Network Working Group
Internet-Draft
Obsoletes: 8007 (if approved)
Intended status: Standards Track
Expires: January 9, 2022

O. Finkelman
Qwilt
S. Mishra
Verizon
N. Sopher
Qwilt

July 8, 2021

Content Delivery Network Interconnection (CDNI) Control Interface / Triggers 2nd Edition
draft-sopher-cdni-triggers-extensions-rfc8007bis-01

Abstract

This document obsoletes RFC8007. This document describes the part of the Content Delivery Network Interconnection (CDNI) 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-position metadata or content or to request that it invalidate or purge metadata or content. The upstream CDN can monitor the status of activity that it has triggered in the downstream CDN.

Table of Contents

1. Introduction

[RFC6707] introduces the problem scope for Content Delivery Network Interconnection (CDNI) and lists the four categories of interfaces that may be used to compose a CDNI solution (Control, Metadata, Request Routing, and 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/T interface; requirements applicable to the CI/T interface are CI-1 to CI-6.

- Section 2 outlines the model for the CI/T interface at a high level.
- Section 3 describes collections of Trigger Status Resources.
- Section 4 defines the web service provided by the downstream CDN.
- Section 5 lists properties of CI/T Commands and Status Resources.
- Section 8 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.

Additionally, the following terms are used throughout this document and are defined as follows:

- HLS - HTTP Live Streaming
- DASH - Dynamic Adaptive Streaming Over HTTP
- MSS - Microsoft Smooth Streaming

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].

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 Commands: 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:

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

- invalidate - used to instruct the dCDN to revalidate specific metadata or content before reusing it.

- 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 allows triggered activity to be tracked. The CI/T interface builds on top of HTTP/1.1 [RFC7230]. References to URL in this document relate to HTTP/HTTPS URIs, as defined in Section 2.7 of [RFC7230].

When the dCDN accepts a CI/T Command, it creates a resource describing the 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 CDNI 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 8.

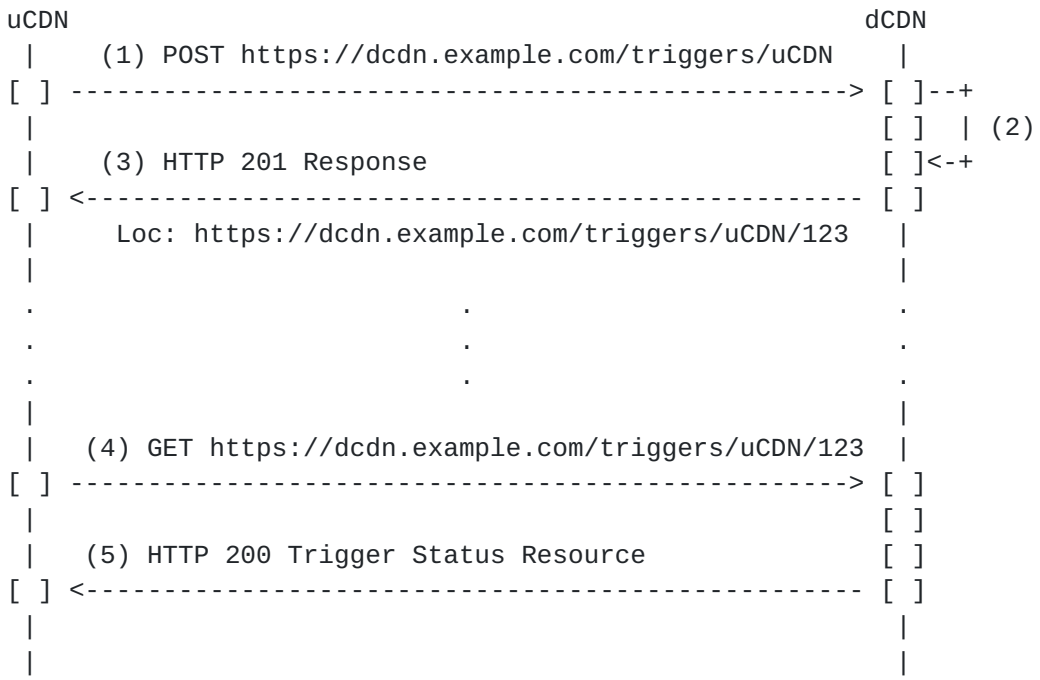


Figure 1: Basic CDNI Message Flow for Triggers

The steps in Figure 1 are as follows:

1. The uCDN triggers action in the dCDN by POSTing a CI/T Command to a collection of Trigger Status Resources -- "https://dcdn.example.com/triggers/uCDN". This URL 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 the 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 and then immediately pre-position replacement content at the same URLs, it SHOULD ensure that the dCDN has completed the invalidate/purge before initiating the pre-positioning. 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.

2.2.1. Multiple Interconnected CDNs

In a network of interconnected CDNs, a single uCDN will originate a given item of metadata and associated content. It may distribute that metadata and content to more than one dCDN, which may in turn distribute that metadata and content to CDNs located further downstream.

An intermediate CDN is a dCDN that passes on CDNI Metadata and content to dCDNs located further downstream.

A "diamond" configuration is one where a dCDN can acquire metadata and content originated in one uCDN from that uCDN itself and an intermediate CDN, or via more than one intermediate CDN.

CI/T Commands originating in the single source uCDN affect metadata and content in all dCDNs; however, in a diamond configuration, it may not be possible for the dCDN to determine which uCDN it acquired content from. In this case, a dCDN MUST allow each uCDN from which it may have acquired the content to act upon that content using CI/T Commands.

In all other cases, a dCDN MUST reject CI/T Commands from a uCDN that attempts to act on another uCDN's content by using, for example, HTTP 403 ("Forbidden").

Security considerations are discussed further in Section 10.

The diamond configuration may lead to inefficient interactions, but the interactions are otherwise harmless. For example:

- When the uCDN issues an "invalidate" CI/T Command, a dCDN will receive that command from multiple directly connected uCDNs. The dCDN may schedule multiple such commands separately, and the last scheduled command may affect content already revalidated following execution of the "invalidate" command that was scheduled first.

- If one of a dCDN's directly connected uCDNs loses its rights to distribute content, it may issue a CI/T "purge" command. That purge may affect content the dCDN could retain because it's distributed by another directly connected uCDN. But, that content can be reacquired by the dCDN from the remaining uCDN.

- When the uCDN originating an item of content issues a CI/T purge followed by a pre-position, two directly connected uCDNs will pass those commands to a dCDN. That dCDN implementation need not merge those operations or notice the repetition, in which case the purge issued by one uCDN will complete before the other. The first uCDN to finish its purge may then forward the "preposition" trigger, and content pre-positioned as a result might be affected by the still-running purge issued by the other uCDN. However, the dCDN will reacquire that content as needed, or when it's asked to pre-position the content by the second uCDN. A dCDN implementation could avoid this interaction by knowing which uCDN it acquired the content from, or it could minimize the consequences by recording

the time at which the "invalidate"/"purge" command was received and not applying it to content acquired after that time.

2.3. Trigger Results

Possible states for a Trigger Status Resource are defined in 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 details, see Section 4.8.

If a dCDN is also acting as a uCDN in a cascade, it MUST forward CI/T Commands to any dCDNs 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 dCDNs. 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 dCDNs. A canceled 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 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 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 MAY also maintain collections representing

filtered views of the collection of all Trigger Status Resources. These filtered collections are "optional-to-implement", but if they are implemented, the dCDN MUST include links to them in the collection of all Trigger Status Resources. The filtered collections are:

- Pending - Trigger Status Resources for CI/T Trigger Commands that have been accepted but not yet acted upon.
- Active - Trigger Status Resources for CI/T Trigger Commands that are currently being processed in the dCDN.
- 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.
- Failed - Trigger Status Resources representing CI/T Commands that failed or were canceled by the uCDN.

4. CDNI Trigger Interface

This section describes an interface to enable a uCDN to trigger activity in a dCDN.

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 [RFC8259]. 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 that 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 [RFC7231]. 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 reuse its URI, even after that Trigger Status Resource has been removed.

The dCDN SHOULD track and report on the progress of CI/T Trigger Commands using a Trigger Status Resource (Section 5.1.2). 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 pre-positioning 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 "preposition" request for data that it has already acquired and that 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 canceled 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 the progress of CI/T Commands it has issued to the dCDN, as described in Sections 4.2.1 and 4.2.2.

To allow the uCDN to check for changes in the status of a Trigger Status Resource or collection of Trigger Status Resources without refetching the whole resource or collection, the dCDN SHOULD include entity-tags (ETags) 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 that 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 the 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 ETags to monitor for change, rather than repeatedly fetching the whole collection. An example of this is given in Section 8.2.4.

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 ETags to monitor for change, rather than repeatedly fetching the Trigger Status Resource.

4.3. Canceling 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 the processing of that activity. Issuing a CI/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 CI/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 the 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 does remove Trigger Status Resources automatically, it MUST report the length of time after which it will do so, using 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 that 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 (PID) to uniquely identify itself. In every CI/T Command it originates or cascades, each CDN MUST append an array element containing its CDN PID 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 that already contains its own CDN PID 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 the cdn-path.

