SVA Configuration Interface

IETF/CDNi Metadata Model Extensions

July 2022 (IETF 114)

streaming video alliance

SVA Configuration Metadata Interface

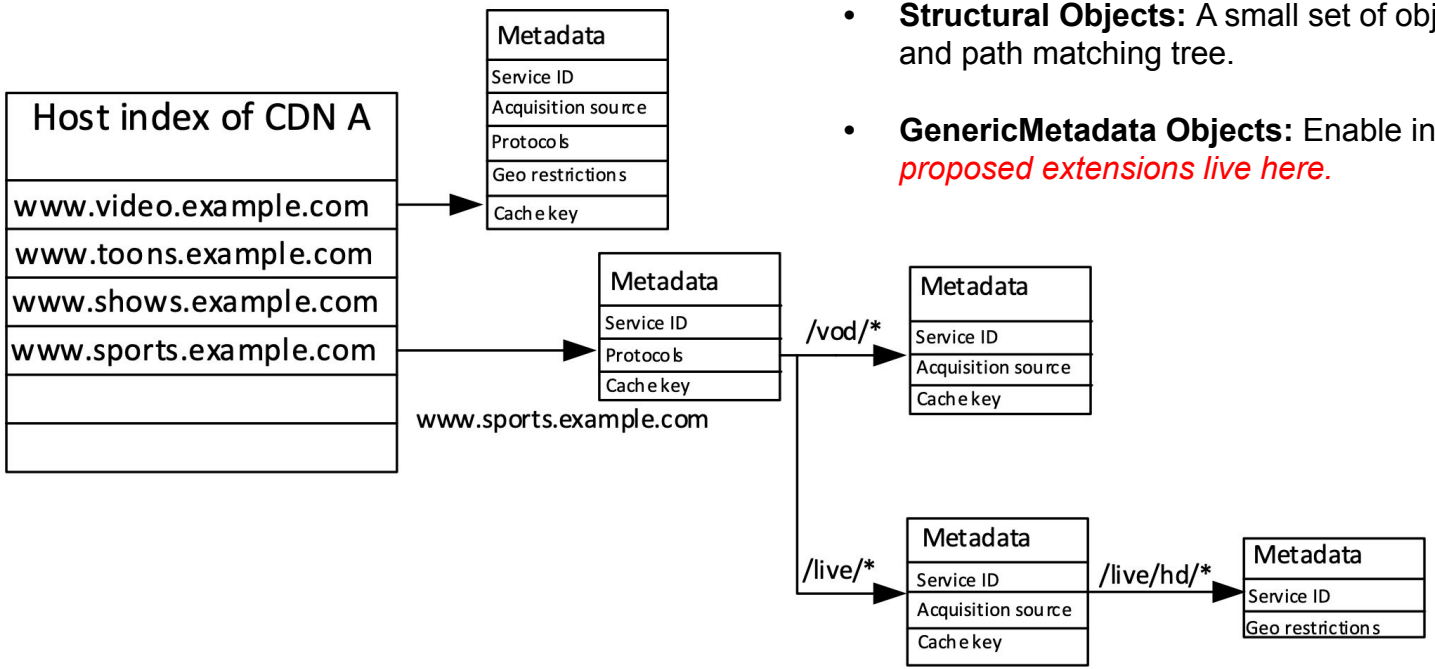
- **Problem Statement:** The SVA membership identified the need for an industry-standard API and configuration metadata model becomes increasingly important as content and service providers automate more of their operations, and as technologies such as Open Caching require coordinated content delivery configurations.

- **The SVA Plan:**
  - Use the CDNI Metadata Object Model (RFC-8006) as a starting point, layering in additional GenericMetadata Objects and an expression language to meet more complex requirements.
  - Use the CDNI Metadata Interface as the basis for a *Simple Configuration Metadata API*, adding a simple publishing capability for uCDN to push metadata to dCDN.
  - Create an *Advanced Configuration Metadata API* to breakout metadata publishing and deployment as distinct activities, and to provide reusable metadata definitions.
  - Submit metadata model extensions and the expression language to IETF CDNI WG as a single draft.

