SVA Configuration Interface

IETF/CDNi Metadata Model Extensions

November 2021 (IETF 112)

streaming video alliance

Background

CDN Configuration Metadata Challenge

- **Problem Statement:** The need for an industry-standard API and configuration metadata model becomes increasingly important as content and service providers automate more of their operations, and as technologies such as open caching require coordination of content delivery configurations.

- **Motivational Drivers:** The SVA Membership (which includes Content Providers, CDNs, ISPs, and Open Caching technology vendors) agreed on the following drivers for defining a standard configuration interface:

    - Make integrations more efficient
    - Enable self-service configuration (DevOps)
    - Standardize CDN feature definitions
    - Improve governance & compliance
    - Create opportunities for differentiation

Wasn't CDNI Metadata Sufficient As-Is?

Quick Answer: Almost, but not quite.

- The CDNi Metadata and Control Interfaces (RFCs 8006 & 8007) were designed for the limited scope use cases related interchanges between upstream CDNs and downstream CDNs.

- As we look at the wider set of use cases involving Content Providers managing multiple CDNS, along with use cases in the Open Caching ecosystem, we see gaps.

What CDNI Metadata Provides	**Gaps - What's Missing**
Simple CDN Metadata Object Model	Metadata Object Model meeting more complex requirements of CDN and Open Caching industries
Interfaces for retrieving metadata, and triggering metadata prepositioning, invalidation and purging	Simple push-style metadata publishing (POST)
Interfaces to check status or cancel trigger requests	Advanced configuration publishing capabilities required by Content Providers (publishing, versioning, deployment)

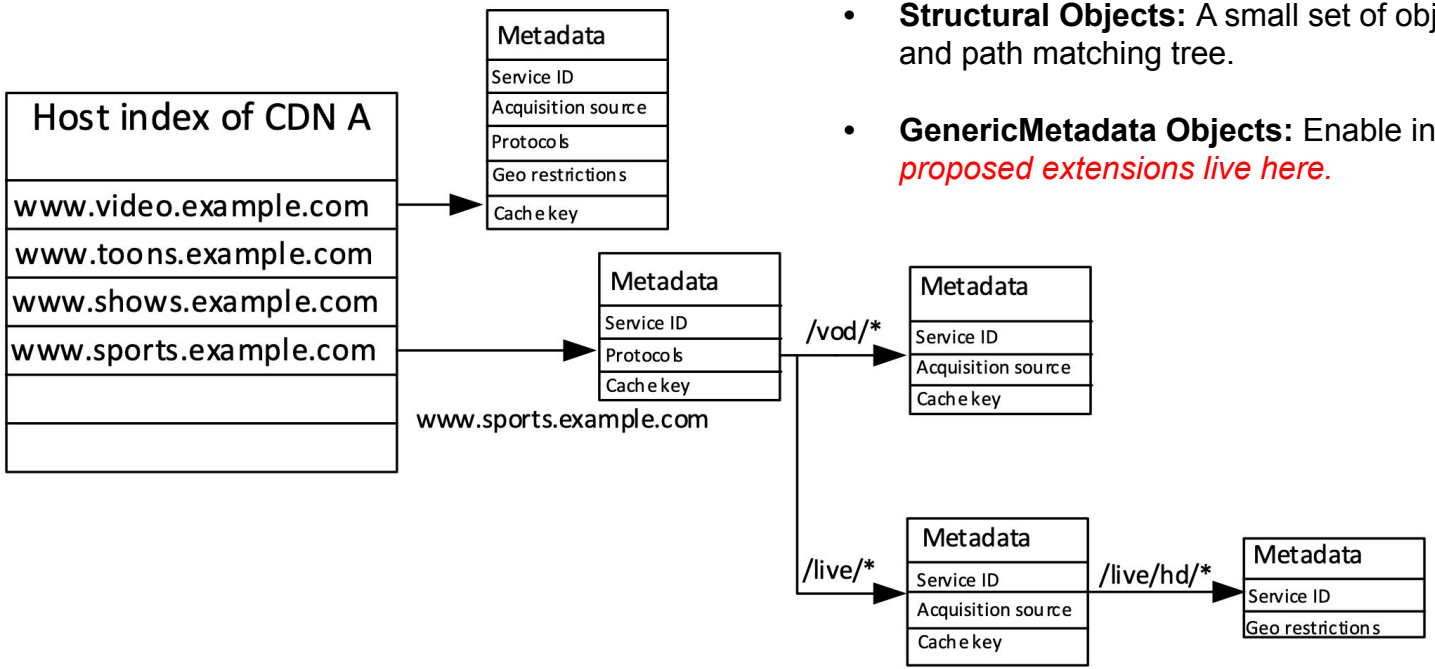
The SVA Configuration Interface initiative addresses these gaps!

