CDNI Control Interface / Triggers
draft-ietf-cdni-control-triggers-11

Abstract

   This document describes the part of the CDN Interconnection 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 invalidates or
   purges 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 9, 2016.

Copyright Notice

Table of Contents

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).

   [RFC7336] 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. Section 4 of
   [RFC7337] identifies the requirements specific to the CI interface,
   requirements applicable to the CI/T interface are CI-1 to CI-6.

   o Section 2 outlines the model for the CI/T Interface at a high
      level.

   o Section 3 describes collections of Trigger Status Resources.

   o Section 4 defines the web service provided by the dCDN.

   o Section 5 lists properties of CI/T Commands and Status Resources.

   o Section 6 contains example messages.

1.1. Terminology

This document reuses the terminology defined in [RFC6707] and uses
"uCDN" and "dCDN" as shorthand for "Upstream CDN" and "Downstream
CDN", respectively.

2. Model for CDNI Triggers

A CI/T Command, sent from the uCDN to the dCDN, is a request for the
dCDN to do some work relating to data associated with content
requests originating from the uCDN.

There are two types of CI/T Command: CI/T Trigger Commands, and CI/T
Cancel Commands. The CI/T Cancel Command can be used to request
cancellation of an earlier CI/T Trigger Command. A CI/T Trigger
Command is of one of the following types:

o preposition - used to instruct the dCDN to fetch metadata from the
   uCDN, or content from any origin including the uCDN.

o invalidate - used to instruct the dCDN to revalidate specific
   metadata or content before re-using it.

o purge - used to instruct the dCDN to delete specific metadata or
   content.

The CI/T interface is a web service offered by the dCDN. It allows
CI/T commands to be issued, and triggered activity to be tracked.
When the dCDN accepts a CI/T Command it creates a resource describing
status of the triggered activity, a Trigger Status Resource. The
uCDN can poll Trigger Status Resources to monitor progress.

The dCDN maintains at least one collection of Trigger Status
Resources for each uCDN. Each uCDN only has access to its own
collections, the locations of which are shared when CDN
interconnection is established.

To trigger activity in the dCDN, the uCDN POSTs a CI/T Command to the
collection of Trigger Status Resources. If the dCDN accepts the CI/T
Command, it creates a new Trigger Status Resource and returns its
location to the uCDN. To monitor progress, the uCDN can GET the
Trigger Status Resource. To request cancellation of a CI/T Trigger
Command the uCDN can POST to the collection of Trigger Status
Resources, or simply DELETE the Trigger Status Resource.

In addition to the collection of all Trigger Status Resources for the
uCDN, the dCDN can maintain filtered views of that collection. These
filtered views are defined in Section 3 and include collections of

Trigger Status Resources corresponding to active and completed CI/T
Trigger Commands. 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 the dCDN, and for the uCDN to discover
the status of that activity. Only successful triggering is shown.
Examples of the messages are given in Section 6.

```
    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              [ ]
    [ ] <------------------------------------------------- [ ]
     |                                                        |
     |                                                        |
```
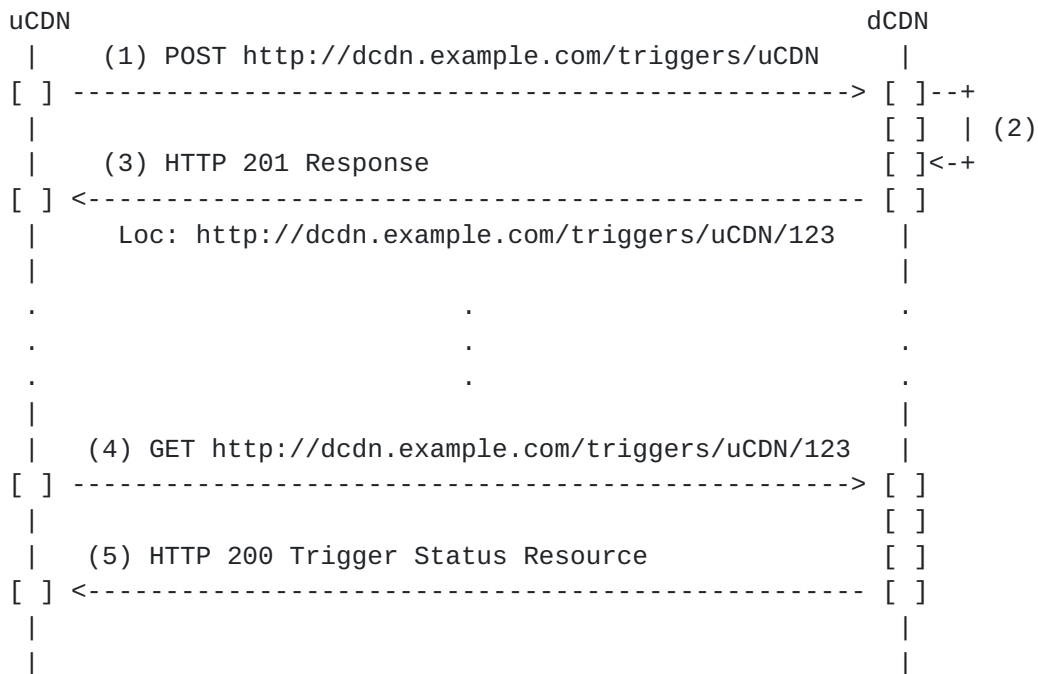
Figure 1: Basic CDNI Message Flow for Triggers

The steps in Figure 1 are:

1. The uCDN triggers action in the dCDN by posting a CI/T Command to
    a collection of Trigger Status Resources,
    "http://dcdn.example.com/triggers/uCDN". The URL of this was
    given to the uCDN when the CI/T interface was established.

2. The dCDN authenticates the request, validates the CI/T Command
    and, if it accepts the request, creates a new Trigger Status
    Resource.

3. The dCDN responds to the uCDN with an HTTP 201 response status,
    and the location of the Trigger Status Resource.

4. The uCDN can poll, possibly repeatedly, the Trigger Status
    Resource in the dCDN.

   5. The dCDN responds with the Trigger Status Resource, describing
       progress or results of the CI/T Trigger Command.

   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 the execution of CI/T Commands is under the dCDN's control,
   including its start-time and pacing of the activity in the network.

   CI/T invalidate and purge commands MUST be applied to all data
   acquired before the command was accepted by the dCDN. The dCDN
   SHOULD NOT apply CI/T invalidate and purge commands to data acquired
   after the CI/T Command was accepted, but this may not always be
   achievable so the uCDN cannot count on that.