The CDN PID consists of the two characters "AS" followed by the CDN provider's Autonomous System number [RFC1930], 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 ASes, the same AS number SHOULD be used in all messages from that CDN provider, unless there are multiple distinct CDNs.

If the CDNI Request Routing Redirection interface (RI) described in [RFC7975] is implemented by the dCDN, the CI/T interface and the RI SHOULD use the same CDN PID.

4.7. Trigger Extensibility

The CDNI Control Interface / Triggers [RFC8007] defines a set of properties and objects used by the trigger commands. In this document we define an extension mechanism to the triggers interface

that enables the application to add various functions that allow finer control over the trigger execution. This document specifies a generic trigger extension object wrapper for managing individual CDNI trigger extensions in an opaque manner.

This document also registers CDNI Payload Types [RFC7736] under the namespace CIT for the initial set of trigger extension types:

- CIT.LocationPolicy (for controlling the locations in which the trigger is executed)
- CIT.TimePolicy (for scheduling a trigger to run in a specific time window)

Example use cases

- Pre-position with cache location policy
- Purge content with cache location policy
- Pre-position at a specific time
- Purge by content acquisition time (e.g. purge all content acquired in the past X hours)

4.8. Error Handling

A dCDN can signal rejection of a CI/T Command using HTTP status codes -- for example, 400 ("Bad Request") if the request is malformed, or 403 ("Forbidden") or 404 ("Not Found") 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.

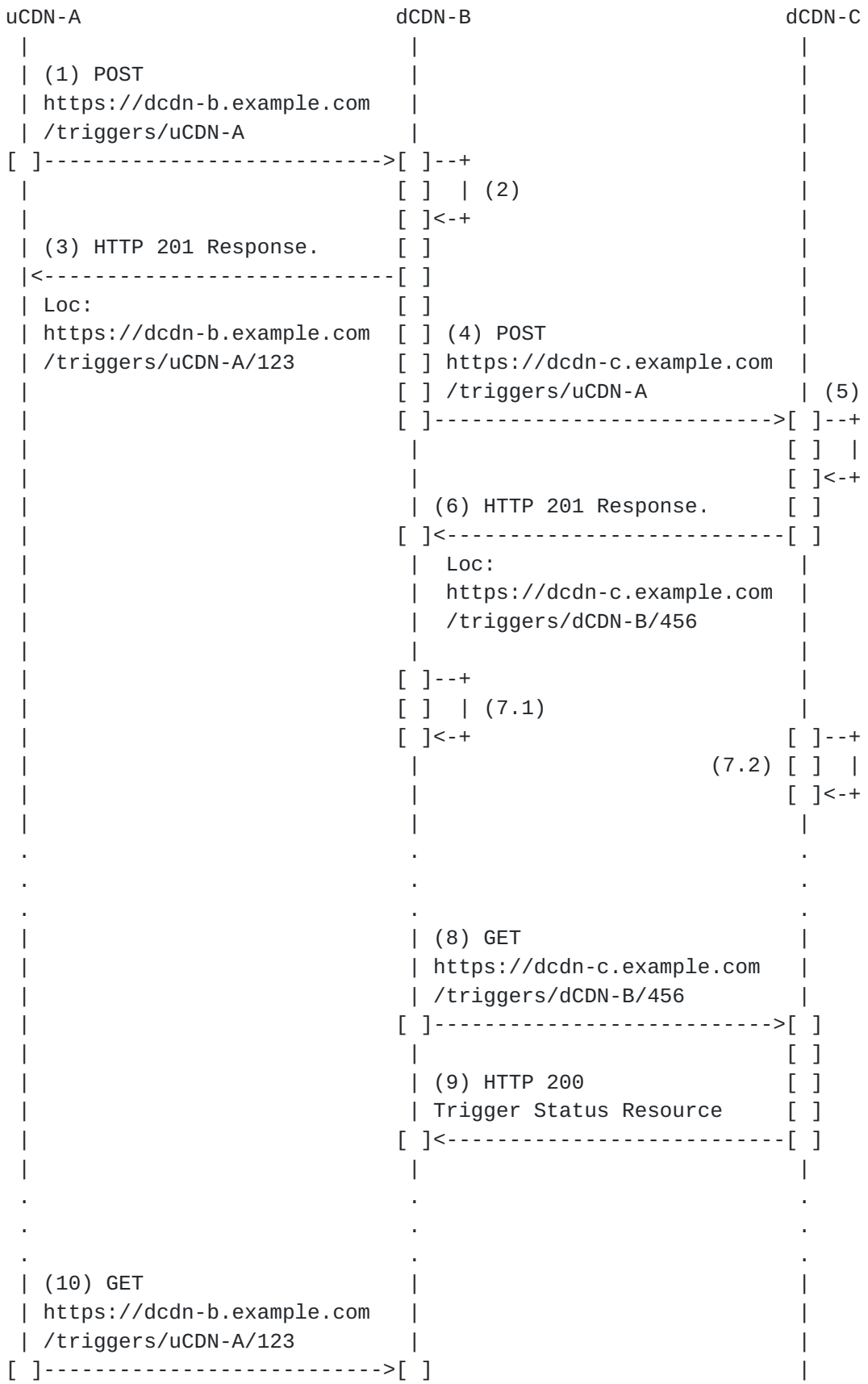
If a Surrogate affected by a CI/T Trigger Command is offline in the dCDN or the dCDN is unable to pass a CI/T Command on to any of its cascaded dCDNs:

- If the CI/T Command is abandoned by the dCDN, the dCDN SHOULD report an error.
- 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.
- 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.
- Otherwise, the dCDN SHOULD keep the Trigger Status Resource in state "pending" or "active" until either the CI/T Command is acted upon or the uCDN chooses to cancel it.

4.8.1. Error propagation

This subsection explains the mechanism for enabling the uCDN to traceback an error to the dCDN in which it occurred. CDNI triggers may be propagated over a chain of downstream CDNs. For example, an upstream CDN A (uCDN-A) that is delegating to a downstream CDN B (dCDN-B) and dCDN-B is delegating to a downstream CDN C (dCDN-C). Triggers sent from uCDN-A to dCDN-B may be redistributed from dCDN-B to dCDN-C and errors can occur anywhere along the path. Therefore, it might be essential for uCDN-A that sets the trigger, to be able to trace back an error to the downstream CDN where it occurred. This document adds a mechanism to propagate the CDN Provider ID (PID) of the dCDN where the fault occured, back to the uCDN by adding the PID to the error description. When dCDN-B propagates a trigger to the further downstream dCDN-C, it MUST also propagate back the errors received in the trigger status resource from dCDN-C by adding them to the errors array in its own status resource to be sent back to the originating uCDN-A. While propagating back the errors, and depending on the implementation, dCDN-B MAY also specify the dCDN-C PID, indicating to which CDN the error relates spefically. The trigger originating upstream CDN will receive an array of errors that occurred in all the CDNs along the execution path, where each error MAY be carrying its own CDN identifier.

Figure 2 below is an example showing the message flow used by uCDN-A to trigger activity in the dCDN-B, followed by dCDN-C, as well as the discovery of the status of that activity, including the Error Propagation.


uCDN-A
dCDN-B
dCDN-C
(1) POST
https://dcdn-b.example.com
/triggers/uCDN-A
(2)
(3) HTTP 201 Response.
Loc:
https://dcdn-b.example.com
/triggers/uCDN-A/123
(4) POST
https://dcdn-c.example.com
/triggers/uCDN-A
(5)
(6) HTTP 201 Response.
Loc:
https://dcdn-c.example.com
/triggers/dCDN-B/456
(7.1)
(7.2)
(8) GET
https://dcdn-c.example.com
/triggers/dCDN-B/456
(9) HTTP 200
Trigger Status Resource
(10) GET
https://dcdn-b.example.com
/triggers/uCDN-A/123

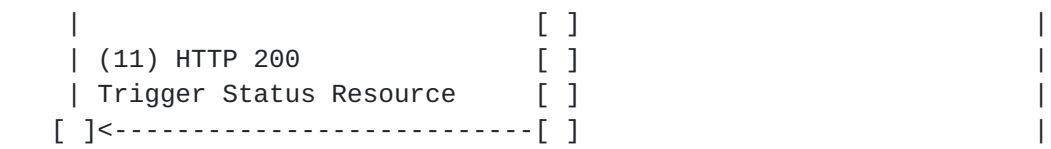


Figure 2: CDNI Message Flow for Triggers, Including Error Propagation

The steps in Figure 2 are as follows:

1. The uCDN-A triggers action in the dCDN-B by POSTing a CI/T Command to a collection of Trigger Status Resources "https://dcdn-b.example.com/triggers/uCDN-A". This URL was given to the uCDN-A when the CI/T interface was established.

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

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

4. The dCDN-B triggers the action in the dCDN-C by POSTing a CI/T Command to a collection of Trigger Status Resources "https://dcdn-c.example.com/triggers/dCDN-B". This URL was given to the uCDN-A when the CI/T interface was established.

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

6. The dCDN-C responds to the dCDN-B with an HTTP 201 response status and the location of the Trigger Status Resource.

7. The dCDN-C acts upon the CI/T Command. However, the command fails at dCDN-C as, for example, the Tigger Specification contains a "type" that is not supported by dCDN-C.

