Network Working Group
Internet-Draft
Updates: 8007 (if approved)
Intended status: Standards Track
Expires: December 22, 2021

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

June 20, 2021

CDNI Control Triggers Interface Extensions
draft-ietf-cdni-triggers-extensions-09

Abstract

Open Caching architecture is a use case of Content Delivery Network Interconnection (CDNI) in which the commercial Content Delivery Network (CDN) is the upstream CDN (uCDN) and the ISP caching layer serves as the downstream CDN (dCDN). This document defines extensions to the Content Delivery Network Interconnection (CDNI) Control Interface/Triggers defined in RFC 8007. These extensions are derived from requirements raised by Open Caching architecture but are also applicable to CDNI use cases in general.

Requirements Language

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

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

This Internet-Draft will expire on December 22, 2021.

Table of Contents

1. Introduction

The Streaming Video Alliance [SVA] is a global association that works to solve streaming video challenges in an effort to improve end-user experience and adoption. The Open Caching Working Group [OCWG] of the Streaming Video Alliance [SVA] is focused on the delegation of video delivery requests from commerical CDNs to a caching layer at the ISP's network. Open Caching architecture is a specific use case of CDNI where the commercial CDN is the upstream CDN (uCDN) and the ISP caching layer is the downstream CDN (dCDN). The Open Caching Content Management Operations Specification [OC-CM] defines objects and extensions required by Open Caching architecture for granular content management operations. This document adds those extensions to the CDNI Control Interface / Triggers [RFC8007] as required for

Open Caching content management options. This document also specifies a generic extension mechanism to enable adding future functions for controlling the trigger execution>.

The CDNI Metadata Interface is described in [RFC8006].

The CDNI Footprint and Capability Interface is described in [RFC8008].

The CDNI Control Interface / Triggers is described in [RFC8007].

For consistency with other CDNI documents, this document follows the CDNI convention of uCDN (upstream CDN) and dCDN (downstream CDN) as described in [RFC6707] to represent the commercial CDN and ISP caching layer, respectively.

1.1. Terminology

This document reuses the terminology defined in [RFC6707], [RFC7736] [RFC8006], [RFC8007], and [RFC8008].

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

1.2. Structure of this document

The remainder of this document is organized as follows:

- Section 2 gives an overview of the extensions specified in this document.
- Section 3 specifies version 2 of the CDNI Control Interface / Triggers.
- Section 4 specifies an initial set of trigger extension objects.
- Section 5 specifies Footprint and Capability objects for CI/T version and extensions.
- Section 6 list the IANA considerations of this document.

- Section 7 describes the security considerations for the specified properties and extensions.

2. Interfaces Extensions Overview

This document defines extensions for the CDNI Control Interface / Triggers (CI/T) [RFC8007] and defines FCI objects as per the CDNI Footprint and Capabilities Interface [RFC8008].

2.1. CDNI Control Interface / Triggers Extensions

2.1.1. CI/T Objects

This document specifies version 2 of the CI/T commands and objects. In this context the CI/T commands and objects as were specified in [RFC8007] are considered to be version 1.

2.1.2. Trigger Specification

This document specifies version 2 of the Trigger Specification which is an enhancement of the Trigger Specification that includes all properties as defined in Section 5.2.1 of [RFC8007] as well as the additional properties required by the use cases listed below in Section 2.1.3 and Section 2.1.4.

2.1.3. Content Selection

The trigger specification as defined in Section 5.2.1 of [RFC8007] provides means to select content objects by matching a full content URL or patterns with wildcards. This document specifies two additional selection options:

- Regular Expression - Using regex, a uCDN can create more complex rules to select the content objects for the cases of "invalidation" and "purge". For example, purging specific content within a specific directory path.

- Content Playlist - Using video playlist files, a uCDN can trigger an operation that will be applied to a collection of distinct media files in a format that is natural for a streaming video content provider. A playlist may have several formats, specifically HTTP Live Streaming (HLS) *.m3u8 manifest [RFC8216], Microsoft Smooth Streaming (MSS) *.ismc client manifest [MSS], and Dynamic Adaptive Streaming over HTTP (DASH) *.mpd file [ISO/IEC 23009-1:2014] [MPEG-DASH].

