SVTA Configuration Interface

IETF/CDNi Metadata Model Extensions Update

November 2023(IETF 118)

Metadata Model Extension Drafts

- **CDNI WG Drafts**
  - draft-ietf-cdni-protected-secrets-metadata-00 replaces: draft-rosenblum-cdni-protected-secrets-metadata
  - draft-ietf-cdni-cache-control-metadata-00 replaces: draft-power-cdni-cache-control-metadata
  - draft-ietf-cdni-edge-control-metadata-00 replaces: draft-siloniz-cdni-edge-control-metadata
- **New Individual submissions**
  - draft-power-metadata-expression-language-00
  - draft-goldstein-processing-stages-metadata-00

Protected Secrets Metadata

- draft-ietf-cdni-protected-secrets-metadata-00

- Addressed feedback from Kevin Ma. Significantly:
    - Lots of language cleanup, RFC2119 compliant language for requirement levels
    - HashiCorp store types now denoted in the MI object name
    - IANA Considerations added
    - Fixed links to external SVTA documents
    - Normative & Informative References: Cleaned up

- Work remaining:
    - Sequence diagrams corresponding to workflow examples
    - Do we eliminate the FCI wrapper objects and use the MI objects directly as Capabilities on the advertisement side? Needs discussion.

Cache Control Metadata

- draft-ietf-cdni-cache-control-metadata-00

- Addressed all the feedback from Kevin Ma. Significantly:
  - **MI.CachePolicy**: clarified definitions of the internal and external properties.
  - **MI.StaleContentCachePolicy**: clarified “revalidating” vs “refreshing” and renamed `failed-refresh-ttl` to `failed-revalidation-delta-seconds` with clearer description.
  - Reorganized all the examples:
    - Each MI object definition has a minimal example
    - A new *Informative Examples* section illustrates the use of these MI objects in context of other structures such as Processing Stages.
  - IANA Considerations added
  - Fixed links to external SVTA documents
  - Normative & Informative References: Cleaned up, with Processing Stages moved to Informative.

Edge Control Metadata

- draft-ietf-cdni-edge-control-metadata-00

- Addressed all the feedback from Kevin Ma. Significantly:
  - **MI.CrossoriginPolicy:**
    - “apply-to-all-methods” default behavior changed and renamed to “preflight-only”. Definition that MI.CrossoriginPolicy affects all HTTP methods by default.
  - **MI.AccessControlAllowOrigin**:
    - “allow-list” type definition changed as MI.PatternMatch was not suitable enough.
  - Reorganized examples:
    - Separate section for MI.CrossoriginPolicy examples
    - A new *Informative Examples* section illustrates the use some MI objects in context of other structures such as Processing Stages
  - Some questions about MtS and StD responded in the mailing list.
  - IANA Considerations added
  - Fixed links to external SVTA documents
  - Normative & Informative References: Cleaned up

Metadata Expression Language (MEL)