8. The dCDN-B can poll, possibly repeatedly, the Trigger Status Resource in dCDN-C.

9. The dCDN-C responds with the Trigger Status Resource, describing the progress or results of the CI/T Trigger Command. In the described flow, the returned Status is "failed", with an Error Description Object holding an "eunsupported" Error Code reflecting the status response.

10. The uCDN-A can poll, possibly repeatedly, the Trigger Status Resource in dCDN-B.

11. The dCDN-B responds with the Trigger Status Resource, describing the progress or results of the CI/T Trigger Command. In the flow described above, the returned Status is "failed", and the "eunsupported" error received in the trigger status resource from dCDN-C is propagated along with dCDN-C PID by adding it to the errors array in dCDN-B's own status resource to be sent back to the originating uCDN-A.

4.9. Content URLs

If content URLs are transformed by an intermediate CDN in a cascade, that intermediate CDN MUST similarly 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 used to request it.

5. CI/T Object Properties and Encoding

The CI/T Commands, Trigger Status Resources, and Trigger Collections, as well as their properties, are encoded using JSON, as defined in Sections Section 5.1.1, Section 5.1.2, and Section 5.1.3. They MUST use the MIME media type "application/cdni", with parameter "ptype" values as defined below and in Section 9.1.

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

JSON types, including "object", "array", "number", and "string", are defined in [RFC8259].

Unrecognized name/value pairs in JSON objects SHOULD NOT be treated as an error by either the uCDN or dCDN. They SHOULD be ignored during processing and passed on by the 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.v2

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 PIDs of CDNs that have already issued the CI/T Command to their dCDNs.

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

Mandatory: Yes.

5.1.2. Trigger Status Resources

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-serialize 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 synchronize clocks between interconnected CDNs.

Value: Absolute Time, as defined in Section 5.2.9.

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 synchronize clocks between interconnected CDNs.

Value: Absolute Time, as defined in Section 5.2.9.

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 synchronize clocks between interconnected CDNs.

Value: Absolute Time, as defined in Section 5.2.9.

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 Descriptions, as defined in Section 5.2.10. An empty array is allowed and is equivalent to omitting "errors" from the object.

Mandatory: No.

5.1.3. Trigger Collections

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 PID of the dCDN.

Value: A JSON string, the dCDN's CDN PID, 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 unrecognized 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: 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 4.9.

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 [RFC8006].

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 PatternMatch objects, 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 Trigger Type is "preposition".

Name: content.patterns

Description: The content data the trigger applies to.

Value: A JSON array of PatternMatch objects, 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 Trigger Type is "preposition".

Name: content.regexs

Description: Regexs of content URLs to which the CI/T trigger command applies.

Value: A JSON array of RegexMatch objects (see Section 5.2.5).

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 Trigger Type is "preposition".

Name: content.playlists

Description: Playlists of content the CI/T trigger command applies to.

Value: A JSON array of Playlist objects (see Section 5.2.6).

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 Trigger Type is "preposition".

Name: extensions

Description: Array of trigger extension data.

Value:Array of GenericTriggerExtension objects (see Section 5.2.8.2).

Mandatory: No. The default is no extensions.

5.2.2. Trigger Type

Trigger Type is used in a Trigger Specification to describe trigger action.

All trigger types MUST be registered in the IANA "CDNI CI/T Trigger Types" registry (see Section 9.2).

A dCDN receiving a request containing a trigger type it does not recognize or does not support MUST reject the request by creating a Trigger Status Resource with a status of "failed" and the "errors" array containing an Error Description with error "eunsupported".

The following trigger types are defined by this document:

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 revalidating 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 reacquire the metadata or content from the uCDN if it needs it).

5.2.3. Trigger Status

Trigger Status describes the current status of the triggered activity. It MUST be one of the JSON strings in the following table:

JSON String	Description
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.
canceling	Processing of the CI/T Trigger Command is still in progress, but the CI/T Trigger Command has been canceled by the uCDN.
canceled	The CI/T Trigger Command was canceled by the uCDN.

5.2.4. PatternMatch

A PatternMatch 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 can contain the wildcards * and ?, where * matches any sequence of [RFC3986] pchar or "/" characters (including the empty string) and ? matches exactly one [RFC3986] pchar character. The three literals $, * and ? MUST be escaped as $$, $* and $? (where $ is the designated escape character). All other characters are treated as literals.

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 to include the query part of the URI when comparing against the pattern.

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 "https://www.example.com/trailers/":

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

5.2.5. RegexMatch

A RegexMatch consists of a regular expression string a URI is matched against, and flags describing the type of match. It is encoded as a JSON object with following properties:

Property: regex

Description: A regular expression for URI matching.

Type: A regular expression to match against the URI, i.e against the path-absolute and the query string parameters [RFC3986]. The regular expression string MUST be compatible with PCRE [PCRE841].

Note: Because '\' has a special meaning in JSON [RFC8259] as the escape character within JSON strings, the regular expression character '\' MUST be escaped as '\\'.

Mandatory: Yes.

Property: case-sensitive

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

Type: JSON boolean. Either "true" (the matching is case sensitive) or "false" (the matching is case insensitive).

Mandatory: No; default is case-insensitive match (i.e., a value of "false").

Property: match-query-string

Description: Flag indicating whether to include the query part of the URI when comparing against the regex.

Type: JSON boolean. Either "true" (the full URI, including the query part, should be compared against the regex) or "false" (the query part of the URI should be dropped before comparison with the given regex).

Mandatory: No; default is "false". The query part of the URI MUST be dropped before comparison with the given regex. This makes the regular expression simpler and safer for cases in which the query parameters are not relevant for the match.

Example of a case sensitive, no query parameters, regex match against:

```
"^(https:\/\/video\.example\.com)\/([a-z])\/
 movie1\/([1-7])\/*(index.m3u8|\d{3}.ts)$"
```

```
{
  "regex": "^(https:\\/\\/video\\.example\\.com)\\/([a-z])\\/movie1\
          \/([1-7])\\/*(index.m3u8|\\d{3}.ts)$",
  "case-sensitive": true,
  "match-query-string": false
}
```

This regex matches URLs of domain video.example.com where the path structure is /(single lower case letter)/(name-of-title)/(single digit between 1 to 7)/(index.m3u8 or a 3 digit number with ts extension). For example:

https://video.example.com/d/movie1/5/index.m3u8
or
https://video.example.com/k/movie1/4/013.ts

5.2.6. Playlist

A Playlist consists of a full URL and a media protocol identifier. An implementation that supports a specific playlist media protocol MUST be able to parse playlist files of that protocol type and extract, possibly recursively, the URLs to all media objects and/or sub playlist files, and apply the trigger to each one of them separately.

Playlist is encoded as a JSON object with following properties:

Property: playlist

Description: A URL to the playlist file.

Type: A URL represented as a JSON string.

Mandatory: Yes.

Property: media-protocol

Description: Media protocol to be when parsing and interpreting this playlist.

Type: MediaProtocol (see Section 5.2.7).

Mandatory: Yes.

Example of a JSON serialized HLS playlist object:

```
{
  "playlist": "https://www.example.com/hls/title/index.m3u8",
  "media-protocol": "hls"
}
```

5.2.7. MediaProtocol

Media Protocol objects are used to specify registered type of media protocol (see Section 9.4) used for protocol related operations like pre-position according to playlist.

Type: JSON string

Example:

"dash"

5.2.8. CI/T Trigger Extensions

A "trigger.v2" object, as defined in Section 5.2.1 includes an optional array of trigger extension objects. A trigger extension contain properties that are used as directives for dCDN when executing the trigger command -- for example, location policies, time policies and so on. Each such CDNI Trigger extension is a specialization of a CDNI GenericTriggerExtension object. The GenericTriggerExtension object abstracts the basic information required for trigger distribution from the specifics of any given property (i.e., property semantics, enforcement options, etc.). All trigger extensions are optional, and it is thus the responsibility of the extension specification to define a consistent default behavior for the case the extension is not present.

5.2.8.1. Enforcement Options

The trigger enforcement options concept is in accordance with the metadata enforcement options as defined in Section 3.2 of [RFC8006].

The GenericTriggerExtension object defines the properties contained within it as well as whether or not the properties are "mandatory-to-enforce". If the dCDN does not understand or support a mandatory-to-enforce property, the dCDN MUST NOT execute the trigger command. If the extension is not mandatory-to-enforce, then that GenericTriggerExtension object can be safely ignored and the trigger command can be processed in accordance with the rest of the CDNI Trigger spec.

Although, a CDN MUST NOT execute a trigger command if a mandatory-to-enforce extension cannot be enforced, it could still be safe to redistribute that trigger (the "safe-to-redistribute" property) to another CDN without modification. For example, in the cascaded CDN case, a transit CDN (tCDN) could convey mandatory-to-enforce trigger extension to a dCDN. For a trigger extension that does not require customization or translation (i.e., trigger extension that is safe-to-redistribute), the data representation received off the wire MAY be stored and redistributed without being understood or supported by the tCDN. However, for trigger extension that requires translation, transparent redistribution of the uCDN trigger values might not be appropriate. Certain triggers extensions can be safely, though perhaps not optimally, redistributed unmodified. For example, pre-position command might be executed in suboptimal times for some geographies if transparently redistributed, but it might still work.