streaming video alliance

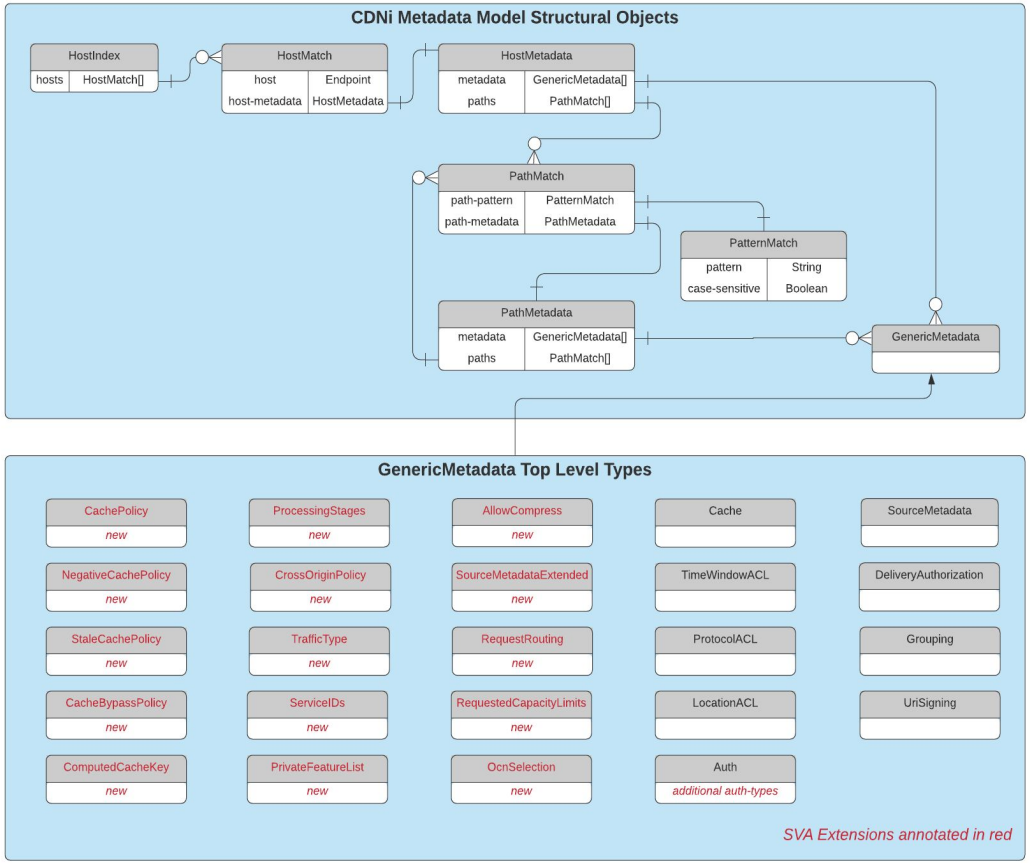
CDNI Metadata Model (RFC 8006)

Key Concepts:

- **Inheritance model**: Content Delivery Metadata (caching and access rules) defined at the host level can be overridden at the path level.
- **Structural Objects:** A small set of objects that define the host and path matching tree.
- **GenericMetadata Objects:** Enable infinite extensibility. *All our proposed extensions live here.*



CDNi Metadata Model Schema



Metadata Model Extensions

The following requirements and proposed extensions are documented in detail in the *SVA Configuration Interface Part 2* specification and are being submitted to the IETF as extensions to RFC-8006:

- Enhanced Source/Origin Definitions:
  - Origin Load Balancing, Failover
  - Origin Authentication Methods

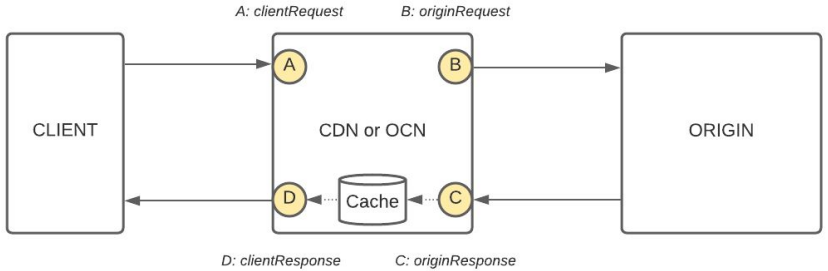
- Cache Control Policies & Computed Keys
- Dynamic CORS Headers
- Traffic Type Metadata
- Service ID Metadata
- SVA Open Caching Configuration Metadata
- Private Features for extensibility
- Processing Stages with an Expression Language(see next slide)

CDNi Metadata Extension: Processing Stages

Allows metadata rules to be applied conditionally at a specific stage in the pipeline, based on matching elements of HTTP requests & responses. A rich expression language is provided to specify matching rules and synthesis dynamic values.

Stage-specific processing enables:

- Application of metadata (such as cache policies)
- Request Transformations (Header modifications, URI re-writes)
- Response Transformations (Header modifications, status code overrides)
- Generating Synthetic Responses



clientRequest - Rules run on the client request prior to further processing.

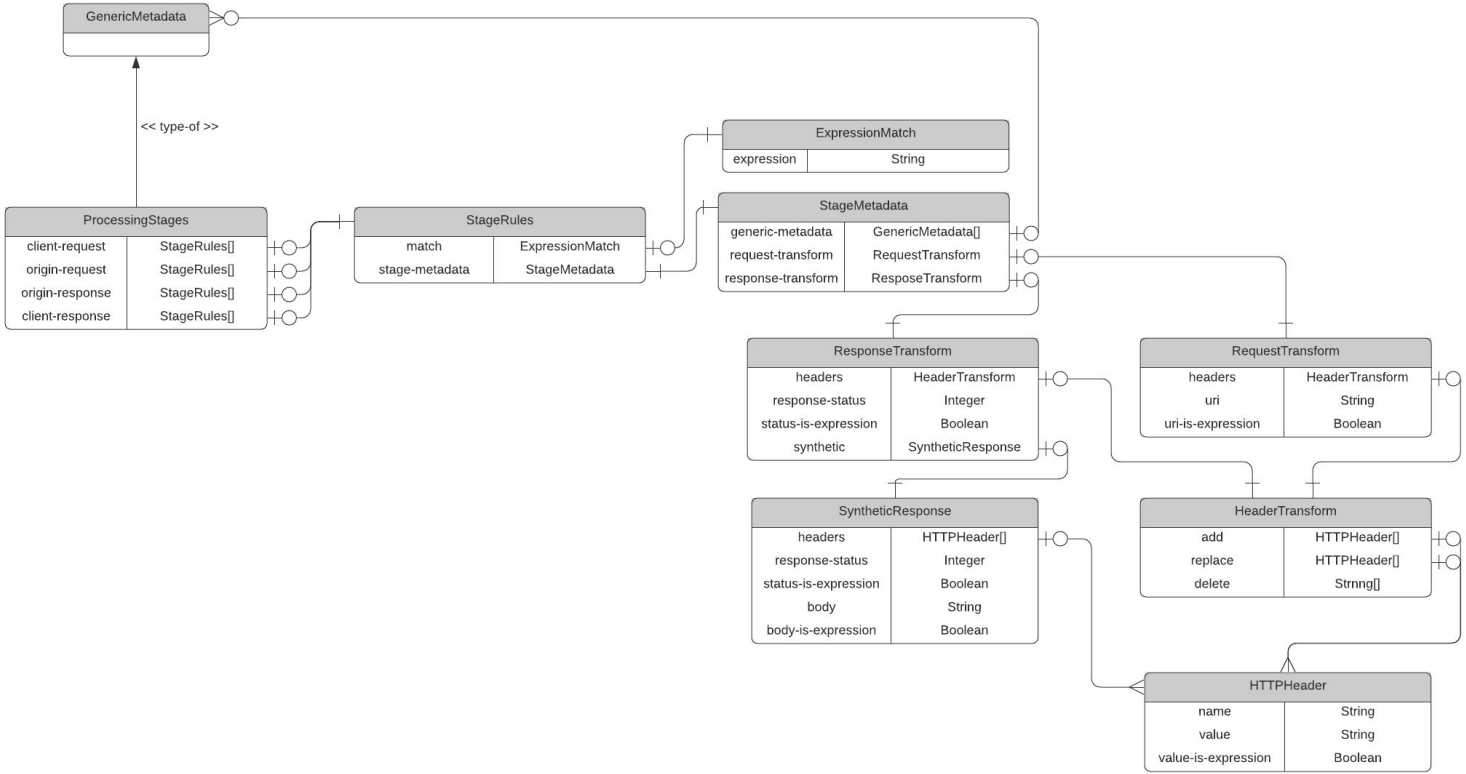
originRequest - Rules run prior to making a request to the origin.