2.1.4. 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)

2.1.5. Error Handling

This document extends the CI/T Error Handling (see Section 4.7 of [RFC8007]) to support the following:

- Playlists and Regexs - report errors that happened due to specific playlists and/or regexs.
- Extension errors - report an error that happened due to an extension object.
- Error propagation - enable the uCDN to traceback an error to the dCDN in which it occurred.

2.2. CDNI Footprint and Capabilities Interface Extensions

Extending the trigger mechanism with optional properties requires the ability for the dCDN to advertise which optional properties it supports.

The CDNI Footprint and Capabilities Interface [RFC8008] enables the dCDN to advertise the capabilities it supports across different footprints. This document introduces FCI objects to support the advertisement of these optional properties.

Example use cases

- Trigger types: Advertise which trigger types are supported by the dCDN. CDNI defines three trigger types (purge, invalidate, pre-position), but it does not necessarily mean that all dCDNs support all of them. The uCDN may prefer to work only with dCDN that support what the uCDN needs.

- Content selection rule types: Advertise which selection types are supported. For example, if adding content regex as a means to match on content URLs, not all dCDN would support it. For playlist mapping, advertise which types and versions of protocols are supported, e.g. HLS.vX/DASH.vY/MSS.vX, DASH templates. Note that the version string or schema are protocol specific.

- Trigger extensions: Advertise which trigger extensions object types are supported by the dCDN.

3. CI/T Version 2

[RFC8007] does not define a version number and versioning scheme. We, therefore, designate the interface and objects as defined in Section 5 of [RFC8007] as version 1. The following sections define version 2 of the CI/T objects and their properties as extensions of version 1.

3.1. CI/T Objects V2

Version 2 of the CI/T interface requires the support of the following objects:

- CI/T Commands v2: A trigger command request using the payload type ci-trigger-command.v2. Version 2 MUST only use "trigger.v2" objects as defined in Section 3.3.1, instead of "trigger" objects. All other properties of the trigger command v2 are as defined in Section 5.1.1 of [RFC8007].

- Trigger Status Resource v2: A trigger status resource response using the payload type ci-trigger-status.v2. Version 2 MUST only use "trigger.v2" objects as defined in Section 3.3.1, instead of a "trigger" object, as well as "errors.v2" array as defined in Section 3.3.6, instead of a "errors" array. All other properties of the trigger status v2 are as defined in Section 5.1.2 of [RFC8007]. The errors array "errors.v2" is a list of all errors that occurred in any of the downstream CDNs along the execution path. When a downstream CDN, dCDN-A, propagates a trigger to another downstream CDN, dCDN-B, it MUST also propagate back all errors reported by dCDN-B in the trigger status resource and add them to its own trigger status resource.

- Trigger Collections: The payload type ci-trigger-collection is used with no changes and as defined in 5.1.3 of [RFC8007].

Usage example of version 2 of trigger command

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": { <properties of a trigger.v2 object> },
  "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": [ { <properties of 1st error.v2 object> },
                  ...,
                { <properties of Nth error.v2 object> }
   ],
   "ctime": 1462351690,
   "etime": 1462351698,
   "mtime": 1462351690,
   "status": "pending",
   "trigger.v2": { <properties of a trigger.v2 object> }
}
```

Usage example of version 2 of trigger status for the trigger created in the above trigger command example:

REQUEST:

```
GET /triggers/0 HTTP/1.1
User-Agent: example-user-agent/0.1
Host: triggers.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.v2