Redistribution safety MUST be specified for each GenericTriggerExtension property. If a CDN does not understand or support a given GenericTriggerExtension property that is not safe-to-redistribute, the CDN MUST set the "incomprehensible" flag to true for that GenericTriggerExtension object before redistributing it. The "incomprehensible" flag signals to a dCDN that trigger metadata was not properly transformed by the tCDN. A CDN MUST NOT attempt to execute a trigger with an extension that has been marked as "incomprehensible" by a uCDN.

tCDNs MUST NOT change the value of mandatory-to-enforce or safe-to-redistribute when propagating a trigger to a dCDN. Although a tCDN can set the value of "incomprehensible" to true, a tCDN MUST NOT change the value of "incomprehensible" from true to false.

Table 1 describes the action to be taken by a tCDN for the different combinations of mandatory-to-enforce ("MtE") and safe-to-redistribute ("StR") properties when the tCDN either does or does not understand the trigger extension object in question:

MtE	StR	Extension object understood by tCDN	Trigger action
False	True	True	Can execute and redistribute.
False	True	False	Can execute and redistribute.
False	False	False	Can execute. MUST set "incomprehensible" to true when redistributing.
False	False	True	Can execute. Can redistribute after transforming the trigger extension (if the CDN knows how to do so safely); otherwise, MUST set "incomprehensible" to true when redistributing.
True	True	True	Can execute and redistribute.
True	True	False	MUST NOT execute but can redistribute..
True	False	True	Can execute. Can redistribute after transforming the trigger extension (if the CDN knows how to do so safely); otherwise, MUST set "incomprehensible" to true when redistributing.
True	False	False	MUST NOT serve. MUST set "incomprehensible" to true when redistributing.

Table 1: Action to be taken by a tCDN for the different combinations of MtE and StR properties

Table 2 describes the action to be taken by a dCDN for the different combinations of mandatory-to-enforce and "incomprehensible" ("Incomp") properties, when the dCDN either does or does not understand the trigger extension object in question:

MtE	Incomp	Extension object understood by dCDN	Trigger action
False	False	True	Can execute.
False	True	True	Can execute but MUST NOT interpret/apply any trigger extension marked as "incomprehensible".
False	False	False	Can execute.
False	True	False	Can execute but MUST NOT interpret/apply any trigger extension marked as "incomprehensible".
True	False	True	Can execute.
True	True	True	MUST NOT execute.
True	False	False	MUST NOT execute.
True	True	False	MUST NOT execute.

Table 2: Action to be taken by a dCDN for the different combinations of MtE and Incomp properties

5.2.8.2. GenericExtensionObject

A GenericTriggerExtension object is a wrapper for managing individual CDNI Trigger extensions in an opaque manner.

Property: generic-trigger-extension-type

Description: Case-insensitive CDNI Trigger extension object type.

Type: String containing the CDNI Payload Type [RFC7736] of the object contained in the generic-trigger-extension-value property (see table in Section 9.1).

Mandatory: Yes.

Property: generic-trigger-extension-value

Description: CDNI Trigger extension object.

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

Mandatory: Yes.

Property: mandatory-to-enforce

Description: Flag identifying whether or not the enforcement of this trigger extension is mandatory.

Type: Boolean

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

Property: safe-to-redistribute

Description: Flag identifying whether or not this trigger extension can be safely redistributed without modification, even if the CDN fails to understand the extension.

Type: Boolean

Mandatory: No. Default is to allow transparent redistribution (i.e., a value of True).

Property: incomprehensible

Description: Flag identifying whether or not any CDN in the chain of delegation has failed to understand and/or failed to properly transform this trigger extension object. Note: This flag only applies to trigger extension objects whose safe-to-redistribute property has a value of False.

Type: Boolean

Mandatory: No. Default is comprehensible (i.e., a value of False).

Example of a JSON serialized GenericTriggerExtension object containing a specific trigger extension object:

```
{
  "generic-trigger-extension-type":
     <Type of this trigger extension object>,
  "generic-trigger-extension-value":
      {
        <properties of this trigger extension object>
      },
  "mandatory-to-enforce": true,
  "safe-to-redistribute": true,
  "incomprehensible": false
}
```

5.2.9. Absolute Time

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

5.2.10. Error Description

An Error Description is used to report the failure of a CI/T Command or failure 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 5.2.11.

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 generalize the URLs. (For example, if the uCDN requests pre-positioning of URLs "https://content.example.com/a" and "https://content.example.com/b", the dCDN must not generalize its error report to the pattern "https://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.

Name: content.regexs, content.playlists

Description: Content Regex and Playlist references copied from the Trigger Specification. Only those regexs and playlists to which the error applies are included in each property, but those references MUST be exactly as they appear in the request; the dCDN MUST NOT change or generalize the URLs or Regexs. Note that these properties are added on top of the already existing properties: "metadata.urls", "content.urls", "metadata.patterns" and "content.patterns".

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

Mandatory: At least one of "content.regexs", "content.playlists", "metadata.urls", "content.urls", "metadata.patterns" or "content.patterns" is mandatory in each Error Description object.

Name: extensions

Description: Array of trigger extension objects copied from the corresponding "extensions" array from the Trigger Specification. Only those extensions to which the error applies are included, but those extensions MUST be exactly as they appear in the request.

Value: Array of GenericTriggerExtension objects, where each extension object is copied from the "extensions" array values in the Trigger Specification.

Mandatory: No. The "extensions" array SHOULD be used only if the error relates to extension objects.

Name: cdn

Description: The CDN PID of the CDN where the error occurred. The "cdn" property is used by the originating uCDN or by propagating dCDN in order to distinguish in which CDN the error occured.

Value: A non-empty JSON string, where the string is a CDN PID as defined in Section Section 4.6

Mandatory: Yes. In the case the dCDN does not like to expose this information, it should provide its own CDN PID.

Example of a JSON serialized Error Description object reporting a malformed Playlist:

```
{
   "content.playlists": [
      {
        "playlist": "https://www.example.com/hls/title/index.m3u8",
        "media-protocol": "hls"
      }
    ],
   "description": "Failed to parse HLS playlist",
   "error": "econtent",
   "cdn": "AS64500:0"
},
```

Example of a JSON serialized Error Description object reporting an unsupported extension object:

```
{
  "errors.v2": [
   {
     "extensions": [
       {
        "generic-trigger-extension-type":
          <Type of this erroneous trigger extension object>,
        "generic-trigger-extension-value":
          {
           <properties of this erroneous trigger extension object>
          },
       }
     ],
     "description": "unrecognized extension <type>",
     "error": "eextension",
     "cdn": "AS64500:0"
   },
  ]
}
```

5.2.11. Error Code

This type is used by the dCDN to report failures in trigger processing. All Error Codes MUST be registered in the IANA "CDNI CI/T Error Codes" registry (see Section 9.3). Unknown Error Codes MUST be treated as fatal errors, and the request MUST NOT be automatically retried without modification.

The following Error Codes are defined by this document and MUST be supported by an implementation of the CI/T interface.

Error Code	Description
emeta	The dCDN was unable to acquire metadata required to fulfill the request.
econtent	The dCDN was unable to acquire content (CI/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 fulfill 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 dCDNs.
ecanceled	The uCDN canceled the request.
eunsupported	The Trigger Specification contained a "type" that is not supported by the dCDN. No action was taken by the dCDN other than to create a Trigger Status Resource in state "failed".
eextension	An error occurred while parsing a generic trigger extension, or that the specific extension is not supported by the CDN. The Trigger Specification contained a "type" that.

6. Trigger Extension Objects

The objects defined below are intended to be used in the GenericTriggerExtension object's generic-trigger-extension-value field as defined in Section Section 5.2.8.2, and their generic-trigger-extension-type property MUST be set to the appropriate CDNI Payload Type as defined in Section 9.1 .

6.1. LocationPolicy extension

A content operation may be relevant for a specific geographical region, or need to be excluded from a specific region. In this case, the trigger should be applied only to parts of the network that are either "included" or "not excluded" by the location policy. Note that the restrictions here are on the cache location rather than the client location.

The LocationPolicy object defines which CDN or cache locations for which the trigger command is relevant.

Example use cases:

- Pre-position: Certain contracts allow for pre-positioning or availability of contract in all regions except for certain excluded regions in the world, including caches. For example, some content cannot ever knowingly touch servers in a specific country, including cached content. Therefore, these regions MUST be excluded from a pre-positioning operation.

- Purge: In certain cases, content may have been located on servers in regions where the content must not reside. In such cases, a purge operation to remove content specifically from that region, is required.

Object specification

Property: locations

Description: An Access List that allows or denies (blocks) the trigger execution per cache location.

Type: Array of LocationRule objects (see Section 4.2.2.1 of [RFC8006])

Mandatory: Yes.

If a location policy object is not listed within the trigger command, the default behavior is to execute the trigger in all available caches and locations of the dCDN.

