CDNI Working Group
IETF 118

Cache Management Interface
Motivation

○ Industry-standard interface for CDN cache management
  ▪ Prepositioning, invalidation and purge
  ▪ Automation of manual and semi-manual processes
  ▪ Multi-CDN
  ▪ Span operator CDNs and public CDNs

○ Regional content placement and cache management

○ Performance optimization through cache hints

○ Edge storage management
  ▪ Quota management
  ▪ Catalog replication
Interface Architecture

- **Operations API**
  - CDNI trigger v2 compliant
  - Purge, invalidate, preposition

- **Cache bucket API**
  - Cache bucket lifecycle
  - REST API
○ **Containers of cache objects**
  - Storage allocation per content type and/or geographical footprint
  - Default cache bucket
  - All trigger API operations take place within bucket context
  - “Cache” type (dCDN managed) and “storage” type (uCDN managed)

○ **API support**
  - Separate bucket management API
  - Attach configuration MI objects to cache bucket metadata objects via configuration API
Key Features

○ CI/T v2 API
  ○ Object list
  ○ Preposition and purge policies
  ○ Advanced scheduling
  ○ Extended status and error reporting to support object lists
  ○ Use of content category (content.ccid)

○ Cache buckets: footprint support and more
○ Object tagging (MI GroupingExtended)
○ Object priority (MI CachePriority)
○ Cache hints
○ Re-use of configuration metadata
Proposed CDNI Roadmap

- **Core CI/T v2 draft**
  - Non-manifest object lists
  - Trigger execution priority
  - Published URL vs. cache key URL
  - Status v2: return object list (MI ObjectList)
  - Error v2: etimeout error code and object lists

- **Trigger extension policies**
  - Preposition
  - Scheduling and recurrence via JCAL/iCalendar
  - Purge

- **Cache management**
  - MI CacheBucket objects and API
  - MI Grouping (Extended)
  - MI CachePriority
Motivation

○ New dCDN use cases emerged 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
- CDNI operation changes to include retrieval and periodic refreshment of footprint advertisements
- 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")
Proposed Edits for v01

- Miscellaneous syntax fixes
- Push footprint updates
- Support for self-published geofeeds via RFC8805

"footprint-def": {
    "footprint-type": "countrycode",
    "footprint-source": {
        "footprint-source-type": "rfc8805",
        "footprint-source-uri": "http://noc.ietf.org/geo/google.csv",
    }
    "footprint-value": ["us", "ca"]
}
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