{
   "errors.v2": [ { <properties of 1st error.v2 object> },
                  ...,
                { <properties of Nth error.v2 object> }
   ],
   "ctime": 1462351690,
   "etime": 1462351698,
   "mtime": 1462351690,
   "status": "pending",
   "trigger.v2": { <properties of a trigger.v2 object> }
}
```

3.2. Error Handling V2

The CDNI CI/T interface defines a mechanism for error reporting (see Section 4.7 of [RFC8007]) and an Error Description object for reporting errors (see Section 5.2.6 of [RFC8007]). This document specifies version 2 of CI/T error handling in order to support the following:

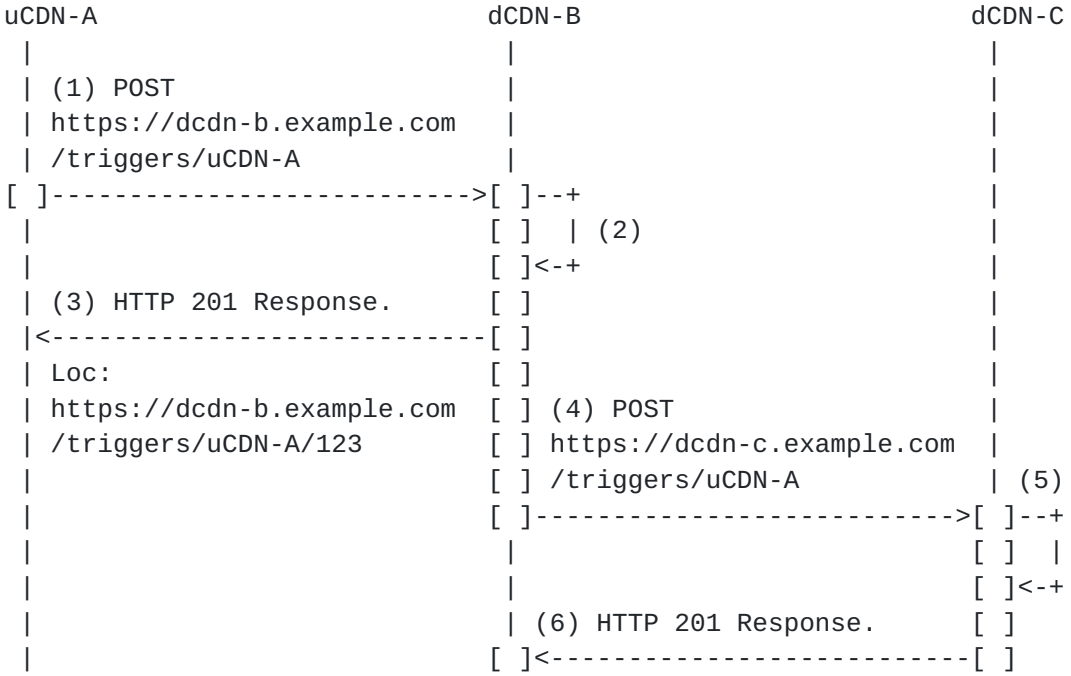
3.2.1. Extension Errors

Report an error that occures due to an extension object. As extension objects are expected to be added to the interface whenever new requirement comes along, it is expected that in some cases a dCDN may receive a trigger that it cannot process or it does not understand. It is therefore essential for the trigger caller to be able to know when such errors occur so they can take actions to fix them. This document adds a mechanism to report extension errors.

3.2.2. 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 1 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.



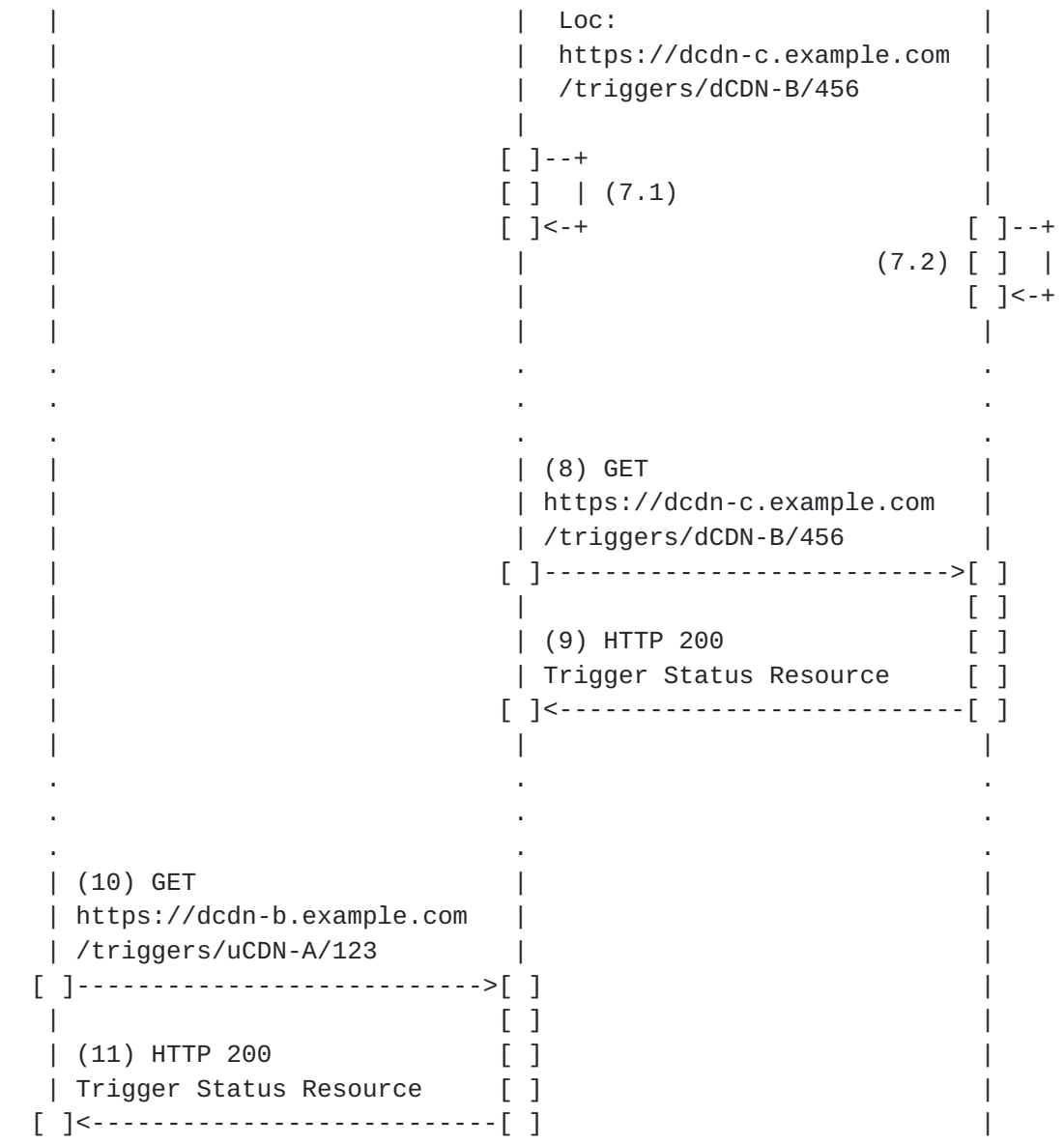


Figure 1: CDNI Message Flow for Triggers, Including Error Propagation

The steps in Figure 1 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.

3.3. Properties of CI/T Version 2 objects

This section defines the values that can appear in the top-level objects described in Section 3.1, and their encodings.

3.3.1. Trigger Specification Version 2

Version 2 of the Trigger Specification adds the following properties on top of the existing properties of the trigger specification defined in Section 5.2.1 of [RFC8007].

Property: content.regexs

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

Type: A JSON array of RegexMatch objects (see Section 3.3.2).

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

Property: content.playlists

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

Type: A JSON array of Playlist objects (see Section 3.3.3).

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

Property: extensions

Description: Array of trigger extension data.

Type: Array of GenericTriggerExtension objects (see Section 3.3.5.2).

Mandatory: No. The default is no extensions.

Example of a JSON serialized invalidation trigger.v2 object with a list of regex objects, a list of playlist objects, and extensions:

```
{
  "trigger.v2": {
    "type": "invalidate",
    "content.regexs": [ <list of RegexMatch objects> ],
    "content.playlists": [ <list of Playlist objects> ],
    "extensions": [ <list of GenericTriggerExtension objects ]
  },
  "cdn-path": [ "AS64496:0" ]
}
```

3.3.2. 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

3.3.3. 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 3.3.4).

Mandatory: Yes.

Example of a JSON serialized HLS playlist object:

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

3.3.4. MediaProtocol

Media Protocol objects are used to specify registered type of media protocol (see Section 6.3) used for protocol related operations like pre-position according to playlist.

Type: JSON string

Example:

"dash"

3.3.5. CI/T Trigger Extensions

A "trigger.v2" object, as defined in Section 3.3.1 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.

3.3.5.1. 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

3.3.5.2. 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 6.1).

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
}
```

3.3.6. Error Description Version 2

Version 2 of the Error Description adds the "content.playlists", "content.regexs", "extensions" and "cdn" properties on top of the existing properties of version 1 of the trigger Error Description as defined in Section 5.2.6 of [RFC8007].

Properties: 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".

Type: 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.

Property: 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.

Type: 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.

Property: 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.

Type: A non-empty JSON string, where the string is a CDN PID as defined in Section 4.6 of [RFC8007].