The trigger command is allowed, or denied, for a specific cache location according to the action of the first location whose footprint matches against that cache's location. If two or more footprints overlap, the first footprint that matches against the cache's location determines the action a CDN MUST take. If the "locations" property is an empty list or if none of the listed footprints match the location of a specific cache location, then the result is equivalent to a "deny" action.

The following is an example of a JSON serialized generic trigger extension object containing a location policy object that allows the trigger execution in the US but blocks its execution in Canada:

```
{
   "generic-trigger-extension-type": "CIT.LocationPolicy",
   "generic-trigger-extension-value":
    {
       "locations": [
         {
           "action": "allow",
           "footprints": [
             {
               "footprint-type": "countrycode",
               "footprint-value": ["us"]
             }
           ]
         },
         {
           "action": "deny",
           "footprints": [
             {
               "footprint-type": "countrycode",
               "footprint-value": ["ca"]
             }
           ]
         }
       ]
    },
   "mandatory-to-enforce": true,
   "safe-to-redistribute": true,
   "incomprehensible": false
}
```

6.2. TimePolicy Extension

A uCDN may wish to perform content management operations on the dCDN in a specific schedule. The TimePolicy extensions allows the uCDN to instruct the dCDN to execute the trigger command in a desired time window. For example, a content provider that wishes to pre-populate a new episode at off-peak time so that it would be ready on caches at prime time when the episode is released for viewing. A scheduled operation enables the uCDN to direct the dCDN in what time frame to execute the trigger.

A uCDN may wish to to schedule a trigger such that the dCDN will execute it in local time, as it is measured in each region. For example, a uCDN may wish the dCDN to pull the content at off-peak hours, between 2AM-4AM, however, as a CDN is distributed across multiple time zones, the UTC definition of 2AM depends on the actual location.

We define two alternatives for localized scheduling:

- Regional schedule: When used in conjunction with the Location Policy defined in Section 6.1, the uCDN can trigger separate commands for different geographical regions, for each region using a different schedule. This allows the uCDN to control the execution time per region.

- Local Time schedule: We introduce a "local time" version for Internet timestamps that follows the notation for local time as defined in Section 4.2.2 of [ISO8601]. When local time is used, that dCDN SHOULD execute the triggers at different absolute times, according the local time of each execution location.

Object specification

Property: unix-time-window

Description: A UNIX epoch time window in which the trigger SHOULD be executed.

Type: TimeWindow object using UNIX epoch timestamps (see Section 4.2.3.2 of [RFC8006])

Mandatory: No, but exactly one of "unixEpochWindow", "utcWindow" or "localTimeWindow" MUST be present.

Property: utc-window

Description: A UTC time window in which the trigger SHOULD be executed.

Type: UTCWindow object as defined in Section 6.2.1.

Mandatory: No, but exactly one of "unixEpochWindow", "utcWindow" or "localTimeWindow" MUST be present.

Property: local-time-window

Description: A local time window. The dCDN SHOULD execute the trigger at the defined time frame, interpreted as the the local time per location.

Type: LocalTimeWindow object as defined in Section 6.2.2.

Mandatory: No, but exactly one of "unixEpochWindow", "utcWindow" or "localTimeWindow" MUST be present.

If a time policy object is not listed within the trigger command, the default behavior is to execute the trigger in a time frame most suitable to the dCDN taking under consideration other constrains and / or obligations.

Example of a JSON serialized generic trigger extension object containing a time policy object that schedules the trigger execution to a window between 09:00 01/01/2000 UTC and 17:00 01/01/2000 UTC, using the "unix-time-window" property:

```
{
   "generic-trigger-extension-type": "CIT.TimePolicy",
   "generic-trigger-extension-value":
    {
      "unix-time-window": {
         "start": 946717200,
         "end": 946746000
      }
    }
   "mandatory-to-enforce": true,
   "safe-to-redistribute": true,
   "incomprehensible": false
}
```

6.2.1. UTCWindow

A UTCWindow object describes a time range in UTC or UTC and a zone offset that can be applied by a TimePolicy.

Property: start

Description: The start time of the window.

Type: Internet date and time as defined in [RFC3339].

Mandatory: No, but at least one of "start" or "end" MUST be present and non-empty.

Property: end

Description: The end time of the window.

Type: Internet date and time as defined in [RFC3339].

Mandatory: No, but at least one of "start" or "end" MUST be present and non-empty.

Example JSON serialized UTCWindow object that describes a time window from 02:30 01/01/2000 UTC to 04:30 01/01/2000 UTC:

```
{
  "start": 2000-01-01T02:30:00.00Z,
  "end": 2000-01-01T04:30:00.00Z,
}
```

Example JSON serialized UTCWindow object that describes a time window in New York time zone offset UTC-05:00 from 02:30 01/01/2000 to 04:30 01/01/2000:

```
{
  "start": 2000-01-01T02:30:00.00-05:00,
  "end": 2000-01-01T04:30:00.00-05:00,
}
```

6.2.2. LocalTimeWindow

A LocalTimeWindow object describes a time range in local time. The reader of this object MUST interpret it as "the local time at the location of execution". For example, if the time window states 2AM to 4AM local time then a dCDN that has presence in both London (UTC) and New York (UTC-05:00) will execute the trigger at 2AM-4AM UTC in London and at 2AM-4AM UTC-05:00 in New York.

Property: start

Description: The start time of the window.

Type: JSON string formatted as DateLocalTime as defined in Section 6.2.3.

Mandatory: No, but at least one of "start" or "end" MUST be present and non-empty.

Property: end

Description: The end time of the window.

Type: JSON string formatted as DateLocalTime as defined in Section 6.2.3.

Mandatory: No, but at least one of "start" or "end" MUST be present and non-empty.

Example JSON serialized LocalTimeWindow object that describes a local time window from 02:30 01/01/2000 to 04:30 01/01/2000.

```
{
  "start": 2000-01-01T02:30:00.00,
  "end": 2000-01-01T04:30:00.00,
}
```

6.2.3. DateLocalTime

DateLocalTime is a timestamp that follows the date and local time notation in Section 4.3.2 of [ISO8601] as a complete date and time extended representation, where the time zone designator is omitted. In addition, for simplicity and as exact accuracy is not an objective in this case, this specification does not support the decimal fractions of seconds, and does not take leap second into consideration.

Type: JSON string using the format "date-local-time" as defined in Section 6.2.3.1.

6.2.3.1. Date and Local Time Format

The Date and Local Time format is specified here using the syntax description notation defined in [ABNF].

```
date-fullyear   = 4DIGIT
date-month      = 2DIGIT  ; 01-12
date-mday       = 2DIGIT  ; 01-28, 01-29, 01-30, 01-31 based on
                          ; month/year
time-hour       = 2DIGIT  ; 00-23
time-minute     = 2DIGIT  ; 00-59
time-second     = 2DIGIT  ; 00-59 leap seconds are not supported

local-time      = time-hour ":" time-minute ":" time-second
full-date       = date-fullyear "-" date-month "-" date-mday
date-local-time = full-date "T" local-time
```

Example time representing 09:00AM on 01/01/2000 local time:

```
2000-01-01T09:00:00.00
```

> NOTE: Per [ABNF] and [ISO8601], the "T" character in this syntax may alternatively be lower case "t". For simplicity, Applications that generate the "date-local-time" format defined here, SHOULD only use the upper case letter "T".

6.2.3.2. Restrictions

The grammar element date-mday represents the day number within the current month. The maximum value varies based on the month and year as follows:

Month Number	Month/Year	Maximum value of date-mday
01	January	31
02	February, normal	28
02	February, leap year	29
03	March	31
04	April	30
05	May	31
06	June	30
07	July	31
08	August	31
09	September	30
10	October	31
11	November	30
12	December	31

See Appendix C of [RFC3339] for a sample C code that determines if a year is a leap year.

The grammar element time-second may have the values 0-59. The value of 60 that is used in [ISO8601] to represent a leap second MUST NOT be used.

Although [ISO8601] permits the hour to be "24", this profile of [ISO8601] only allows values between "00" and "23" for the hour in order to reduce confusion.

7. Footprint and Capabilities

This section covers the FCI objects required for advertisement of the extensions and properties introduced in this document.

7.1. CI/T Playlist Protocol Capability Object

The CI/T Playlist Protocol capability object is used to indicate support for one or more MediaProtocol types listed in Section 9.4 by the playlists property of the "trigger.v2" object.

Property: media-protocols

Description: A list of media protocols.

Type: A list of MediaProtocol (from the CDNI Triggers media protocol types Section 9.4)

Mandatory: No. The default, in case of a missing or an empty list, is none supported.

7.1.1. CI/T Playlist Protocol Capability Object Serialization

The following shows an example of a JSON serialized CI/T Playlist Protocol Capability object serialization for a dCDN that supports "hls" and "dash".

```
{
 "capabilities": [
   {
     "capability-type": "FCI.TriggerPlaylistProtocol",
     "capability-value": {
       "media-protocols": ["hls", "dash"]
     },
     "footprints": [
       <Footprint objects>
     ]
   }
 ]
}
```

7.2. CI/T Trigger Extension Capability Object

The CI/T Generic Extension capability object is used to indicate support for one or more GenericExtensionObject types.