   If the uCDN wishes to invalidate or purge content then immediately
   pre-position replacement content at the same URLs, it SHOULD ensure
   the dCDN has completed the invalidate/purge before initiating the
   prepositioning. Otherwise, there is a risk that the dCDN pre-
   positions the new content, then immediately invalidates or purges it
   (as a result of the two uCDN requests running in parallel).

   Because the CI/T Command timing is under the dCDN's control, the dCDN
   implementation can choose whether to apply CI/T invalidate and purge
   commands to content acquisition that has already started when the
   command is received.

2.2. Scope of Triggered Activity

   Each CI/T Command can operate on multiple metadata and content URLs.

   Multiple representations of an HTTP resource may share the same URL.
   CI/T Trigger Commands that invalidate or purge metadata or content
   apply to all resource representations with matching URLs.

   The dCDN MUST reject CI/T Commands from a uCDN that act on another
   uCDN's data. Security considerations are discussed further in
   section Section 8.

2.3. Trigger Results

   Possible states for a Trigger Status Resource are defined in section
   Section 5.2.3.

The CI/T Trigger Command MUST NOT be reported as 'complete' until all actions have been completed successfully. The reasons for failure, and URLs or Patterns affected, SHOULD be enumerated in the Trigger Status Resource. For more detail, see section Section 4.7.

If a dCDN is also acting as a uCDN in a cascade, it MUST forward CI/T Commands to any downstream CDNs that may be affected. The CI/T Trigger Command MUST NOT be reported as 'complete' in a CDN until it is 'complete' in all of its downstream CDNs. If a CI/T Trigger Command is reported as 'processed' in any dCDN, intermediate CDNs MUST NOT report 'complete', instead they must also report 'processed'. A CI/T Command MAY be reported as 'failed' as soon as it fails in a CDN or in any of its downstream CDNs. A cancelled CI/T Trigger Command MUST be reported as 'cancelling' until it has been reported as 'cancelled', 'complete', or 'failed' by all dCDNs in a cascade.

3. Collections of Trigger Status Resources

As described in Section 2, Trigger Status Resources exist in the 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.

The dCDN MUST make a collection of a uCDN's Trigger Status Resources available to that uCDN. This collection includes all of the Trigger Status Resources created for CI/T Commands from the uCDN that have been accepted by the dCDN, and have not yet been deleted by the uCDN, or expired and removed by the dCDN (as described in section Section 4.4). Trigger Status Resources belonging to a uCDN MUST NOT be visible to any other CDN. The dCDN could, for example, achieve this by offering different collection URLs to each uCDN, and/or by filtering the response based on the uCDN with which the HTTP client is associated.

To trigger activity in a dCDN, or to cancel triggered activity, the uCDN POSTs a CI/T Command to the dCDN's collection of the uCDN's Trigger Status Resources.

In order to allow the uCDN to check the status of multiple jobs in a single request, the dCDN SHOULD also maintain collections representing filtered views of the collection of all Trigger Status Resources. These filtered collections are optional-to-implement but, if implemented, the dCDN MUST include links to them in the collection of all Trigger Status Resources. The filtered collections are:

o Pending - Trigger Status Resources for CI/T Trigger Commands that
   have been accepted, but not yet acted upon.

o Active - Trigger Status Resources for CI/T Trigger Commands that
   are currently being processed in the dCDN.

o Complete - Trigger Status Resources representing activity that
   completed successfully, and 'processed' CI/T Trigger Commands for
   which no further status updates will be made by the dCDN.

o Failed - Trigger Status Resources representing CI/T Commands that
   failes or were cancelled by the uCDN.

4. CDNI Trigger Interface

This section describes an interface to enable an upstream CDN to
trigger activity in a downstream CDN.

The CI/T interface builds on top of HTTP, so dCDNs may make use of
any HTTP feature when implementing the CI/T interface. For example,
a dCDN SHOULD make use of HTTP's caching mechanisms to indicate that
a requested response/representation has not been modified, reducing
the uCDN's processing needed to determine whether the status of
triggered activity has changed.

All dCDNs implementing CI/T MUST support the HTTP GET, HEAD, POST and
DELETE methods as defined in [RFC7231].

The only representation specified in this document is JSON,
[RFC7159]. It MUST be supported by the uCDN and by the dCDN.

The URL of the dCDN's collection of all Trigger Status Resources
needs to be either discovered by, or configured in, the uCDN. The
mechanism for discovery of that URL is outside the scope of this
document.

CI/T Commands are POSTed to the dCDN's collection of all Trigger
Status Resources. If a CI/T Trigger Command is accepted by the dCDN,
the dCDN creates a new Trigger Status Resource and returns its URI to
the uCDN in an HTTP 201 response. The triggered activity can then be
monitored by the uCDN using that resource and the collections
described in Section 3.

The URI of each Trigger Status Resource is returned to the uCDN when
it is created, and URIs of all Trigger Status Resources are listed in
the dCDN's collection of all Trigger Status Resources. This means
all Trigger Status Resources can be discovered by the uCDN, so dCDNs
are free to assign whatever structure they desire to the URIs for CI/

T resources. Therefore uCDNs MUST NOT make any assumptions regarding
the structure of CI/T URIs or the mapping between CI/T objects and
their associated URIs. URIs present in the examples in this document
are purely illustrative and are not intended to impose a definitive
structure on CI/T interface implementations.

4.1. Creating Triggers

To issue a CI/T Command, the uCDN makes an HTTP POST to the dCDN's
collection of all of the uCDN's Trigger Status Resources. The
request body of that POST is a CI/T Command, as described in
Section 5.1.1.

The dCDN validates the CI/T Command. If the command is malformed or
the uCDN does not have sufficient access rights, the dCDN MUST either
respond with an appropriate 4xx HTTP error code and not create a
Trigger Status Resource, or create a 'failed' Trigger Status Resource
containing an appropriate error description.

When a CI/T Trigger Command is accepted, the uCDN MUST create a new
Trigger Status Resource which will convey a specification of the CI/T
Command and its current status. The HTTP response to the dCDN MUST
have status code 201 and MUST convey the URI of the Trigger Status
Resource in the Location header field. The HTTP response SHOULD
include the content of the newly created Trigger Status Resource.
This is particularly important in cases where the CI/T Trigger
Command has completed immediately.

Once a Trigger Status Resource has been created the dCDN MUST NOT re-
use its URI, even after that Trigger Status Resource has been
removed.