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"
   },
  ]
}
```

3.3.7. Error codes

This document adds the error code "eextension" to the error codes table defined in Section 5.2.6 of [RFC8007]. This error code designates that an error occurred while parsing a generic trigger extension, or that the specific extension is not supported by the CDN. A CDN that fails to execute a trigger due a generic extension object which is "mandatory-to-enforce" MUST report it using the "errors.v2" array within the trigger status resource, while setting the error code to "eextension" and providing an appropriate description. The "eextension" error code is a registered type of "CDNI CI/T Trigger Error Codes" (see Section 6.2).

3.4. Examples

The following subsections provides usage examples of the specified interface extensions being used by the trigger command and status resource.

3.4.1. 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> }
    }
```

3.4.2. 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> }
    }
```

3.4.3. 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> }
      }
```

4. 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 3.3.5.2, and their generic-trigger-extension-type property MUST be set to the appropriate CDNI Payload Type as defined in Section 6.1 .

4.1. 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
}
```

4.2. 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 4.1, 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 4.2.1.

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 4.2.2.

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
}
```

4.2.1. 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,
}
```

4.2.2. 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 4.2.3.

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 4.2.3.

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,
}
```

4.2.3. 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 4.2.3.1.

4.2.3.1. 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".

4.2.3.2. 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.

5. Footprint and Capabilities

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

5.1. CI/T Versions Capability Object

The CI/T versions capability object is used to indicate support for one or more CI/T objects versions. Note that the default version as originally defined in [RFC8007] MUST be implicitly supported regardless of the versions listed in this capability object.

Property: versions

Description: A list of version numbers.

Type: An array of JSON strings

Mandatory: No. The default is version 1. A missing or an empty versions list means that only version 1 of the interface and objects is supported.

5.1.1. CI/T Versions Capability Object Serialization

The following shows an example of a JSON serialized CI/T Versions Capability object serialization for a dCDN that supports versions 2 and 2.1 of the CI/T interface.

```
{
 "capabilities": [
   {
     "capability-type": "FCI.TriggerVersion",
     "capability-value": {
       "versions": [ "1", "2", "2.1" ]
     },
     "footprints": [
       <Footprint objects>
     ]
   }
 ]
}
```

5.2. 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 6.3 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 6.3)

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

5.2.1. 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>
     ]
   }
 ]
}
```

5.3. 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".

5.3.1. 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>
     ]
   }
 ]
}
```

6. IANA Considerations

6.1. CDNI Payload Types

This document requests the registration of the following CDNI Payload Types under the IANA "CDNI Payload Types" registry defined in [RFC7736]:

Payload Type	Specification
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.]

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

Purpose: 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 3.1

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

Purpose: 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 3.1

6.1.3. 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 4.1

6.1.4. 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 4.2

6.1.5. CDNI FCI CI/T Versions Payload Type

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

Interface: FCI

Encoding: see Section 5.1.1

6.1.6. 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 5.2.1

6.1.7. 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 5.3.1

6.2. CDNI CI/T Trigger Error Codes types

The IANA is requested to update the "CDNI CI/T Error Codes" subregistry (defined in Section 7.3 of [RFC8007] and located at <https://www.iana.org/assignments/cdni-parameters>) with the following registration:

Error Code	Description	Specification
eextension	The dCDN failed to parse a generic "mandatory-to-enforce" extension object, or does not support this extension.	Section Section 3.3.7 of this document.

6.3. 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 3.3.4, 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.]

7. Security Considerations

All security considerations listed in Section 8 of [RFC8007] and Section 7 of [RFC8008] apply to this document as well.

This document defines the capability to use regular expression within the trigger specification for more granular content selection. The usage of regex introduced the risk of regex complexity attacks, a.k.a ReDos attacks. An attacker may be able to craft a regular expression that can exhaust server resources and may take exponential time in the worst case. An implementation MUST protect itself at a minimum by accepting triggers only from an authenticated party over a secured connection. An implementation SHOULD also protect itself by using secure programing techniques and decline trigger commands that use potentially risky regex, such techniques are readily available in secure programming literature and are beyond the scope of this document.

8. Acknowledgments

The authors thank Kevin J. Ma for his guidance as well as careful and methodical reviews and feedback.

9. References

9.1. 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>.

[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>.

[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>.

[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>.

[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>.

9.2. Informative References

[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/>.

[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>.

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