Slide Title

• Abstract
• MI.CrossoriginPolicy
• MI.AllowCompress
• MI.ClientConnectionControl
• MI.TrafficType
Abstract

• Extension to RFC 8006 with new MI.GenericMetadata objects related to controlling edge access to resources via CDNs and Open Caching systems.
• This objects cover functionalities that represent day-by-day operations in configurations between Content Providers and CDNs like:
  – configuring Cross-Origin Resource Sharing (CORS) access rules and the dynamic generation of CORS headers
  – response body compression rules in the edge
  – client connection timeouts
  – Traffic type hints for optimized caching.
• Most of the content included in this draft was included in a previous draft (draft-goldstein-cdni-metadata-model-extensions-02) now expired and splitted as result of previous IETF CDN-I meetings
When delivering video content is typical that Videoservice platforms use different domain realms for video portal that for video delivery (more even if using a third-party CDN).

In such cases CORS mechanism implemented in browsers and devices are in use, making necessary to generate Access-Control-Allow-* headers in the responses according to a predefined logic to authorise those Cross-domain requests.

Without any other mechanism, a cache system or Surrogate acting on behalf of the Content Provider Source should validate those requests based on the Origin header received in the request.

It is a typical use case in current CDNs to generate Synthetic responses in the edge servers, based on logic that permit to implement the policy a Content Provider want to use for CORS responses.

The proposed object is design to permit configuring those policies to a downstream CDN to generate those headers and synthetic responses without needing to ask the CP source.

* [https://www.w3.org/TR/2020/SPSD-cors-20200602/](https://www.w3.org/TR/2020/SPSD-cors-20200602/)
• Simple CORS configuration
  – Only controlling Access-Control-Allow-Origin header in responses
  – Permit the usage of wildcard in the response

• Advanced CORS configuration
  – Managing all the rest of Access-Control-Allow-* headers in responses
  – Generation of synthetic responses for preflight requests (CORS)
MI.AllowCompress is a new GenericMetadata object that allows the dCDN to compress content before sending to the client.

Thus, the uCDN source can send only one copy of one object, being cached in the edge, while the edge server handles the different options based on the Accept-Encoding header the UA is using in the request.

It is expected this object is used along with other proposed MI objects as MI.HeaderTransform to restrict this functionality to specific object types as DASH manifests or HLS playlists, as using compression in the video segments brings no advantage.

```json
{
  "match": {
    "expression": "req.h.uri *= '*.m3u8'"
  },
  "stage-metadata": {
    "generic-metadata": [
      {
        "generic-metadata-type": "MI.AllowCompress",
        "generic-metadata-value": {
          "allow-compress": "true"
        }
      }
    ]
  }
}
```
• MI.ClientConnectionControl is a new GenericMetadata object that specifies how a dCDN manages its connections to clients/players.

• Configuration metadata is required to define how connections against a client are maintained by a dCDN. Since the clients are typically owned/operated by a uCDN, giving this control to the uCDN allows it to accommodate device specific constraints and performance optimizations. A dCDN can also benefit from this configuration metadata to meet its security and resource consumption requirements.

```json
{
  "generic-metadata-type": "MI.ClientConnectionControl",
  "generic-metadata-value": {
    "connection-keep-alive-time-ms": 3
  }
}
```
MI.TrafficType metadata defines a set of descriptors that characterize either the type or usage of the traffic, enabling dCDNs to apply any internal configuration rules without exposing an unnecessary number of internal details.

CDNs in the last years are implementing different strategies to handle VoD and Live content, as the way those are requested, or the concurrency on the cached objects are very different. Thus, a dCDN that knows the type of traffic it is going to deliver for one specific service can activate those strategies that are implementation specific.

Thus, the dCDNs receiving delegated traffic with this information can be more efficient delivering the content and handle more traffic.

Note that the interpretation of these traffic types and application of rules such as rate limiting or delivery pacing are implementation specific.

The property traffic-type can set a specific value (vod, live, object-download), while the property hints permit the uCDN to provide more information that could, or could not, be used or understood by the dCDN, without being a problem to configure the service.

```json
{
   "generic-metadata-type": "MI.TrafficType",
   "generic-metadata-value": {
      "traffic-type": "vod",
      "hints": [ "low-latency", "catch-up" ]
   }
}
```
Next steps

• Please, review the draft and provide any comments to the mailing list or the authors
• Kindly ask for adoption of this draft in the CDN-I WG
Thank you