The dCDN SHOULD track and report on progress of CI/T Trigger
Commands. If the dCDN is not able to do that, it MUST indicate that
it has accepted the request but will not be providing further status
updates. To do this, it sets the "status" of the Trigger Status
Resource to "processed". In this case, CI/T processing should
continue as for a "complete" request, so the Trigger Status Resource
MUST be added to the dCDN's collection of Complete Trigger Status
Resources. The dCDN SHOULD also provide an estimated completion time
for the request, by using the "etime" property of the Trigger Status
Resource. This will allow the uCDN to schedule prepositioning after
an earlier delete of the same URLs is expected to have finished.

If the dCDN is able to track the execution of CI/T Commands and a CI/
T Command is queued by the dCDN for later action, the "status"
property of the Trigger Status Resource MUST be "pending". Once
processing has started the "status" MUST be "active". Finally, once

the CI/T Command is complete, the status MUST be set to "complete" or
"failed".

A CI/T Trigger Command may result in no activity in the dCDN if, for
example, it is an invalidate or purge request for data the dCDN has
not yet acquired, or a prepopulate request for data it has already
acquired and which is still valid. In this case, the "status" of the
Trigger Status Resource MUST be "processed" or "complete", and the
Trigger Status Resource MUST be added to the dCDN's collection of
Complete Trigger Status Resources.

Once created, Trigger Status Resources can be cancelled or deleted by
the uCDN, but not modified. The dCDN MUST reject PUT and POST
requests from the uCDN to Trigger Status Resources by responding with
an appropriate HTTP status code, for example 405 "Method Not
Allowed".

4.2. Checking Status

The uCDN has two ways to check progress of CI/T Commands it has
issued to the dCDN, described in sections Section 4.2.1 and
Section 4.2.2.

To allow the uCDN to check for change in status of a Trigger Status
Resource or collection of Trigger Status Resources without re-
fetching the whole Resource or Collection, the dCDN SHOULD include
Entity Tags for the uCDN to use as cache validators, as defined in
[RFC7232].

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 the uCDN should poll for change.

4.2.1. Polling Trigger Status Resource collections

The 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 CI/T Trigger Commands in
a single request. If the dCDN moves a Trigger Status Resource from
the Active to the Completed collection, the uCDN can fetch the result
of that activity.

When polling in this way, the uCDN SHOULD use HTTP Entity Tags to
monitor for change, rather than repeatedly fetching the whole
collection. An example of this is given in section Section 6.2.4.

4.2.2. Polling Trigger Status Resources

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

   This can be used to retrieve progress information, and to fetch the
   result of the CI/T Command.

   When polling in this way, the uCDN SHOULD use HTTP Entity Tags to
   monitor for change, rather than repeatedly fetching the Trigger
   Status Resource.

4.3. Cancelling Triggers

   The uCDN can request cancellation of a CI/T Trigger Command by
   POSTing a CI/T Cancel Command to the collection of all Trigger Status
   Resources.

   The dCDN is required to accept and respond to the CI/T Cancel
   Command, but the actual cancellation of a CI/T Trigger Command is
   optional-to-implement.

   The dCDN MUST respond to the CI/T Cancel Command appropriately, for
   example with HTTP status code 200 "OK" if the cancellation has been
   processed and the CI/T Command is inactive, 202 "Accepted" if the
   command has been accepted but the CI/T Command remains active, or 501
   "Not Implemented" if cancellation is not supported by the dCDN.

   If cancellation of a "pending" Trigger Status Resource is accepted by
   the dCDN, the dCDN SHOULD NOT start processing of that activity.
   Issuing a CT/T Cancel Command for a "pending" Trigger Status Resource
   does not however guarantee that the corresponding activity will not
   be started, because the uCDN cannot control the timing of that
   activity. Processing could, for example, start after the POST is
   sent by the uCDN but before that request is processed by the dCDN.

   If cancellation of an "active" or "processed" Trigger Status Resource
   is accepted by the dCDN, the dCDN SHOULD stop processing the CI/T
   Command. However, as with cancellation of a "pending" CI/T Command,
   the dCDN does not guarantee this.

   If the CI/T Command cannot be stopped immediately, the status in the
   corresponding Trigger Status Resource MUST be set to "cancelling",
   and the Trigger Status Resource MUST remain in the collection of
   Trigger Status Resources for active CI/T Commands. If processing is
   stopped before normal completion, the status value in the Trigger
   Status Resource MUST be set to "cancelled", and the Trigger Status

Resource MUST be included in the collection of failed CT/T Trigger
Commands.

Cancellation of a "complete" or "failed" Trigger Status Resource
requires no processing in the dCDN, its status MUST NOT be changed to
"cancelled".

4.4. Deleting Triggers

The uCDN can delete Trigger Status Resources at any time, using the
HTTP DELETE method. The effect is similar to cancellation, but no
Trigger Status Resource remains afterwards.

Once deleted, the references to a Trigger Status Resource MUST be
removed from all Trigger Status Resource collections. Subsequent
requests to GET the deleted Trigger Status Resource SHOULD be
rejected by the dCDN with an HTTP error.

If a "pending" Trigger Status Resource is deleted, the dCDN SHOULD
NOT start processing of that activity. Deleting a "pending" Trigger
Status Resource does not however guarantee that it has not started
because the uCDN cannot control the timing of that activity.
Processing may, for example, start after the DELETE is sent by the
uCDN but before that request is processed by the dCDN.

If an "active" or "processed" Trigger Status Resource is deleted, the
dCDN SHOULD stop processing the CI/T Command. However, as with
deletion of a "pending" Trigger Status Resource, the dCDN does not
guarantee this.

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

4.5. Expiry of Trigger Status Resources

The dCDN can choose to automatically delete Trigger Status Resources
some time after they become "complete", "processed", "failed" or
"cancelled". In this case, the dCDN will remove the Trigger Status
Resource and respond to subsequent requests for it with an HTTP
error.

If the dCDN performs this housekeeping, it MUST have reported the
length of time after which completed Trigger Status Resources will be
deleted via a property of the collection of all Trigger Status
Resources. It is RECOMMENDED that Trigger Status Resources are not
automatically deleted by the dCDN for at least 24 hours after they
become "complete", "processed", "failed" or "cancelled".

To ensure it is able to get the status of its Trigger Status
Resources for completed and failed CI/T Commands, it is RECOMMENDED
that the uCDN polling interval is less than the time after which
records for completed activity will be deleted.

4.6. Loop Detection and Prevention

Given three CDNs, A, B and C. If CDNs B and C delegate delivery of
CDN A's content to each other, CDN A's CI/T Commands could be passed
between CDNs B and C in a loop. More complex networks of CDNs could
contain similar loops involving more hops.