originResponse - Rules run after response is received from the origin and before being placed in cache.

clientResponse - Rules run prior to sending the response to the client. If response is from cache, rules are applied to the response retrieved from cache prior to sending to the client.

streaming video alliance

Stage Processing Object Model



Processing Stages Example

```
{
  "hosts": [
    {
      "host": "edge.example.com",
      "host-metadata": {
        "metadata": [
          {
            "generic-metadata-type": "MI.ProcessingStages",
            "generic-metadata-value": {
              "origin-response": [
                {
                  "match": {
                    "expression": "resp.status == 200"
                  },
                  "stage-metadata": {
                    "generic-metadata": [
                      {
                        "generic-metadata-type": "MI.CachePolicy",
                        "generic-metadata-value": {
                          "internal": "5",
                          "external": "no-cache",
                          "force": "false"
                        }
                      }
                    ]
                  }
                }
              ]
            }
          }
        ]
      }
    }
  ]
}
```

A complete example using the SVA `ProcessingStages` and `CachePolicy` extensions to the CDNI metadata model.

In this example, clients are directed to not cache content when there is a 200 response from the origin, with the CDN maintaining internal caches for 5 seconds to protect the origin from being overwhelmed.

Expression Language Examples

The CDNI Metadata Expression Language provides a syntax with a rich set of variables, operators, and built-in functions to facilitate use cases within the extended CDNi metadata model.

ExpressionMatch where the expression is true if the user-agent (glob) matches *Safari* and the referrer equals www.example.com.

```
{
  "expression": "req.h.user-agent *= '*Safari*'
      and req.h.referrer == 'www.example.com'"
}
```

Add a Set-Cookie header with a dynamically computed cookie value (concatenating user agent and host name).

```
{
    "response-transform":{
        "headers":{
            "add":[
                {
                    "name":"Set-Cookie",
                    "value":"$req.h.user-agent - $req.h.host",
                    "value-is-expression":true
                }
            ]
        }
    }
}
```

streaming video alliance

Capabilities Interface

- The proposed CDNi Metadata Model extensions are optional, with dCDNs able to advertise their support via the Footprint & Capabilities Interface (FCI).
- Any extension that is embodied as a new GenericMetadata object can be advertised as supported via the CDNi standard FCI.Metadata object.
- Some proposed extensions entail many features, and it is quite possible that a dCDN may support some (but not all) of these features.
- To allow for more fine-grained advertisement of feature support, additional FCI objects will be defined containing feature flags that are specific to each extended GenericMetadata object.