Property: trigger-extension

Description: A list of supported CDNI CI/T GenericExtensionObject types.

Type: List of strings corresponding to entries from the "CDNI Payload Types" registry [RFC7736] that are under the CIT namespace, and that correspond to CDNI CI/T GenericExtensionObject objects.

Mandatory: No. The default, in case of a missing or an empty list, MUST be interpreted as "no GenericExtensionObject types are supported". A non-empty list MUST be interpreted as containing "the only GenericExtensionObject types that are supported".

7.2.1. CI/T Trigger Extension Capability Object Serialization

The following shows an example of a JSON serialized CI/T Trigger Extension Capability object serialization for a dCDN that supports the "CIT.LocationPolicy" and the "CIT.TimePolicy" objects.

```
{
 "capabilities": [
   {
     "capability-type": "FCI.TriggerGenericExtension",
     "capability-value": {
       "trigger-extension": ["CIT.LocationPolicy", "CIT.TimePolicy"]
     },
     "footprints": [
       <Footprint objects>
     ]
   }
 ]
}
```

8. Examples

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

Discovery of the CI/T interface is out of scope for 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 "https://dcdn.example.com/triggers" is used as the location of the collection of all Trigger Status Resources, and the CDN PID of the uCDN is "AS64496:1".

8.1. Creating Triggers

Examples of the uCDN triggering activity in the dCDN:

8.1.1. Preposition

Below is 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 pre-position 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: 352

{
  "trigger": {
    "type": "preposition",

    "metadata.urls": [ "https://metadata.example.com/a/b/c" ],
    "content.urls": [
        "https://www.example.com/a/b/c/1",
        "https://www.example.com/a/b/c/2",
        "https://www.example.com/a/b/c/3",
        "https://www.example.com/a/b/c/4"
      ]
  },
  "cdn-path": [ "AS64496:1" ]
}
```

RESPONSE:

```
HTTP/1.1 201 Created
Date: Wed, 04 May 2016 08:48:10 GMT
Content-Length: 467
Content-Type: application/cdni; ptype=ci-trigger-status
Location: https://dcdn.example.com/triggers/0
Server: example-server/0.1

{
    "ctime": 1462351690,
    "etime": 1462351698,
    "mtime": 1462351690,
    "status": "pending",
    "trigger": {
        "content.urls": [
            "https://www.example.com/a/b/c/1",
            "https://www.example.com/a/b/c/2",
            "https://www.example.com/a/b/c/3",
            "https://www.example.com/a/b/c/4"
        ],
        "metadata.urls": [
            "https://metadata.example.com/a/b/c"
        ],
        "type": "preposition"
    }
```

```
    }
```

8.1.2. Invalidate

Below is an example of a CI/T "invalidate" command -- another POST to the collection of all Trigger Status Resources. This instructs the dCDN to revalidate the content at "https://www.example.com/a/index.html", as well as any metadata and content whose URLs are prefixed by "https://metadata.example.com/a/b/" using case-insensitive matching, and "https://www.example.com/a/b/" using case-sensitive matching, respectively.

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: 387

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

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

      "content.urls": [ "https://www.example.com/a/index.html" ],
      "content.patterns": [
          { "pattern": "https://www.example.com/a/b/*",
            "case-sensitive": true
          }
        ]
    },
    "cdn-path": [ "AS64496:1" ]
  }
```

RESPONSE:

```
  HTTP/1.1 201 Created
  Date: Wed, 04 May 2016 08:48:11 GMT
  Content-Length: 545
  Content-Type: application/cdni; ptype=ci-trigger-status
  Location: https://dcdn.example.com/triggers/1
  Server: example-server/0.1
```

```
    {
        "ctime": 1462351691,
        "etime": 1462351699,
        "mtime": 1462351691,
        "status": "pending",
        "trigger": {
            "content.patterns": [
                {
                    "case-sensitive": true,
                    "pattern": "https://www.example.com/a/b/*"
                }
            ],
            "content.urls": [
                "https://www.example.com/a/index.html"
            ],
            "metadata.patterns": [
                {
                    "pattern": "https://metadata.example.com/a/b/*"
                }
            ],
            "type": "invalidate"
        }
    }
```

8.1.3. Invalidation with Regex

In the following example a CI/T "invalidate" command uses the Regex property to specify the range of content objects for invalidation, the command is rejected by the dCDN due to regex complexity, and an appropriate error is reflected in the status response.

REQUEST:

```
  POST /triggers HTTP/1.1
  User-Agent: example-user-agent/0.1
  Host: triggers.dcdn.example.com
  Accept: */*
  Content-Type: application/cdni; ptype=ci-trigger-command.v2
  {
    "trigger.v2": {
      "type": "invalidate",
      "content.regexs": [
        {
          "regex": "^(https:\\/\\/video\\.example\\.com)\\/
          ([a-z])\\/movie1\\/([1-7])\\/*(index.m3u8|\\d{3}.ts)$",
          "case-sensitive": true,
          "match-query-string": false
        },
```

```
          { <RegexMatch #2> },
          ...
          { <RegexMatch #N> },
        ],
      },
      "cdn-path": [ "AS64496:0" ]
    }
```

RESPONSE:

```
  HTTP/1.1 201 Created
  Date: Wed, 04 May 2016 08:48:10 GMT
  Content-Length: 467
  Content-Type: application/cdni; ptype=ci-trigger-status.v2
  Location: https://triggers.dcdn.example.com/triggers/0
  Server: example-server/0.1

  {
    "errors.v2": [
      {
        "content.regexs": [
          {
            "regex": "^(https:\\/\\/video\\.example\\.com)\\/
            ([a-z])\\/movie1\\/([1-7])\\/*(index.m3u8|\\d{3}.ts)$",
            "case-sensitive": true,
            "match-query-string": false
          },
        ],
        "description": "The dCDN rejected a regex due to complexity",
        "error": "ereject",
        "cdn": "AS64500:0"
      },
    ],
    "ctime": 1462351690,
    "etime": 1462351698,
    "mtime": 1462351690,
    "status": "failed",
    "trigger.v2": { <content of trigger object from the command> }
  }
```

8.1.4. Preposition with Playlists

In the following example a CI/T "preposition" command uses the Playlist property to specify the full media library of a specific content. The command fails due to playlist parse error and an appropriate error is reflected in the status response.

REQUEST:

```
POST /triggers HTTP/1.1
User-Agent: example-user-agent/0.1
Host: triggers.dcdn.example.com
Accept: */*
Content-Type: application/cdni; ptype=ci-trigger-command.v2
{
  "trigger.v2": {
    "type": "preposition",
    "content.playlists": [
     {
      "playlist": "https://www.example.com/hls/title/index.m3u8",
      "media-protocol": "hls"
     },
     { <Playlist #2> },
     ...
     { <Playlist #N> },
    ],
  },
  "cdn-path": [ "AS64496:0" ]
}
```

RESPONSE:

```
HTTP/1.1 201 Created
Date: Wed, 04 May 2016 08:48:10 GMT
Content-Length: 467
Content-Type: application/cdni; ptype=ci-trigger-status.v2
Location: https://triggers.dcdn.example.com/triggers/0
Server: example-server/0.1

{
  "errors.v2": [
    {
     "content.playlists": [
       {
        "playlist": "https://www.example.com/hls/title/index.m3u8",
        "media-protocol": "hls"
       },
      ],
     "description": "The dCDN was not able to parse the playlist",
     "error": "econtent",
     "cdn": "AS64500:0"
    },
  ],
  "ctime": 1462351690,
  "etime": 1462351698,
  "mtime": 1462351690,
  "status": "failed",
```

```
    "trigger.v2": { <content of trigger object from the command> }
  }
```

8.2. Examining Trigger Status

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

8.2.1. 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: 341
Expires: Wed, 04 May 2016 08:49:11 GMT
Server: example-server/0.1
ETag: "-936094426920308378"
Cache-Control: max-age=60
Date: Wed, 04 May 2016 08:48:11 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": [
        "https://dcdn.example.com/triggers/0",
        "https://dcdn.example.com/triggers/1"
    ]
}
```

8.2.2. 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: 152
Expires: Wed, 04 May 2016 08:49:11 GMT
Server: example-server/0.1
ETag: "4331492443626270781"
Cache-Control: max-age=60
Date: Wed, 04 May 2016 08:48:11 GMT
Content-Type: application/cdni; ptype=ci-trigger-collection

{
    "staleresourcetime": 86400,
    "triggers": [
        "https://dcdn.example.com/triggers/0",
        "https://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: Wed, 04 May 2016 08:49:11 GMT
  Server: example-server/0.1
  ETag: "7958041393922269003"
  Cache-Control: max-age=60
  Date: Wed, 04 May 2016 08:48:11 GMT
  Content-Type: application/cdni; ptype=ci-trigger-collection

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

8.2.3. Individual Trigger Status Resources

The Trigger Status Resources can also be examined for details 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: 467
  Expires: Wed, 04 May 2016 08:49:10 GMT
  Server: example-server/0.1
  ETag: "6990548174277557683"
  Cache-Control: max-age=60
  Date: Wed, 04 May 2016 08:48:10 GMT
  Content-Type: application/cdni; ptype=ci-trigger-status

  {
      "ctime": 1462351690,
      "etime": 1462351698,
```

```
        "mtime": 1462351690,
        "status": "pending",
        "trigger": {
            "content.urls": [
                "https://www.example.com/a/b/c/1",
                "https://www.example.com/a/b/c/2",
                "https://www.example.com/a/b/c/3",
                "https://www.example.com/a/b/c/4"
            ],
            "metadata.urls": [
                "https://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: 545
  Expires: Wed, 04 May 2016 08:49:11 GMT
  Server: example-server/0.1
  ETag: "-554385204989405469"
  Cache-Control: max-age=60
  Date: Wed, 04 May 2016 08:48:11 GMT
  Content-Type: application/cdni; ptype=ci-trigger-status
  {
      "ctime": 1462351691,
      "etime": 1462351699,
      "mtime": 1462351691,
      "status": "pending",
      "trigger": {
          "content.patterns": [
              {
                  "case-sensitive": true,
                  "pattern": "https://www.example.com/a/b/*"
              }
          ],
          "content.urls": [
              "https://www.example.com/a/index.html"
          ],
```

```
            "metadata.patterns": [
                {
                    "pattern": "https://metadata.example.com/a/b/*"
                }
            ],
            "type": "invalidate"
        }
    }
```