In order to prevent and detect such CI/T loops, each CDN uses a CDN
Provider ID to uniquely identify itself. In every CI/T Command it
originates or cascades, each CDN MUST append an array element
containing its CDN Provider ID to a JSON array under an entry named
"cdn-path". When receiving CI/T Commands a dCDN MUST check the cdn-
path and reject any CI/T Command which already contains its own CDN
Provider ID in the cdn-path. Transit CDNs MUST check the cdn-path
and not cascade the CI/T Command to dCDNs that are already listed in
cdn-path.

The CDN Provider Id consists of the two characters "AS" followed by
the CDN Provider's Autonomous System number, then a colon (':') and
an additional qualifier that is used to guarantee uniqueness in case
a particular AS has multiple independent CDNs deployed. For example
"AS64496:0".

If the CDN provider has multiple Autonomous Systems, the same AS
number SHOULD be used in all messages from that CDN provider, unless
there are multiple distinct CDNs.

If the RI interface described in [I-D.ietf-cdni-redirection] is
implemented by the dCDN, the CI/T and RI interfaces SHOULD use the
same CDN Provider Id.

4.7. Error Handling

A dCDN can signal rejection of a CI/T Command using HTTP status
codes. For example, 400 if the request is malformed, or 403 or 404
if the uCDN does not have permission to issue CI/T Commands or it is
trying to act on another CDN's data.

If any part of the CI/T Trigger Command fails, the trigger SHOULD 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 can be present while the Trigger Status

Resource is still "pending" or "active", if the CI/T Trigger Command is still running for some URLs or Patterns in the Trigger Specification.

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

If a surrogate affected by a CI/T Trigger Command is offline in the dCDN, or the dCDN is unable to pass a CI/T Command on to any of its cascaded dCDNs:

o If the CI/T Command is abandoned by the dCDN, the dCDN SHOULD report an error.

o A CI/T "invalidate" command may be reported as "complete" when surrogates that may have the data are offline. In this case, surrogates MUST NOT use the affected data without first revalidating it when they are back online.

o CI/T "preposition" and "purge" commands can be reported as "processed" if affected caches are offline and the activity will complete when they return to service.

o Otherwise, the dCDN SHOULD keep the Trigger Status Resource in state "pending" or "active" until the CI/T Command is acted upon, or the uCDN chooses to cancel it.

4.8. Content URLs

If content URLs are transformed by an intermediate CDN in a cascade, that intermediate CDN MUST transform URLs in CI/T Commands it passes to its dCDN.

When processing Trigger Specifications, CDNs MUST ignore the URL scheme (http or https) in comparing URLs. For example, for a CI/T invalidate or purge command, content MUST be invalidated or purged regardless of the protocol clients use to request it.

5. CI/T Object Properties and Encoding

CI/T Commands, Trigger Status Resources and Trigger Collections and their properties are encoded using JSON, as defined in sections Section 5.1.1, Section 5.2.1, and Section 5.1.2. They MUST use the MIME Media Type 'application/cdni', with parameter 'ptype' values as defined below and in Section 7.1.

Names in JSON are case sensitive. The names and literal values
specified in the present document MUST always use lower-case.

JSON types, including 'object', 'array', 'number' and 'string' are
defined in [RFC7159].

Unrecognised name/value pairs in JSON objects SHOULD NOT be treated
as an error by either the uCDN or dCDN. They SHOULD be ignored in
the processing, and passed on by dCDN to any further dCDNs in a
cascade.

5.1. CI/T Objects

The top-level objects defined by the CI/T interface are described in
this section.

The encoding of values used by these objects is described in
Section 5.2.

5.1.1. CI/T Commands

CI/T Commands MUST use a MIME Media Type of 'application/cdni;
ptype=ci-trigger-command'.

A CI/T Command is encoded as a JSON object containing the following
name/value pairs.

   Name: trigger

      Description: A specification of the trigger type, and a set of
      data to act upon.

      Value: A Trigger Specification, as defined in Section 5.2.1.

      Mandatory: No, but exactly one of "trigger" or "cancel" MUST be
      present in a CI/T Command.

   Name: cancel

      Description: The URLs of Trigger Status Resources for CI/T
      Trigger Commands that the uCDN wants to cancel.

      Value: A non-empty JSON array of URLs represented as JSON
      strings.

      Mandatory: No, but exactly one of "trigger" or "cancel" MUST be
      present in a CI/T Command.

   Name: cdn-path

      Description: The CDN Provider Identifiers of CDNs that have
      already accepted the CI/T Command.

      Value: A non-empty JSON array of JSON strings, where each
      string is a CDN Provider Identifier as defined in Section 4.6.

      Mandatory: Yes.

5.1.2. Trigger Status Resource

   Trigger Status Resources MUST use a MIME Media Type of 'application/
   cdni; ptype=ci-trigger-status'.

   A Trigger Status Resource is encoded as a JSON object containing the
   following name/value pairs.

   Name: trigger

      Description: The Trigger Specification posted in the body of
      the CI/T Command. 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.

      Value: A Trigger Specification, as defined in Section 5.2.1.

      Mandatory: Yes

   Name: ctime

      Description: Time at which the CI/T Command was received by the
      dCDN. Time is determined by the dCDN, there is no requirement
      to synchronise clocks between interconnected CDNs.

      Value: Absolute Time, as defined in Section 5.2.5.

      Mandatory: Yes

   Name: mtime

      Description: Time at which the Trigger Status Resource was last
      modified. Time is determined by the dCDN, there is no
      requirement to synchronise clocks between interconnected CDNs.

      Value: Absolute Time, as defined in Section 5.2.5.

      Mandatory: Yes

      Name: etime

         Description: Estimate of the time at which the dCDN expects to
         complete the activity. Time is determined by the dCDN, there
         is no requirement to synchronise clocks between interconnected
         CDNs.

         Value: Absolute Time, as defined in Section 5.2.5.

         Mandatory: No

      Name: status

         Description: Current status of the triggered activity.

         Value: Trigger Status, as defined in Section 5.2.3.

         Mandatory: Yes

      Name: errors

         Description: Descriptions of errors that have occurred while
         processing a Trigger Command.

         Value: An array of Error Description, as defined in
         Section 5.2.6. An empty array is allowed, and equivalent to
         omitting "errors" from the object.

         Mandatory: No.

5.1.3. Trigger Collection

   Trigger Collections MUST use a MIME Media Type of 'application/cdni;
   ptype=ci-trigger-collection'.

   A Trigger Collection is encoded as a JSON object containing the
   following name/value pairs.

      Name: triggers

         Description: Links to Trigger Status Resources in the
         collection.

         Value: A JSON array of zero or more URLs, represented as JSON
         strings.

         Mandatory: Yes

Name: staleresourcetime

   Description: The length of time for which the dCDN guarantees
   to keep a completed Trigger Status Resource. After this time,
   the dCDN SHOULD delete the Trigger Status Resource and all
   references to it from collections.

