

CDNI Working Group

IETF 119

CI/T Triggers v2

Motivation

- **Open Caching cache management interface**
 - **Core trigger features that cannot be satisfied via extensions**
 - **Interoperability**
- **Conversion of CI/T Triggers v2 into OpenAPI spec**

ObjectList

- CI/T v2 has support for content playlist spec (HLS, MPEG-Dash and HSS) to simplify cache management operations
- **Introduce two additional formats: JSON and plain text**
 - Supports non-video content and/or combination of video manifests with non-video content (thumbnails, subtitles etc.)
 - The JSON format allows for granular file-by-file trigger extension
- Trigger spec by reference creates uncertainty as to how the trigger spec is reduced to a list of objects
- **Allow dCDN to return a list of objects derived from the trigger spec using the new JSON ObjectList format**

Execution Policy: Needs

- In the current CI/T v2 draft dCDN is at liberty how to choose trigger commands for execution, whether to process trigger commands sequentially or in parallel, immediately upon acceptance or with delay in batches
- There exists a need for uCDN to be able to control trigger processing in more detail, including order of execution, dependencies, concurrency
- uCDN also needs better tools to monitor trigger processing status to support such policies

Execution Policy: Priority

uCDN MAY prescribe the order in which dCDN picks up its trigger commands for processing from the “pending” queue, by indicating a relative priority of each trigger

- Urgent trigger (e.g. purge or invalidation)
- Staggering of prepositioning triggers

Execution Policy: Urgency

uCDN MAY request that trigger be accepted for processing immediately without delay

dCDN may deny such trigger due to its internal business logic and/or due to conflict with other trigger attributes (e.g. low priority or dependency)

Execution Policy: Dependency

uCDN MAY request that trigger be processed only after other triggers have been processed in full (reached “complete”, “failed” or “cancelled” state)

- Purge-then-preposition
- Cancel preposition, retry when cancelled
- Chain prepositioning triggers

Label Support

- When managing large number of triggers, uCDN needs better tools for monitoring trigger status than afforded today – group related triggers, monitor trigger attributes related to execution in one API call
- **Introduce labels – at the trigger spec and extension level**
- **Support TriggerCollection filtered by labels, in addition to state-based filters today**
- **Return set of trigger labels (trigger and extension) together with the trigger URL in the TriggerCollection object, exposing extension attributes as specified in ExecutionPolicy**

URL Type Attribute

- When managing cache, uCDN may refer to published URLs or cache key URLs
- The use of published URLs is the default, assumes that dCDN has the relevant configuration metadata objects in place and **MUST** invoke uCDN-provided metadata objects (e.g. origin access, processing stages etc.), when processing the trigger, if such exist
- **Add “url-type” attribute to all URL-based trigger spec types: URLs, URI Pattern Match, URI Regex Match and Content Playlist (ObjectList)**

RESTful API

- The current trigger API is not fully RESTful, completed triggers are deleted using DELETE method, active or pending triggers are cancelled using special CI/T Cancel Command using POST method
- Use of DELETE for both is undesirable even when using HTTP 201 Accepted async response, we want to query cancelled triggers
- **Allow POST method to modify trigger and set “desired-state” attribute, which can be “pending” (don’t process yet), “complete” (process) or “cancelled” (cancel)**
- **Allow modification of trigger spec for pending triggers via POST**
- **Eliminate CI/T Trigger commands, manage REST trigger resources**

Multiple CI/T Interfaces

- Open Caching envisions use of trigger-based APIs in two different interfaces:
 - Configuration for “metadata” trigger subject
 - Cache Management for “content” trigger subject
- Each interface has its own endpoint and can manage its own trigger objects, the point of contention is in FCI advertisement
- **Rename CI/T Objects Versions Capability Object to CI/T Trigger Endpoint object**
 - trigger-endpoint URL (Cache Management or Configuration)
 - trigger version (e.g. v1 for metadata and v2 for content)
 - trigger subject (metadata vs. content)

CDN Path

- OpenCaching today deals with directly attached CDNs only
- CDN cascading supported in an opaque manner: CDNs in a chain are unaware of additional CDNs down or up the chain beyond the peer CDN
- CI/T Triggers v2 mandates use of ASN-based CDN provider ID when creating triggers (“cdn-path”) and reporting trigger status (“cdn-id”)
- ASN-based IDs have become obsolete, upcoming security related work may call for complex CDN identifiers (identity provider URL + identity)
- **Make “cdn-path” attribute in trigger spec and “cdn-id” in Error.v2 Description optional AND**
- **Convert these attributes to opaque data type (JSON objects) to allow for multiple future CDN PID schemes to be slotted in**

CDNI Working Group

IETF 119

Named Footprints

Motivation

- **New dCDN use cases that require advanced footprint capabilities**
 - **Distinct access networks under common dCDN management**
 - **Differentiated CDN layers (edge and “last-mile” cache layers)**
 - **CDN requirements by geography (e.g. GDPR)**
- **These use cases call require**
 - **Footprints to be used in metadata inside and outside of FCI (e.g. in configuration, logging, cache management) in a consistent manner**
 - **Complex footprint definition logic**
 - **Support for dynamically changing footprints**

Initial Scope

- **Extend FCI to advertise referenceable named footprint objects**
 - **Footprints accessible via both common advertisement and individually**
 - **Hierarchical advertisement**
 - **Namespace support**
 - **Client-side caching support**
- **Add two new footprint types**
 - **“named” footprint type references the FCI advertisement**
 - **“expr” footprint uses CDNI MEL expressions to define a footprint**
- **Change complex footprint types to specify an optional datasource (“asn”, “country” and “subdivisioncode”)**

v01 Changes compared to v00

- **Miscellaneous syntax fixes**
- **MI FootprintSource metadata object**
- **Support for self-published geofeeds via RFC8805**

```
{
  "footprint-source-type": "rfc8805",
  "footprint-source-uri": "http://noc.ietf.org/geo/google.csv"
  "footprint-source-footprint-type": [ "country", "subdivisioncode" ]
}

{
  "footprint-source-type": "private",
  "footprint-source-uri": "https://www.maxmind.com",
  "footprint-source-footprint-type": [ "country", "subdivisioncode" ]
}

{
  "footprint-source-type": "private",
  "footprint-source-uri": "https://tools.iplocation.net/ip-to-asn",
  "footprint-source-footprint-type": [ "asn" ]
}
```

Named Footprints vs ALTO

○ ALTO (Application Layer Traffic Optimization)

- Access network provider to publish its network topology, route availability and associated cost to enhance request routing for application end points on its network (e.g. P2P, multi-CDN client steering)

○ Named Footprints

- dCDN to publish its coverage footprints so that uCDN can associate specific application end points with this dCDN and delegate requests to dCDN
- dCDN doesn't necessarily have authoritative topology and route availability information for all footprints it covers
- uCDN may use ALTO-aware steering to choose between multiple dCDNs advertising coverage for same client

Next Steps

- **FCI Convergence**
- **Specifying footprint-related API, including**
 - **“pull”**: uCDN queries dCDN for footprint updates
 - **“push”**: dCDN informs uCDN of its footprint updates

THANKS!