8.2.4. Polling for Changes in Status

The uCDN SHOULD use the ETags of collections or Trigger Status Resources when polling for changes 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: "4331492443626270781"
```

RESPONSE:

HTTP/1.1 304 Not Modified Content-Length: 0 Expires: Wed, 04 May 2016 08:49:11 GMT Server: example-server/0.1 ETag: "4331492443626270781" Cache-Control: max-age=60 Date: Wed, 04 May 2016 08:48:11 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: "6990548174277557683"
```

RESPONSE:

```
HTTP/1.1 304 Not Modified
Content-Length: 0
Expires: Wed, 04 May 2016 08:49:10 GMT
Server: example-server/0.1
ETag: "6990548174277557683"
Cache-Control: max-age=60
Date: Wed, 04 May 2016 08:48:10 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 ETags. 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: Wed, 04 May 2016 08:49:15 GMT
  Server: example-server/0.1
  ETag: "1337503181677633762"
  Cache-Control: max-age=60
  Date: Wed, 04 May 2016 08:48:15 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: 152
  Expires: Wed, 04 May 2016 08:49:22 GMT
  Server: example-server/0.1
  ETag: "4481489539378529796"
  Cache-Control: max-age=60
  Date: Wed, 04 May 2016 08:48:22 GMT
  Content-Type: application/cdni; ptype=ci-trigger-collection

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

8.2.5. Deleting Trigger Status Resources

The uCDN can delete completed and failed Trigger Status Resources to reduce the size of the collections, as described in Section 4.4. 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: Wed, 04 May 2016 08:48:22 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: 105
Expires: Wed, 04 May 2016 08:49:22 GMT
Server: example-server/0.1
ETag: "-6938620031669085677"
Cache-Control: max-age=60
Date: Wed, 04 May 2016 08:48:22 GMT
Content-Type: application/cdni; ptype=ci-trigger-collection

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

8.2.6. Extensions with Error Propagation

In the following example a CI/T "preposition" command is using two extensions to control the way the trigger is executed. In this example the receiving dCDN identified as "AS64500:0" does not support the first extension in the extensions array. dCDN "AS64500:0" further distributes this trigger to another downstream CDN that is identified as "AS64501:0", which does not support the second extension in the extensions array. The error is propagated from "AS64501:0" to "AS64500:0" and the errors.v2 array reflects both errors.

REQUEST:

```
POST /triggers HTTP/1.1
User-Agent: example-user-agent/0.1
Host: triggers.dcdn.example.com
Accept: */*
Content-Type: application/cdni; ptype=ci-trigger-command.v2
{
  "trigger.v2": {
    "type": "preposition",
    "content.playlists": [
     {
       "playlist": "https://www.example.com/hls/title/index.m3u8",
       "media-protocol": "hls"
     },
    ],
    "extensions": [
        {
          "generic-trigger-extension-type":
             <Type of trigger extension object #1>,
          "generic-trigger-extension-value":
              {
                <properties of trigger extension object #1>
              },
          "mandatory-to-enforce": true,
          "safe-to-redistribute": true,
        },
        {
          "generic-trigger-extension-type":
             <Type of trigger extension object #2>,
          "generic-trigger-extension-value":
              {
                <properties of trigger extension object #2>
              },
          "mandatory-to-enforce": true,
          "safe-to-redistribute": true,
        },
```

```
        ],
      },
      "cdn-path": [ "AS64496:0" ]
    }
```

RESPONSE:

```
  HTTP/1.1 201 Created
  Date: Wed, 04 May 2016 08:48:10 GMT
  Content-Length: 467
  Content-Type: application/cdni; ptype=ci-trigger-status.v2
  Location: https://triggers.dcdn.example.com/triggers/0
  Server: example-server/0.1

  {
     "errors.v2": [
       {
         "extensions": [
            {
              "generic-trigger-extension-type":
                 <Type of trigger extension object #1>,
              "generic-trigger-extension-value":
                  {
                    <properties of trigger extension object #1>
                  },
              "mandatory-to-enforce": true,
              "safe-to-redistribute": true,
            },
          ],
         "description": "unrecognized extension <type>",
         "error": "eextension",
         "cdn": "AS64500:0"
       },
       {
         "extensions": [
            {
              "generic-trigger-extension-type":
                 <Type of trigger extension object #2>,
              "generic-trigger-extension-value":
                  {
                    <properties of trigger extension object #2>
                  },
              "mandatory-to-enforce": true,
              "safe-to-redistribute": true,
            },
          ],
         "description": "unrecognized extension <type>",
         "error": "eextension",
```