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.*



SVA Project Status

- **SVA Configuration Interface Specification**
  - Version 1.0 published in Jan 2022
    - *Part 1: Overview & Architecture*
    - *Part 2: Proposed Extensions to CDNI Metadata Object Model*
    - *Part 3: Publishing Layer APIs* (Simple Configuration Metadata API)
  - Version 1.1 to be published in August 2022 (minor updates)
  - Version 2.0 to be published Jan 2023
    - Major updates, breaking document into many smaller parts
    - Many additional metadata objects
    - Advanced Publishing API defined

- **IETF CDNI Draft (Same content as SVA part-2 document)**
  - draft-goldstein-cdni-metadata-model-extensions-02 published for IETF 113
  - Work on-hold until Version 2 of SVA Project gets further along, at which time we will carve up the IETF Draft into several smaller drafts (see next slide)

Proposal: Six Smaller IETF Drafts

CDNI Metadata Expression Language (CDNI-MEL)
- Supports matching & value synthesis
- Variables, Expressions, Built-in Functions
- User-defined Variables

Processing Stages Metadata
- HTTP pipeline with conditional metadata
- Request/response transformations

Cache Control Metadata
- Positive & Negative Policies
- Stale Content Policies
- Cache Bypass Control
- Computed Cache Keys

Source Access Control Metadata
- Multi-Source Load Balancing & Failover
- Source Authentication Rules
- Allowed Source Access Protocols
- Source Connection Control (timeouts/retries)

Client Access Control Metadata
- Auth Token Metadata (with CTA-WAVE)
- Certificate & Encryption Metadata

Edge Control Metadata
- Cross Origin Policies (CORS Headers)
- Downstream Compression Policies
- Client Connection Control (timeouts/retries)
- Open Caching Node Selection Metadata
- Traffic Types

streaming video alliance

** Items in red are additions since the current IETF Draft*

Impact on FCI (Capabilities Interface)

- All of the 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 extensions entail many features, and it is 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. *These may be defined in another IETF draft.*

Discussion Items

- What's the best home for defining FCI objects related to GenericMetadata extensions?

- What is the best path for defining named footprint scopes to be used with various interfaces that may consume footprint-level capabilities or configurations?
  - Example Consumers - Logging API, Capacity Insights API Etc.
  - Additional discussion point : Feasibility of overlapping and/or composite footprint objects.

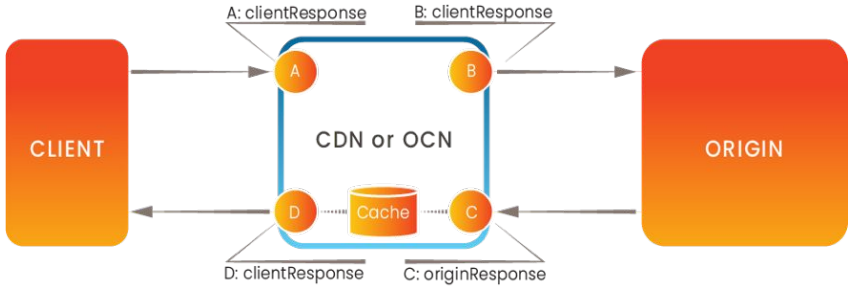
Deeper Dives (Time Permitting…)

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.

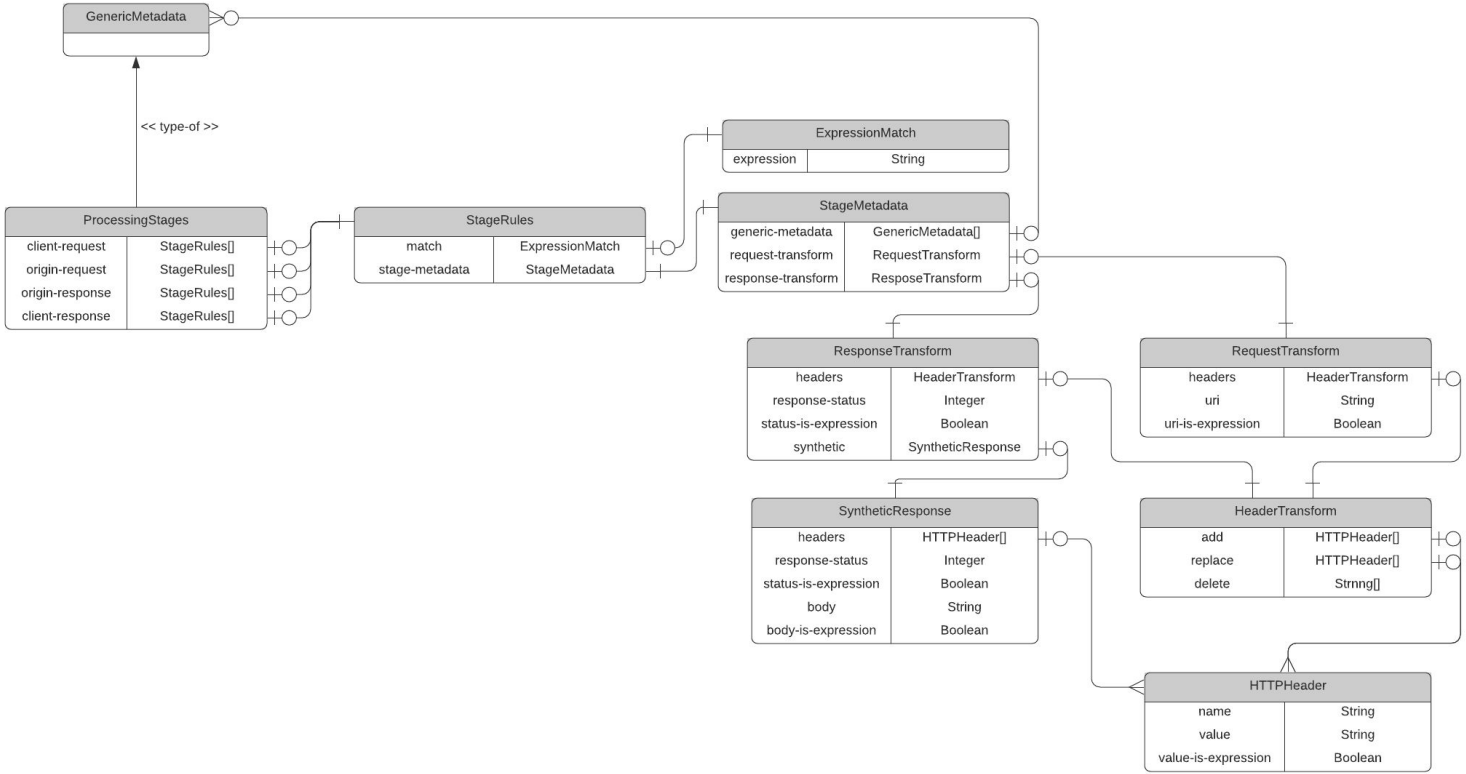
originRequest - Rules run prior to making a request to the origin.