Extending the Metadata Interface

In addition to extending the CDNi Metadata model, the SVA is also working on APIs that extend the interface:

- Extend the basic metadata retrieval interface defined in RFC-8006 with metadata publishing capabilities to allow a uCDN to publish and delete metadata on a dCDN.
- Add capabilities to publish and reference sets of named GenericMetadata Objects (extending the current HREF concept).
- Provide CDN configuration life cycle management capabilities such as publishing, versioning, staged deployments, profiles, and templates.
- Provide configuration for additional aspects of CDN operation such as provisioning of Certificates and configuration of log delivery endpoints.

Draft Status:

draft-goldstein-cdni-metadata-model-extensions-01

streaming
video
alliance

Changes from Revision 00

- Each newly proposed `GenericMetadata` object states the purpose of the object with summary of the requirements.
- Each newly proposed `GenericMetadata` object has a fully specified set of properties and descriptions of each attribute. This was omitted in the original draft and was the source of many questions.
- Some examples added (more to come in revision 02).
- Diagrams illustrating the Processing Stages control flow and data model have been added.
- A specification of the proposed Metadata Expression Language (MEL) has been added.

Questions from Revision 00

AllowCompress: why should a dCDN allow a uCDN to dictate its use of compression?

UCDNs can be a Content Provider origin and may have specific requirements for certain types of objects (such as stream manifests) to be served uncompressed.

CachePolicy: why should a dCDN allow a uCDN to dictate caching policies? could this be achieved with cache-control headers?

This capability extends Cache-control headers by allowing specification of both client and CDN caching parameters. dCDNs add a caching layer that require distinct control parameters.

ComputedCacheKey: is there something specific that is missing from the existing Cache metadata?

Yes, there can be cases, for example, when the cache key is computed based on the value of an HTTP response header.

NegativeCacheKey: what kind of caching policy for what kind of error?

As an example, it may be desirable to cache error responses at the CDN for a short period of time to prevent an overwhelmed origin service from being flooded with requests.

CacheBypassPolicy: is this specifying something that helps the dCDN identify requests for which to bypass caching?

This capability's purpose is to facilitate QA and testing scenarios where caching rules on the CDN need to be overridden on a per-request basis.

Questions from Revision 00

OcnSelection: presumably OCN is open caching node? does this dictate how the dCDN RR selects a surrogate?

Yes, OcnSelection allows the uCDN to indicate to the dCDN a preference in terms of the surrogate node selection.

PrivateFeatureList: it's not clear what this one is?

The goal is to provide a controlled extension mechanism without the need for additional GenericMetadata objects. This will be used by mutual agreement between uCDNs and dCDNs and will enable organizations such as the SVA to define a set of features specific to their needs.

RequestedCapacityLimits: why does the uCDN need to control the dCDN capacity advertisement? couldn't it just ignore the advertisement?

This object allows the uCDN to request a change in the capacities that the dCDN may have advertised via FCI.CapacityLimits. Typical use case will involve a uCDN wanting to delegate more traffic to a dCDN than originally planned for.

RequestRouting: is this so that the uCDN can dictate how the dCDN redirects to further dCDNs? why should a dCDN allow a uCDN to dictate redirection mode?

MI.RequestRouting is a new GenericMetadata object that allows the uCDN to force the dCDN request routing mode(s) to be applied when working in iterative redirection mode.

Questions from Revision 00

ServiceIds: it's not clear what these IDs are?

Provides for two tiers of ideniifers (serviceIDs and propertyIDs) that are typical in commercial CDN and Open Caching systems. Usage is optional, and interpretation of these identifiers is based on mutual understanding between parties.

SourceMetadataExtended: it would be helpful to see the actual properties.

All properties listed in revision 01.

StaleContentCachingPolicy: similar to other questions about dictating dCDN policy.

Typical use would allow the content provider to specify that stale content be served from cache for a specified time period while refreshes from the origin occur asynchronously.

TrafficType: it's not clear what the traffic types are or how they would be used by dCDNs.

Details have been provided in revision 01. Standard types are VOD, LIVE, and OBJECT-DOWNLOAD, along with an option for free-form traffic type hints. Interpretation of this metadata is implementation specific.

Conclusion

Based on the contents of this presentation, Can the CDNI working group accept version 01 of this document as a Working Group Draft?