```
            "cdn": "AS64501:0"
          },
        ],
        "ctime": 1462351690,
        "etime": 1462351698,
        "mtime": 1462351690,
        "status": "failed",
        "trigger.v2": { <content of trigger object from the command> }
     }
```

9. IANA Considerations

9.1. CDNI Payload Type Parameter Registrations

The IANA is requested to register the following new Payload Types in the "CDNI Payload Types" registry defined by [RFC7736], for use with the "application/cdni" MIME media type.

Payload Type	Specification
ci-trigger-collection	RFCthis
ci-trigger-command.v2	RFCthis
ci-trigger-status.v2	RFCthis
CIT.LocationPolicy	RFCthis
CIT.TimePolicy	RFCthis
FCI.TriggerVersion	RFCthis
FCI.TriggerPlaylistProtocol	RFCthis
FCI.TriggerGenericExtension	RFCthis

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

9.1.1. CDNI ci-trigger-command.v2 Payload Type

Purpose: TBD: The purpose of this payload type is to distinguish version 2 of the CI/T command (and any associated capability advertisement)

Interface: CI/T

Encoding: see Section 5.1.1

9.1.2. CDNI ci-trigger-status.v2 Payload Type

Purpose: TBD: The purpose of this payload type is to distinguish version 2 of the CI/T status resource response (and any associated capability advertisement)

Interface: CI/T

Encoding: see Section 5.1.2

9.1.3. CDNI ci-trigger-command.v2 Payload Type

Purpose: TBD (came from 8007)

Interface: CI/T

Encoding: see Section 5.1.3

9.1.4. CDNI CI/T LocationPolicy Trigger Extension Type

Purpose: The purpose of this Trigger Extension type is to distinguish LocationPolicy CIT Trigger Extension objects.

Interface: CI/T

Encoding: see Section 6.1

9.1.5. CDNI CI/T TimePolicy Trigger Extension Type

Purpose: The purpose of this Trigger Extension type is to distinguish TimePolicy CI/T Trigger Extension objects.

Interface: CI/T

Encoding: see Section 6.2

9.1.6. CDNI FCI CI/T Playlist Protocol Payload Type

Purpose: The purpose of this payload type is to distinguish FCI advertisement objects for CI/T Playlist Protocol objects

Interface: FCI

Encoding: see Section 7.1.1

9.1.7. CDNI FCI CI/T Extension Objects Payload Type

Purpose: The purpose of this payload type is to distinguish FCI advertisement objects for CI/T Extension objects

Interface: FCI

Encoding: see Section 7.2.1

9.2. "CDNI CI/T Trigger Types" Registry

The IANA is requested to create a new "CDNI CI/T Trigger Types" subregistry under the "Content Delivery Network Interconnection (CDNI) Parameters" registry.

Additions to the "CDNI CI/T Trigger Types" registry will be made via the RFC Required policy as defined in [RFC8126].

The initial contents of the "CDNI CI/T Trigger Types" registry comprise the names and descriptions listed in Section 5.2.2 of this document, with this document acting as the specification.

9.3. "CDNI CI/T Error Codes" Registry

The IANA is requested to create a new "CDNI CI/T Error Codes" subregistry under the "Content Delivery Network Interconnection (CDNI) Parameters" registry.

Additions to the "CDNI CI/T Error Codes" registry will be made via the Specification Required policy as defined in [RFC8126]. The Designated Expert will verify that new Error Code registrations do not duplicate existing Error Code definitions (in name or functionality), prevent gratuitous additions to the namespace, and prevent any additions to the namespace that would impair the interoperability of CDNI implementations.

The initial contents of the "CDNI CI/T Error Codes" registry comprise the names and descriptions of the Error Codes listed in Section 5.2.7 of this document, with this document acting as the specification.

9.4. CDNI Media protocol types

The IANA is requested to create a new "CDNI MediaProtocol Types" subregistry in the "Content Delivery Networks Interconnection (CDNI) Parameters" registry. The "CDNI MediaProtocol Types" namespace defines the valid MediaProtocol object values in Section Section 5.2.7, used by the Playlist object. Additions to the MediaProtocol namespace conform to the "Specification Required"

policy as defined in Section 4.6 of [RFC8126], where the specification defines the MediaProtocol Type and the protocol to which it is associated. The designated expert will verify that new protocol definitions do not duplicate existing protocol definitions and prevent gratuitous additions to the namespace.

The following table defines the initial MediaProtocol values corresponding to the HLS, MSS, and DASH protocols:

MediaProtocol Type	Description	Specification	Protocol Specification
hls	HTTP Live Streaming	RFCthis	RFC 8216 [RFC8216]
mss	Microsoft Smooth Streaming	RFCthis	MSS [MSS]
dash	Dynamic Adaptive Streaming over HTTP (MPEG-DASH)	RFCthis	MPEG-DASH [MPEG-DASH]

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

10. 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 states. 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 utilization involved in ensuring that presence.

By examining "preposition" requests to a dCDN, and correctly interpreting content and metadata URLs, an attacker could learn the uCDN's 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 catalog.

By injecting CI/T Commands, an attacker or a misbehaving uCDN would generate work in the dCDN and uCDN as they process those requests. 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's caching efficiency by causing it to unnecessarily acquire or reacquire 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 reacquisition load.

An origin that chooses to delegate its delivery to a CDN is trusting that CDN to deliver content on its behalf; the interconnection of CDNs is an extension of that trust to dCDNs. That trust relationship is a commercial arrangement, outside the scope of the CDNI protocols. So, while a malicious CDN could deliberately generate load on a dCDN using the CI/T interface, the protocol does not otherwise attempt to address malicious behavior between interconnected CDNs.

10.1. Authentication, Authorization, Confidentiality, Integrity Protection

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

TLS MUST be used by the server side (dCDN) and the client side (uCDN) of the CI/T interface, including authentication of the remote end, unless alternate methods are used for ensuring the security of the information in the CI/T interface requests and responses (such as setting up an IPsec tunnel between the two CDNs or using a physically secured internal network between two CDNs that are owned by the same corporate entity).

The use of TLS for transport of the CI/T interface allows the dCDN and the uCDN to authenticate each other using TLS client authentication and TLS server authentication.

Once the dCDN and the uCDN have mutually authenticated each other, TLS allows:

- The dCDN and the uCDN to authorize each other (to ensure that they are receiving CI/T Commands from, or reporting status to, an authorized CDN).
- CDNI commands and responses to be transmitted with confidentiality.
- Protection of the integrity of CDNI commands and responses.

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

The mechanisms for access control are dCDN-specific and are not standardized as part of this CI/T specification.

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 unauthorized users from generating unnecessary load in dCDNs or uCDNs due to revalidation, reacquisition, or unnecessary acquisition.

When deploying a network of interconnected CDNs, the possible inefficiencies related to the diamond configuration discussed in Section 2.2.1 should be considered.

10.2. Denial of Service

This document does not define a specific mechanism to protect against Denial-of-Service (DoS) attacks on the CI/T interface. 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 the 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 by performing batch processing.

10.3. 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 that 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 10.1, provides confidentiality while the CI/T data is in transit and prevents parties other than the authorized 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.

11. References

11.1. Normative References

[ABNF] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC1930] Hawkinson, J. and T. Bates, "Guidelines for creation, selection, and registration of an Autonomous System (AS)", BCP 6, RFC 1930, DOI 10.17487/RFC1930, March 1996, <https://www.rfc-editor.org/info/rfc1930>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7232] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, DOI 10.17487/RFC7232, June 2014, <https://www.rfc-editor.org/info/rfc7232>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <https://www.rfc-editor.org/info/rfc7525>.

[RFC8006] Niven-Jenkins, B., Murray, R., Caulfield, M., and K. Ma, "Content Delivery Network Interconnection (CDNI) Metadata", RFC 8006, DOI 10.17487/RFC8006, December 2016, <https://www.rfc-editor.org/info/rfc8006>.

[RFC8007] Murray, R. and B. Niven-Jenkins, "Content Delivery Network Interconnection (CDNI) Control Interface / Triggers", RFC 8007, DOI 10.17487/RFC8007, December 2016, <https://www.rfc-editor.org/info/rfc8007>.

[RFC8008] Seedorf, J., Peterson, J., Previdi, S., van Brandenburg, R., and K. Ma, "Content Delivery Network Interconnection (CDNI) Request Routing: Footprint and Capabilities Semantics", RFC 8008, DOI 10.17487/RFC8008, December 2016, <https://www.rfc-editor.org/info/rfc8008>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

11.2. Informative References

[I-D.greevenbosch-appsawg-cbor-cddl]
Birkholz, H., Vigano, C., and C. Bormann, "Concise data definition language (CDDL): a notational convention to express CBOR data structures", draft-greevenbosch-appsawg-cbor-cddl-11 (work in progress), July 2017.

[ISO8601] ISO, "Data elements and interchange formats -- Information interchange -- Representation of dates and times", ISO 8601:2004, Edition 3, 12 2004, <https://www.iso.org/standard/40874.html>.

[MPEG-DASH]
ISO, "Information technology -- Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment format", ISO/IEC 23009-1:2014, Edition 2, 05 2014, <https://www.iso.org/standard/65274.html>.

[MSS] Microsoft, "[MS-SSTR]: Smooth Streaming Protocol", Protocol Revision 8.0, September 2017, <https://msdn.microsoft.com/en-us/library/ff469518.aspx>.

[OC-CM] Finkelman, O., Ed., Devabhaktuni, J., and M. Stock, "Open Caching Content Management Operations Specification", November 2017, <https://www.streamingvideoalliance.org/document/open-caching-content-management-operations-specification/>.

[OCWG] Streaming Video Alliance, "Open Caching", <https://www.streamingvideoalliance.org/technical-groups/open-caching/>.

[PCRE841] Hazel, P., "Perl Compatible Regular Expressions", Version 8.41, July 2017, <http://www.pcre.org/>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <https://www.rfc-editor.org/info/rfc2818>.

[RFC6707] Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", RFC 6707, DOI 10.17487/RFC6707, September 2012, <https://www.rfc-editor.org/info/rfc6707>.

[RFC7336] Peterson, L., Davie, B., and R. van Brandenburg, Ed., "Framework for Content Distribution Network Interconnection (CDNI)", RFC 7336, DOI 10.17487/RFC7336, August 2014, <https://www.rfc-editor.org/info/rfc7336>.

[RFC7337] Leung, K., Ed. and Y. Lee, Ed., "Content Distribution Network Interconnection (CDNI) Requirements", RFC 7337, DOI 10.17487/RFC7337, August 2014, <https://www.rfc-editor.org/info/rfc7337>.

[RFC7736] Ma, K., "Content Delivery Network Interconnection (CDNI) Media Type Registration", RFC 7736, DOI 10.17487/RFC7736, December 2015, <https://www.rfc-editor.org/info/rfc7736>.

[RFC7975] Niven-Jenkins, B., Ed. and R. van Brandenburg, Ed., "Request Routing Redirection Interface for Content Delivery Network (CDN) Interconnection", RFC 7975, DOI 10.17487/RFC7975, October 2016, <https://www.rfc-editor.org/info/rfc7975>.

[RFC8216] Pantos, R., Ed. and W. May, "HTTP Live Streaming", RFC 8216, DOI 10.17487/RFC8216, August 2017, <https://www.rfc-editor.org/info/rfc8216>.

[SVA] "Streaming Video Alliance", <https://www.streamingvideoalliance.org>.

Appendix A. Formalization of the JSON Data

This appendix is non-normative.

The JSON data described in this document has been formalized using the CBOR Data Definition Language (CDDL) [I-D.greevenbosch-appsawg-cbor-cddl] (where "CBOR" means "Concise Binary Object Representation"), 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 = { ; see Section 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" ; see Section 5.2.2

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

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

 Absolute-Time = number ; seconds since UNIX epoch (Section 5.2.5)

 Error-Description = { ; see Section 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" / "ecanceled"  ; see Section 5.2.7

 Ccid = tstr ; see RFC 8006

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

 URI = tstr
```

Acknowledgments

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

Authors' Addresses

Ori Finkelman
Qwilt
6, Ha'harash
Hod HaSharon 4524079
Israel

Email: ori.finkelman.ietf@gmail.com

Sanjay Mishra
Verizon
13100 Columbia Pike
Silver Spring, MD 20904
USA

Email: sanjay.mishra@verizon.com

Nir B. Sopher
Qwilt
6, Ha'harash
Hod HaSharon 4524079
Israel

Email: nir@apache.org