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

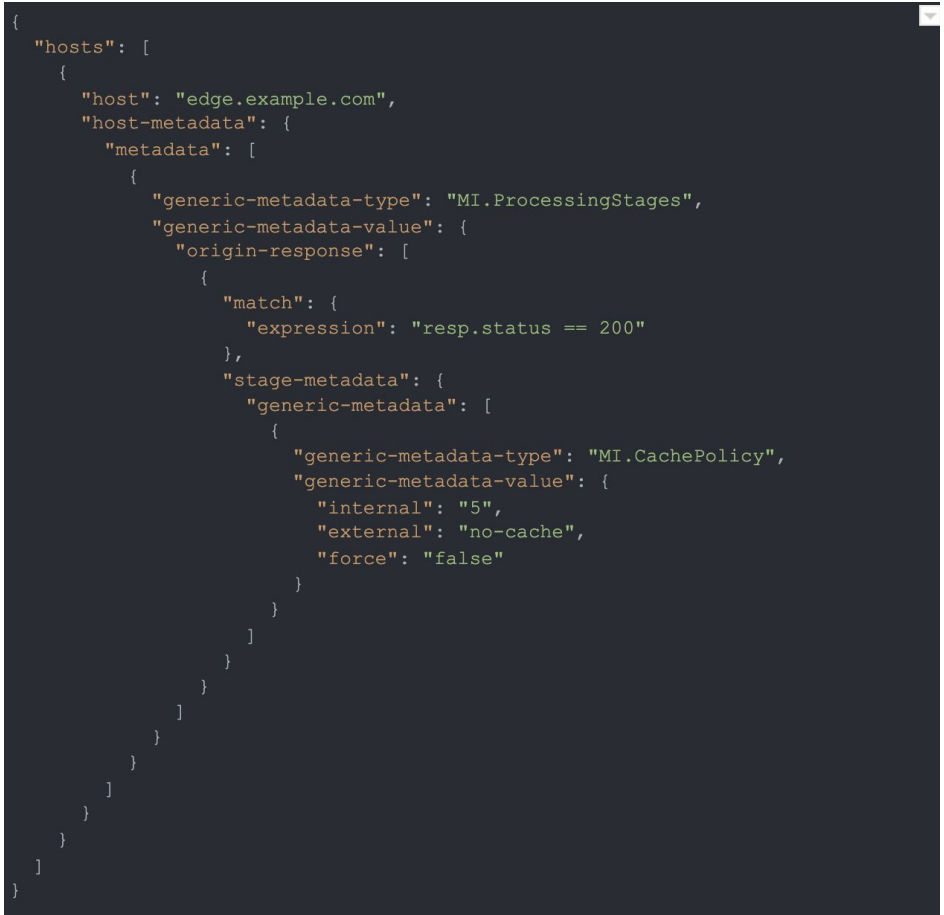
clientResponse - Rules run prior to sending the response to the client. If response is from cache, rules are applied to the response retrieved from cache prior to sending to the client.

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

Source Connection Control Model