   Value: A JSON number, which must be a positive integer,
   representing time in seconds.

   Mandatory: Yes, in the collection of all Trigger Status
   Resources if the 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.

Names: coll-all, coll-pending, coll-active, coll-complete, coll-
failed

   Description: Link to a Trigger Collection.

   Value: A URL represented as a JSON string.

   Mandatory: Links to all of the filtered collections are
   mandatory in the collection of all Trigger Status Resources, if
   the dCDN implements the filtered collections. Otherwise,
   optional.

Name: cdn-id

   Description: The CDN Provider Identifier of the dCDN.

   Value: A JSON string, the dCDN's CDN Provider Identifier, as
   defined in Section 4.6.

   Mandatory: Only in the collection of all Trigger Status
   Resources, if the dCDN implements the filtered collections.
   Optional in the filtered collections (the uCDN can always find
   the dCDN's cdn-id in the collection of all Trigger Status
   Resources, but the dCDN can choose to repeat that information
   in its implementation of filtered collections).

5.2. Properties of CI/T Objects

   This section defines the values that can appear in the top level
   objects described in Section 5.1, and their encodings.

5.2.1. **Trigger Specification**

   A Trigger Collection is encoded as a JSON object containing the
   following name/value pairs.

   An unrecognised name/value pair in the Trigger Specification object
   contained in a CI/T Command SHOULD be preserved in the Trigger
   Specification of any Trigger Status Resource it creates.

      Name: type

         Description: This property defines the type of the CI/T Trigger
         Command.

         Value: Trigger Type, as defined in Section 5.2.2.

         Mandatory: Yes

      Name: metadata.urls

         Description: The uCDN URLs of the metadata the CI/T Trigger
         Command applies to.

         Value: A JSON array of URLs represented as JSON strings.

         Mandatory: No, but at least one of 'metadata.*' or 'content.*'
         MUST be present and non-empty.

      Name: content.urls

         Description: URLs of content the CI/T Trigger Command applies
         to, see Section 4.8.

         Value: A JSON array of URLs represented as JSON strings.

         Mandatory: No, but at least one of 'metadata.*' or 'content.*'
         MUST be present and non-empty.

      Name: content.ccid

         Description: The Content Collection Identifier of content the
         trigger applies to. The 'ccid' is a grouping of content, as
         defined by [I-D.ietf-cdni-metadata].

         Value: A JSON array of strings, where each string is a Content
         Collection Identifier.

         Mandatory: No, but at least one of 'metadata.*' or 'content.*'
         MUST be present and non-empty.

      Name: metadata.patterns

         Description: The metadata the trigger applies to.

         Value: A JSON array of Pattern Match, as defined in
         Section 5.2.4.

         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.

      Name: content.patterns

         Description: The content data the trigger applies to.

         Value: A JSON array of Pattern Match, as defined in
         Section 5.2.4.

         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.

5.2.2. Trigger Type

   Trigger Type is used in a Trigger Specification to describe trigger
   action. It MUST be one of the JSON strings in the following table:

   +-------------+--+
   | JSON String | Description |
   +-------------+--+
preposition	A request for the dCDN to acquire metadata or
	content.
invalidate	A request for the 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.
purge	A request for the dCDN to erase metadata or
	content. After servicing the request, the specified
	data MUST NOT be held on the dCDN (the dCDN should
	re-acquire the metadata or content from uCDN if it
	needs it).
   +-------------+--+

5.2.3. Trigger Status

   This describes the current status of a Trigger. It MUST be one of
   the JSON strings in the following table:

```
+------------+--------------------------------------------------------+
| JSON       | Description                                            |
| String     |                                                        |
+------------+--------------------------------------------------------+
| pending    | The CI/T Trigger Command has not yet been acted        |
|            | upon.                                                  |
| active     | The CI/T Trigger Command is currently being acted      |
|            | upon.                                                  |
| complete   | The CI/T Trigger Command completed successfully.       |
| processed  | The CI/T Trigger Command has been accepted and no      |
|            | further status update will be made (can be used in     |
|            | cases where completion cannot be confirmed).           |
| failed     | The CI/T Trigger Command could not be completed.       |
| cancelling | Processing of the CI/T Trigger Command is still in     |
|            | progress, but the CI/T Trigger Command has been        |
|            | cancelled by the uCDN.                                 |
| cancelled  | The CI/T Trigger Command was cancelled by the uCDN.    |
+------------+--------------------------------------------------------+
```

5.2.4. PatternMatch

   A Pattern Match consists of a string pattern to match against a URI,
   and flags describing the type of match.

   It is encoded as a JSON object with the following name/value pairs:

      Name: pattern

         Description: A pattern for URI matching.

         Value: A JSON string representing the pattern. 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 "?" MUST be
         escaped as "\\", "*" and "\?".

         Mandatory: Yes.

      Name: case-sensitive

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

      Value: One of the JSON values 'true' (the matching is case-
      sensitive) or 'false' (the matching is case-insensitive).

      Mandatory: No, default is case-insensitive match.

   Name: match-query-string

      Description: Flag indicating whether or not the query part of
      the URI should be included in the pattern match.

      Value: One of the JSON values 'true' (the full URI including
      the query part should be compared against the given pattern),
      or 'false' (the query part of the URI should be dropped before
      comparison with the given pattern).

      Mandatory: No, default is 'false', the query part of the URI
      should be dropped before comparison with the given pattern.

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

   {
       "pattern": "http://www.example.com/trailers/*",
       "case-sensitive": true
   }

5.2.5. Absolute Time

   A JSON number, seconds since the UNIX epoch, 00:00:00 UTC on 1
   January 1970.

5.2.6. Error Description

   An Error Description is used to report failure of a CI/T Command, or
   in the activity it triggered. It is encoded as a JSON object with
   the following name/value pairs:

      Name: error

         Value: Error Code, as defined in Section 5.2.7.

         Mandatory: Yes.

      Names: metadata.urls, content.urls, metadata.patterns,
      content.patterns

         Description: Metadata and content references copied from the
         Trigger Specification. Only those URLs and patterns to which

the error applies are included in each property, but those URLs
and patterns MUST be exactly as they appear in the request, the
dCDN MUST NOT generalise the URLs. (For example, if the uCDN
requests prepositioning of URLs "http://content.example.com/a"
and "http://content.example.com/b", the dCDN must not
generalise its error report to Pattern
"http://content.example.com/*".)

Value: A JSON array of JSON strings, where each string is
copied from a 'content.*' or 'metadata.*' value in the
corresponding Trigger Specification.

