1.1 304 Not Modified
Content-Length: 0
Expires: Mon, 11 Nov 2013 03:29:28 GMT
Server: example-server/0.1
ETag: "4936922742974586536"
Cache-Control: max-age=60
Date: Mon, 11 Nov 2013 03:28:28 GMT
Content-Type: application/cdni.ci.TriggerStatus+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: 150
Expires: Mon, 11 Nov 2013 03:29:39 GMT
Server: example-server/0.1
ETag: "-8587750650096537234"
Cache-Control: max-age=60
Date: Mon, 11 Nov 2013 03:28:39 GMT
Content-Type: application/cdni.ci.TriggerCollection+json
```

```
{
  "_links": {
    "self": {
      "href": "http://dcdn.example.com/triggers/pending"
    }
  },
  "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: 498
Expires: Mon, 11 Nov 2013 03:29:39 GMT
Server: example-server/0.1
ETag: "2680225545549998872"
Cache-Control: max-age=60
Date: Mon, 11 Nov 2013 03:28:39 GMT
Content-Type: application/cdni.ci.TriggerCollection+json
```

```
{
  "_links": {
    "Trigger": [
      {
        "href": "http://dcdn.example.com/triggers/0",
        "type": "application/cdni.ci.TriggerStatus+json"
      },
      {
        "href": "http://dcdn.example.com/triggers/1",
        "type": "application/cdni.ci.TriggerStatus+json"
      }
    ],
    "self": {
      "href": "http://dcdn.example.com/triggers/complete"
    }
  },
  "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: Mon, 11 Nov 2013 03:28:39 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: 304
Expires: Mon, 11 Nov 2013 03:29:39 GMT
Server: example-server/0.1
ETag: "535044172999094664"
Cache-Control: max-age=60
Date: Mon, 11 Nov 2013 03:28:39 GMT
Content-Type: application/cdni.ci.TriggerCollection+json
```

```
{
  "_links": {
    "Trigger": {
      "href": "http://dcdn.example.com/triggers/1",
      "type": "application/cdni.ci.TriggerStatus+json"
    },
    "self": {
      "href": "http://dcdn.example.com/triggers/complete"
    }
  },
  "staleresourcetime": 86400
}
```

#### [7.2.6.](#) Error Reporting

In this example uCDN has requested prepositioning of "http://newsite.example.com/index.html", but dCDN was unable to locate metadata for that site:





## REQUEST:

```
GET /triggers/2 HTTP/1.1
User-Agent: example-user-agent/0.1
Host: dcdn.example.com
Accept: */*
```

## RESPONSE:

```
HTTP/1.1 200 OK
Content-Length: 505
Expires: Mon, 11 Nov 2013 03:29:43 GMT
Server: example-server/0.1
ETag: "3841389629056746224"
Cache-Control: max-age=60
Date: Mon, 11 Nov 2013 03:28:43 GMT
Content-Type: application/cdni.ci.TriggerStatus+json
```

```
{
  "ctime": 1384140519,
  "errors": [
    {
      "content.urls": [
        "http://newsite.example.com/index.html"
      ],
      "description":
        "No HostIndex entry found for newsite.example.com",
      "error": "EMETA"
    }
  ],
  "etime": 1384140527,
  "mtime": 1384140523,
  "status": "active",
  "trigger": {
    "content.urls": [
      "http://newsite.example.com/index.html"
    ],
    "type": "preposition"
  }
}
```

## 8. IANA Considerations



### **8.1. CI/T MIME Media Types**

The IANA is requested to allocate the following MIME Media Types in the MIME Media Types registry:

- o application/cdni.ci.TriggerRequest
- o application/cdni.ci.TriggerStatus
- o application/cdni.ci.TriggerCollection

Use of these types is specified in [Section 6.2](#) of the present document.

## **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**

The structure of the Relationship and Link Objects specified in [Section 6](#) is based on Mike Kelly's work on JSON Hypertext Application Language.

## **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.ietf-cdni-framework]  
Peterson, L. and B. Davie, "Framework for CDN Interconnection", [draft-ietf-cdni-framework-07](#) (work in progress), November 2013.
- [I-D.ietf-cdni-metadata]  
Niven-Jenkins, B., Murray, R., Watson, G., Caulfield, M., Leung, K., and K. Ma, "CDN Interconnect Metadata", [draft-ietf-cdni-metadata-03](#) (work in progress), October 2013.
- [I-D.ietf-cdni-requirements]  
Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", [draft-ietf-cdni-requirements-13](#) (work in progress), November 2013.
- [RFC4287] Nottingham, M., Ed. and R. Sayre, Ed., "The Atom Syndication Format", [RFC 4287](#), December 2005.
- [RFC6707] Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", [RFC 6707](#), September 2012.
- [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)  
3 Ely Road  
Milton, Cambridge CB24 6DD  
UK  
  
Email: [rmurray@velocix.com](mailto:rmurray@velocix.com)



Ben Niven-Jenkins  
Velocix (Alcatel-Lucent)  
3 Ely Road  
Milton, Cambridge CB24 6DD  
UK  
  
Email: [ben@velocix.com](mailto:ben@velocix.com)