- draft-power-metadata-expression-language-00

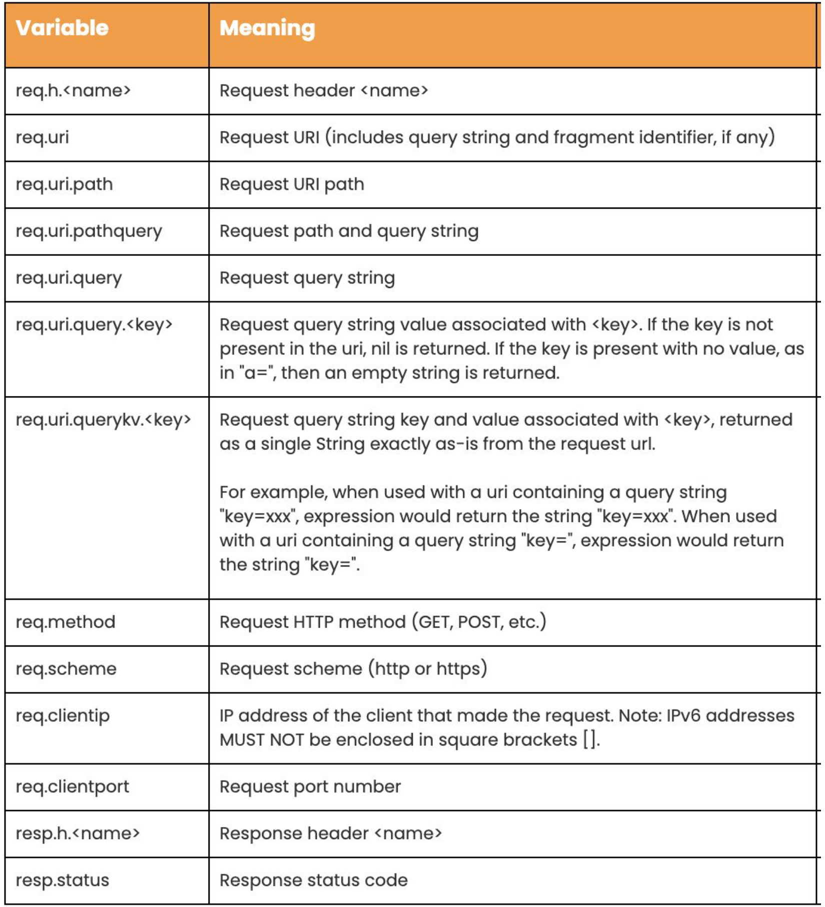
- Provides a syntax with a rich set of variables, operators, and built-in functions to facilitate use cases within the extended CDNI metadata model:

  - **Match Expressions** - Expressions that evaluate to a Boolean are used to match against an HTTP header value and/or query param so that metadata can be applied conditionally.

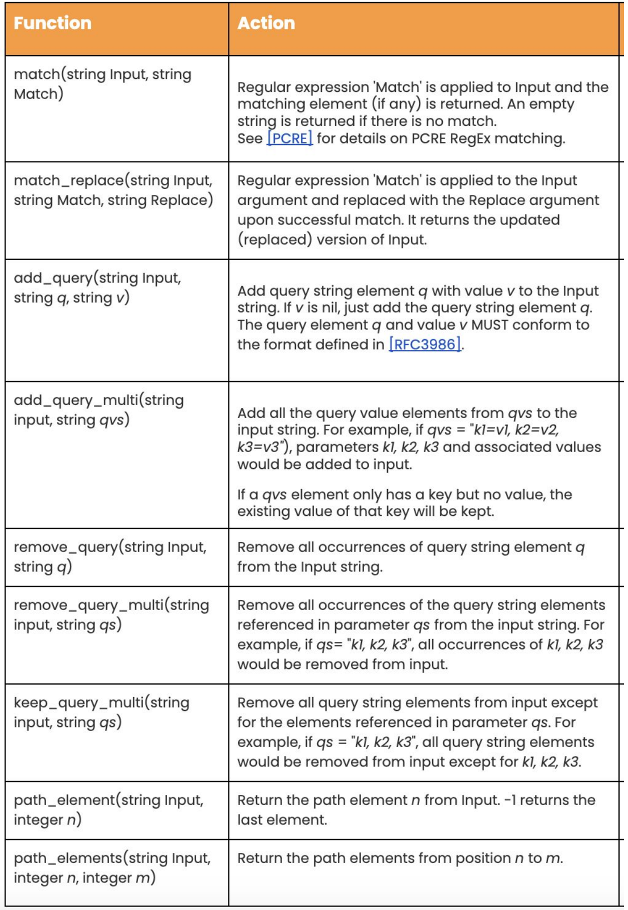
  - **Value Expressions** - Enable the dynamic construction of a value to be used in scenarios such as constructing a cache key, setting an HTTP response header, rewriting a request URI, or dynamically generating a response body.

- This is NOT a programming language!