Mandatory: At least one of these name/value pairs is mandatory
in each Error Description object.

Name: description

Description: A human-readable description of the error.

Value: A JSON string, the human-readable description.

Mandatory: No.

5.2.7. Error Code

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

```
+------------+----------------------------------------------------------+
| Error Code | Description                                              |
+------------+----------------------------------------------------------+
| emeta      | The dCDN was unable to acquire metadata required to      |
|            | fulfil the request.                                      |
| econtent   | The dCDN was unable to acquire content (CT/T             |
|            | preposition commands only).                              |
| eperm      | The uCDN does not have permission to issue the CI/T      |
|            | Command (for example, the data is owned by another       |
|            | CDN).                                                    |
| ereject    | The dCDN is not willing to fulfil the CI/T Command       |
|            | (for example, a preposition request for content at a     |
|            | time when the dCDN would not accept Request Routing      |
|            | requests from the uCDN).                                 |
| ecdn       | An internal error in the dCDN or one of its              |
|            | downstream CDNs.                                         |
| ecancelled | The uCDN cancelled the request.                          |
+------------+----------------------------------------------------------+
```

6. Examples

   The following sections provide examples of different CI/T objects
   encoded as JSON.

   Discovery of the triggers interface is out of scope of this document.
   In an implementation, all CI/T URLs are under the control of the
   dCDN. The uCDN MUST NOT attempt to ascribe any meaning to individual
   elements of the path.

   In examples in this section, the URL 'http://dcdn.example.com/
   triggers' is used as the location of the collection of all Trigger
   Status Resources, and the CDN Provider Id of uCDN is "AS64496:1".

6.1. Creating Triggers

   Examples of the uCDN triggering activity in the dCDN:

6.1.1. Preposition

   An example of a CI/T preposition command, a POST to the collection of
   all Trigger Status Resources.

   Note that "metadata.patterns" and "content.patterns" are not allowed
   in a preposition Trigger Specification.

   REQUEST:

     POST /triggers HTTP/1.1
     User-Agent: example-user-agent/0.1
     Host: dcdn.example.com
     Accept: */*
     Content-Type: application/cdni; ptype=ci-trigger-command
     Content-Length: 347

     {
       "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"
           ]
       },
       "cdn-path" : ["AS64496:1"]

```
      }

   RESPONSE:

     HTTP/1.1 201 Created
     Date: Sun, 06 Dec 2015 17:18:46 GMT
     Content-Length: 462
     Content-Type: application/cdni; ptype=ci-trigger-status
     Location: http://dcdn.example.com/triggers/0
     Server: example-server/0.1

     {
         "ctime": 1449422326,
         "etime": 1449422334,
         "mtime": 1449422326,
         "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"
         }
     }
```

6.1.2. Invalidate

An example of a CI/T invalidate command, another POST to the
collection of all Trigger Status Resources. This instructs the 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/" using case-insensitive matching,
and "http://www.example.com/a/b/" respectively, using case-sensitive
matching.

REQUEST:

```
     POST /triggers HTTP/1.1
     User-Agent: example-user-agent/0.1
     Host: dcdn.example.com
     Accept: */*
     Content-Type: application/cdni; ptype=ci-trigger-command
     Content-Length: 384
```

```
  {
    "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
          }
        ]
    },
    "cdn-path" : [ "AS64496:1" ]
  }

RESPONSE:

  HTTP/1.1 201 Created
  Date: Sun, 06 Dec 2015 17:18:46 GMT
  Content-Length: 542
  Content-Type: application/cdni; ptype=ci-trigger-status
  Location: http://dcdn.example.com/triggers/1
  Server: example-server/0.1

  {
      "ctime": 1449422326,
      "etime": 1449422334,
      "mtime": 1449422326,
      "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"
```

```
        }
    }
```

6.2. Examining Trigger Status

   Once Trigger Status Resources have been created, the uCDN can check
   their status as shown in these examples.

6.2.1. Collection of All Triggers

   The uCDN can fetch the collection of all Trigger Status Resources it
   has created that 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: 339
   Expires: Sun, 06 Dec 2015 17:19:46 GMT
   Server: example-server/0.1
   ETag: "-8770885545613447380"
   Cache-Control: max-age=60
   Date: Sun, 06 Dec 2015 17:18:46 GMT
   Content-Type: application/cdni; ptype=ci-trigger-collection

   {
       "cdn-id": "AS64496:0",
       "coll-active": "/triggers/active",
       "coll-complete": "/triggers/complete",
       "coll-failed": "/triggers/failed",
       "coll-pending": "/triggers/pending",
       "staleresourcetime": 86400,
       "triggers": [
           "http://dcdn.example.com/triggers/0",
           "http://dcdn.example.com/triggers/1"
       ]
   }
```

6.2.2. Filtered Collections of Trigger Status Resources

The filtered collections are also available to the uCDN. Before the
dCDN starts processing the two CI/T Trigger Commands 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: 150
Expires: Sun, 06 Dec 2015 17:19:46 GMT
Server: example-server/0.1
ETag: "-1475121655268178613"
Cache-Control: max-age=60
Date: Sun, 06 Dec 2015 17:18:46 GMT
Content-Type: application/cdni; ptype=ci-trigger-collection

{
    "staleresourcetime": 86400,
    "triggers": [
        "http://dcdn.example.com/triggers/0",
        "http://dcdn.example.com/triggers/1"
    ]
}
```

At this point, if no other Trigger Status Resources had been created,
the other filtered views 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: 54
Expires: Sun, 06 Dec 2015 17:19:46 GMT
Server: example-server/0.1
ETag: "7958041393922269003"
Cache-Control: max-age=60
Date: Sun, 06 Dec 2015 17:18:46 GMT
Content-Type: application/cdni; ptype=ci-trigger-collection

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

6.2.3. Individual Trigger Status Resources

The Trigger Status Resources can also be examined for detail about
individual CI/T Trigger Commands. For example, for the CI/T
"preposition" and "invalidate" commands 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: 462
Expires: Sun, 06 Dec 2015 17:19:46 GMT
Server: example-server/0.1
ETag: "-256278637448610056"
Cache-Control: max-age=60
Date: Sun, 06 Dec 2015 17:18:46 GMT
Content-Type: application/cdni; ptype=ci-trigger-status

{
    "ctime": 1449422326,
    "etime": 1449422334,
    "mtime": 1449422326,
    "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: 542
     Expires: Sun, 06 Dec 2015 17:19:47 GMT
     Server: example-server/0.1
     ETag: "-1202970338696035175"
     Cache-Control: max-age=60
     Date: Sun, 06 Dec 2015 17:18:47 GMT
     Content-Type: application/cdni; ptype=ci-trigger-status

     {
         "ctime": 1449422326,
         "etime": 1449422334,
         "mtime": 1449422326,
         "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"
         }
     }

6.2.4. Polling for Change

   The uCDN SHOULD use the Entity Tags of collections or Trigger Status
   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: "-1475121655268178613"


RESPONSE:

  HTTP/1.1 304 Not Modified
  Content-Length: 0
  Expires: Sun, 06 Dec 2015 17:19:46 GMT
  Server: example-server/0.1
  ETag: "-1475121655268178613"
  Cache-Control: max-age=60
  Date: Sun, 06 Dec 2015 17:18:46 GMT
  Content-Type: application/cdni; ptype=ci-trigger-collection

REQUEST:

  GET /triggers/0 HTTP/1.1
  User-Agent: example-user-agent/0.1
  Host: dcdn.example.com
  Accept: */*
  If-None-Match: "-256278637448610056"


RESPONSE:

  HTTP/1.1 304 Not Modified
  Content-Length: 0
  Expires: Sun, 06 Dec 2015 17:19:46 GMT
  Server: example-server/0.1
  ETag: "-256278637448610056"
  Cache-Control: max-age=60
  Date: Sun, 06 Dec 2015 17:18:46 GMT
  Content-Type: application/cdni; ptype=ci-trigger-status
```

When the CI/T Trigger Command is complete, the contents of the
filtered collections will be updated along with their Entity Tags.
For example, when the two example CI/T Trigger Commands are complete,
the collections of pending and complete Trigger Status Resources
might 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: 54
Expires: Sun, 06 Dec 2015 17:19:51 GMT
Server: example-server/0.1
ETag: "1337503181677633762"
Cache-Control: max-age=60
Date: Sun, 06 Dec 2015 17:18:51 GMT
Content-Type: application/cdni; ptype=ci-trigger-collection

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

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: 150
Expires: Sun, 06 Dec 2015 17:19:58 GMT
Server: example-server/0.1
ETag: "-2588648306194498266"
Cache-Control: max-age=60
Date: Sun, 06 Dec 2015 17:18:58 GMT
Content-Type: application/cdni; ptype=ci-trigger-collection

{
    "staleresourcetime": 86400,
    "triggers": [
        "http://dcdn.example.com/triggers/0",
        "http://dcdn.example.com/triggers/1"
    ]
}
```

6.2.5. Deleting Trigger Status Resources

The dCDN can delete completed and failed Trigger Status Resources to
reduce the size of the collections. For example, to delete 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: Sun, 06 Dec 2015 17:18:59 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 Trigger
Status Resources 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: 104
Expires: Sun, 06 Dec 2015 17:19:59 GMT
Server: example-server/0.1
ETag: "66479246434429037709"
Cache-Control: max-age=60
Date: Sun, 06 Dec 2015 17:18:59 GMT
Content-Type: application/cdni; ptype=ci-trigger-collection

{
    "staleresourcetime": 86400,
    "triggers": [
        "http://dcdn.example.com/triggers/1"
    ]
}
```

6.2.6. Error Reporting

In this example the uCDN has requested prepositioning of
"http://newsite.example.com/index.html", but the 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: 484
Expires: Sun, 06 Dec 2015 17:20:08 GMT
Server: example-server/0.1
ETag: "8302815253703938792"
Cache-Control: max-age=60
Date: Sun, 06 Dec 2015 17:19:08 GMT
Content-Type: application/cdni; ptype=ci-trigger-status

{
    "ctime": 1449422340,
    "errors": [
        {
            "content.urls": [
                "http://newsite.example.com/index.html"
            ],
            "description": "newsite.example.com not in HostIndex",
            "error": "emeta"
        }
    ],
    "etime": 1449422348,
    "mtime": 1449422344,
    "status": "active",
    "trigger": {
        "content.urls": [
            "http://newsite.example.com/index.html"
        ],
        "type": "preposition"
    }
}
```

7. IANA Considerations

7.1. CDNI Payload Type Parameter Registrations

The IANA is requested to register the following new Payload Types in
the CDNI Payload Type Parameter registry defined by

[I-D.ietf-cdni-media-type], for use with the 'application/cdni' MIME
media type.

RFC Editor Note: Please replace references to [RFCthis] below with
this document's RFC number before publication.

```
+-----------------------+---------------+
| Payload Type          | Specification |
+-----------------------+---------------+
| ci-trigger-command    | [RFCthis]     |
| ci-trigger-status     | [RFCthis]     |
| ci-trigger-collection | [RFCthis]     |
+-----------------------+---------------+
```

8. Security Considerations

The CI/T interface provides a mechanism to allow a uCDN to generate
requests into the dCDN and to inspect its own CI/T requests and their
current state. The CI/T interface does not allow access to or
modification of the uCDN or dCDN metadata relating to content
delivery, or to the content itself. It can only control the presence
of that metadata in the dCDN, and the processing work and network
utilisation involved in ensuring that presence.

By examining pre-positioning requests to a dCDN, and correctly
interpreting content and metadata URLs, an attacker could learn the
uCDN or content owner's predictions for future content popularity.
By examining invalidate or purge requests, an attacker could learn
about changes in the content owner's catalogue.

By injecting CI/T commands an attacker, or a misbehaving uCDN, would
generate work in the dCDN and uCDN as they process those requests.
And so would a man in the middle attacker modifying valid CI/T
commands generated by the uCDN. In both cases, that would decrease
the dCDN caching efficiency by causing it to unnecessarily acquire or
re-acquire content metadata and/or content.

A dCDN implementation of CI/T MUST restrict the actions of a uCDN to
the data corresponding to that uCDN. Failure to do so would allow
uCDNs to detrimentally affect each other's efficiency by generating
unnecessary acquisition or re-acquisition load.

8.1. Authentication, Authorization, Confidentiality, Integrity Protection

A CI/T implementation MUST support TLS transport for HTTP (https) as
per [RFC2818] and [RFC7230].

The use of TLS for transport of the CI/T interface allows:

o The dCDN and the uCDN to authenticate each other.

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

o The dCDN and the uCDN to authorize each other (to ensure they are
   receiving CI/T Commands from, or reporting status to, an
   authorized CDN).

o CDNI commands and responses to be transmitted with
   confidentiality.

o Protection of the integrity of CDNI commands and responses.

In an environment where any such protection is required, mutually
authenticated encrypted transport MUST be used to ensure
confidentiality of the CI/T information. To that end, TLS MUST be
used by CI/T, including authentication of the remote end.

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

HTTP requests that attempt to access or operate on CI/T data
belonging to another CDN MUST be rejected using, for example, HTTP
"403 Forbidden" or "404 Not Found". This is intended to prevent
unauthorised users from generating unnecessary load in dCDN or uCDN
due to revalidation, reacquisition, or unnecessary acquisition.

Note that in a "diamond" configuration, where one uCDN's content can
be acquired via more than one directly-connected uCDN, it may not be
possible for the dCDN to determine from which uCDN it acquired
content. In this case, the dCDN MUST allow each uCDN from which the
content could have been acquired to act upon that content using CI/T
Commands.

8.2. Denial of Service

This document does not define a specific mechanism to protect against
Denial of Service (DoS) attacks on the CI/T. However, CI/T endpoints
can be protected against DoS attacks through the use of TLS transport
and/or via mechanisms outside the scope of the CI/T interface, such
as firewalling or use of Virtual Private Networks (VPNs).

Depending on the implementation, triggered activity may consume
significant processing and bandwidth in the dCDN. A malicious or
faulty uCDN could use this to generate unnecessary load in the dCDN.
The dCDN should consider mechanisms to avoid overload, for example by

rate-limiting acceptance or processing of CI/T Commands, or batching
up its processing.

8.3. Privacy

The CI/T protocol does not carry any information about individual End
Users of a CDN, there are no privacy concerns for End Users.

The CI/T protocol does carry information which could be considered
commercially sensitive by CDN operators and content owners. The use
of mutually authenticated TLS to establish a secure session for the
transport of CI/T data, as discussed in Section 8.1, provides
confidentiality while the CI/T data is in transit, and prevents
parties other party than the authorised dCDN from gaining access to
that data. The dCDN MUST ensure that it only exposes CI/T data
related to a uCDN to clients it has authenticated as belonging to
that uCDN.

9. Acknowledgements

The authors thank Kevin Ma for his input, and Carsten Bormann for his
review and formalization of the JSON data.

10. References

10.1. Normative References

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

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

[RFC6707]   Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content
            Distribution Network Interconnection (CDNI) Problem
            Statement", RFC 6707, September 2012.

[RFC7159]   Bray, T., "The JavaScript Object Notation (JSON) Data
            Interchange Format", RFC 7159, March 2014.

[RFC7230]   Fielding, R. and J. Reschke, "Hypertext Transfer Protocol
            (HTTP/1.1): Message Syntax and Routing", RFC 7230, June
            2014.

[RFC7231]   Fielding, R. and J. Reschke, "Hypertext Transfer Protocol
            (HTTP/1.1): Semantics and Content", RFC 7231, June 2014.

[RFC7232]   Fielding, R. and J. Reschke, "Hypertext Transfer Protocol
            (HTTP/1.1): Conditional Requests", RFC 7232, June 2014.

   [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre,
              "Recommendations for Secure Use of Transport Layer
              Security (TLS) and Datagram Transport Layer Security
              (DTLS)", BCP 195, RFC 7525, May 2015.

10.2. Informative References

   [I-D.greenbosch-appsawg-cbor-cddl]
              Vigano, C. and H. Birkholz, "CBOR data definition language
              (CDDL): a notational convention to express CBOR data
              structures", draft-greevenbosch-appsawg-cbor-cddl-07 (work
              in progress), October 2015.

   [I-D.ietf-cdni-media-type]
              Ma, K., "CDNI Media Type Registration", draft-ietf-cdni-
              media-type-06 (work in progress), October 2015.

   [I-D.ietf-cdni-metadata]
              Niven-Jenkins, B., Murray, R., Caulfield, M., and K. Ma,
              "CDN Interconnection Metadata", draft-ietf-cdni-
              metadata-12 (work in progress), October 2015.

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

   [RFC7336] Peterson, L., Davie, B., and R. van Brandenburg,
              "Framework for Content Distribution Network
              Interconnection (CDNI)", RFC 7336, August 2014.

   [RFC7337] Leung, K. and Y. Lee, "Content Distribution Network
              Interconnection (CDNI) Requirements", RFC 7337, August
              2014.

Appendix A. Formalization of the JSON Data

   This appendix is non-normative.

   The JSON data described in this document has been formalised using
   CDDL [I-D.greenbosch-appsawg-cbor-cddl] as follows:

```
 CIT-object = CIT-command / Trigger-Status-Resource / Trigger-Collection

 CIT-command ; use media type application/cdni; ptype=ci-trigger-command
 = {
   ? trigger: Triggerspec
   ? cancel: [* URI]
```

```
   cdn-path: [* Cdn-PID]
 }

 Trigger-Status-Resource ; application/cdni; ptype=ci-trigger-status
 = {
   trigger: Triggerspec
   ctime: Absolute-Time
   mtime: Absolute-Time
   ? etime: Absolute-Time
   status: Trigger-Status
   ? errors: [* Error-Description]
 }

 Trigger-Collection ; application/cdni; ptype=ci-trigger-collection
 = {
   triggers: [* URI]
   ? staleresourcetime: int ; time in seconds
   ? coll-all: URI
   ? coll-pending: URI
   ? coll-active: URI
   ? coll-complete: URI
   ? coll-failed: URI
   ? cdn-id: Cdn-PID
 }

 Triggerspec = { ; 5.2.1
   type: Trigger-Type
   ? metadata.urls: [* URI]
   ? content.urls: [* URI]
   ? content.ccid: [* Ccid]
   ? metadata.patterns: [* Pattern-Match]
   ? content.patterns: [* Pattern-Match]
 }

 Trigger-Type = "preposition" / "invalidate" / "purge" ; 5.2.2

 Trigger-Status = "pending" / "active" / "complete" / "processed"
    / "failed" / "cancelling" / "cancelled" ; 5.2.3

 Pattern-Match = { ; 5.2.4
   pattern: tstr
   ? case-sensitive: bool
   ? match-query-string: bool
 }

 Absolute-Time = number ; seconds since UNIX epoch, 5.2.5

 Error-Description = { ; 5.2.6
```

```
      error: Error-Code
      ? metadata.urls: [* URI]
      ? content.urls: [* URI]
      ? metadata.patterns: [* Pattern-Match]
      ? content.patterns: [* Pattern-Match]
      ? description: tstr
   }

   Error-Code = "emeta" / "econtent" / "eperm" / "ereject"
      / "ecdn" / "ecancelled"   ; 5.2.7

   Ccid = tstr ; see I-D.ietf-cdni-metadata

   Cdn-PID = tstr .regexp "AS[0-9]+:[0-9]+"

   URI = tstr
```

Authors' Addresses

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

      Email: rob.murray@alcatel-lucent.com

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

      Email: ben.niven-jenkins@alcatel-lucent.com