MEL: Variables & Built-In Functions



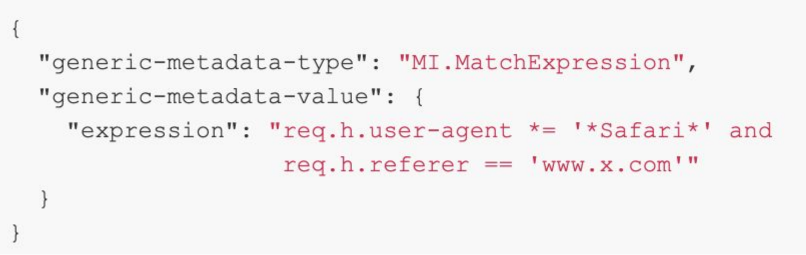
Variable	Meaning
req.h.<name>	Request header <name>
req.uri	Request URI (includes query string and fragment identifier, if any)
req.uri.path	Request URI path
req.uri.pathquery	Request path and query string
req.uri.query	Request query string
req.uri.query.<key>	Request query string value associated with <key>. If the key is not present in the uri, nil is returned. If the key is present with no value, as in "a=", then an empty string is returned.
req.uri.querykv.<key>	Request query string key and value associated with <key>, returned as a single String exactly as-is from the request url. For example, when used with a uri containing a query string "key=xxx", expression would return the string "key=xxx". When used with a uri containing a query string "key=", expression would return the string "key=".
req.method	Request HTTP method (GET, POST, etc.)
req.scheme	Request scheme (http or https)
req.clientip	IP address of the client that made the request. Note: IPv6 addresses MUST NOT be enclosed in square brackets [].
req.clientport	Request port number
resp.h.<name>	Response header <name>
resp.status	Response status code



Function	Action
match(string Input, string Match)	Regular expression 'Match' is applied to Input and the matching element (if any) is returned. An empty string is returned if there is no match. See [PCRE] for details on PCRE RegEx matching.
match_replace(string Input, string Match, string Replace)	Regular expression 'Match' is applied to the Input argument and replaced with the Replace argument upon successful match. It returns the updated (replaced) version of Input.
add_query(string Input, string *q*, string *v*)	Add query string element *q* with value *v* to the Input string. If *v* is nil, just add the query string element *q*. The query element *q* and value *v* MUST conform to the format defined in [RFC3986].
add_query_multi(string input, string *qvs*)	Add all the query value elements from *qvs* to the input string. For example, if *qvs* = "*k1=v1, k2=v2, k3=v3*"), parameters *k1, k2, k3* and associated values would be added to input. If a *qvs* element only has a key but no value, the existing value of that key will be kept.
remove_query(string Input, string *q*)	Remove all occurrences of query string element *q* from the Input string.
remove_query_multi(string input, string *qs*)	Remove all occurrences of the query string elements referenced in parameter *qs* from the input string. For example, if *qs*= "*k1, k2, k3*", all occurrences of *k1, k2, k3* would be removed from input.
keep_query_multi(string input, string *qs*)	Remove all query string elements from input except for the elements referenced in parameter *qs*. For example, if *qs* = "*k1, k2, k3*", all query string elements would be removed from input except for *k1, k2, k3*.
path_element(string Input, integer *n*)	Return the path element *n* from Input. -1 returns the last element.
path_elements(string Input, integer *n*, integer *m*)	Return the path elements from position *n* to *m*.

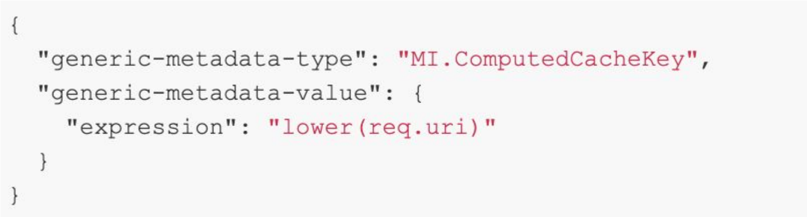
MEL: Examples

Match Expression evaluating multiple request headers:



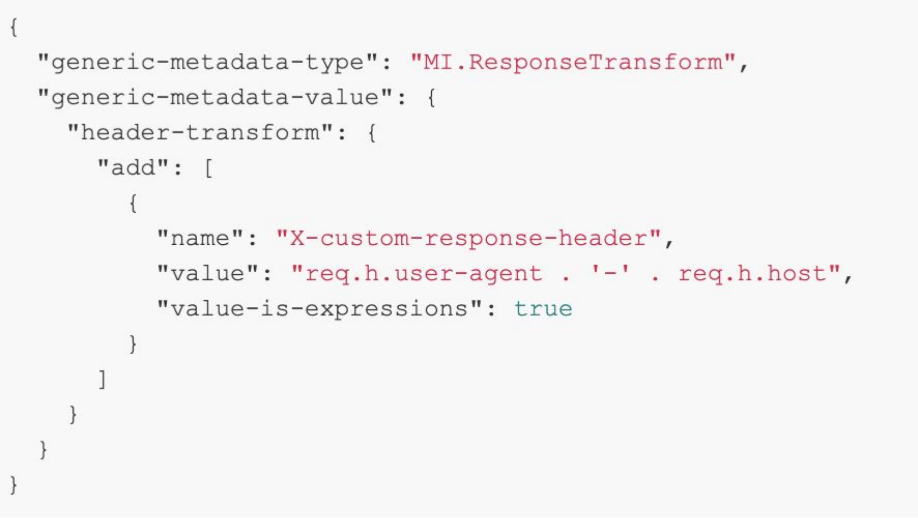
```
{
  "generic-metadata-type": "MI.MatchExpression",
  "generic-metadata-value": {
    "expression": "req.h.user-agent *= '*Safari*' and
                   req.h.referer == 'www.x.com'"
  }
}
```

Value Expression setting cache key to the lower-cased request URI:



```
{
  "generic-metadata-type": "MI.ComputedCacheKey",
  "generic-metadata-value": {
    "expression": "lower(req.uri)"
  }
}
```

Value Expression constructing a response header from concatenated request headers:



```
{
  "generic-metadata-type": "MI.ResponseTransform",
  "generic-metadata-value": {
    "header-transform": {
      "add": [
        {
          "name": "X-custom-response-header",
          "value": "req.h.user-agent . '-' . req.h.host",
          "value-is-expressions": true
        }
      ]
    }
  }
}
```

MEL: Error Handling

- **Compile-Time Errors**
  - To ensure reliable service, all CDNI metadata configurations MUST be validated for syntax errors before they are ingested into a dCDN.
  - If errors are detected in a new configuration, the configuration MUST be rejected.
  - Examples:
    - Unknown MEL variable name referenced in an expression
    - Unknown MEL operator, keyword, or functions referenced in an expression
    - Incorrect number of arguments used in an expression operator or function
    - Incorrect type of argument used in an expression operator or function

- **Run-Time Errors**
  - If a runtime error is detected when processing a request, the request SHOULD be terminated, and an HTTP 500 'Internal Server Error' returned to the caller.
  - Examples:
    - Failure to allocate memory when evaluating a MEL expression
    - Incorrect runtime argument type in a MEL expression

MEL: FCI Capabilities Advertisement

- Since implementing the full MEL specification may be complex and onerous, a mechanism is provided for a dCDN to advertise what portions of the MEL standard it supports (if any).
- **`FCI.SupportedMELFeatures`** can be provided within an FCI Capabilities Advertisement object for a given footprint.
- `FCI.SupportedMELFeatures` allows the dCDN to advertise support for specific:
  - MEL keywords
  - MEL operators
  - MEL variables
  - MEL built-in functions

Processing Stages Metadata

- draft-goldstein-processing-stages-metadata-00

- Processing Stages Metadata is designed to leverage the Metadata Expression Language (MEL) to address common CDN and Open Caching requirements for:
    - Conditional application of caching rules.
    - Transformations of HTTP requests and responses.

- Defines four stages in the request processing pipeline, where conditional matching and transformations can be applied at any of the stages.

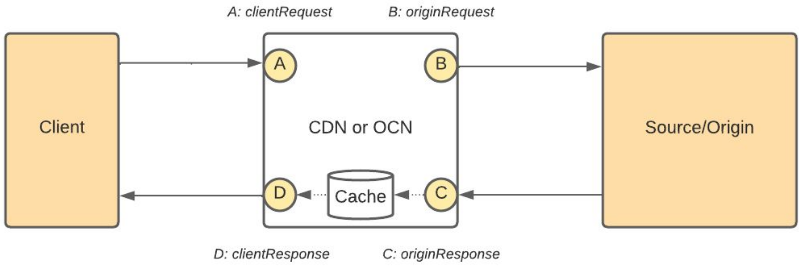
- This is NOT a programming language! But it does provide structured if-else constructs.

Processing Stages Flow

Allows metadata rules to be applied conditionally at a specific stage in the pipeline, based on matching elements of HTTP requests & responses.

Typical stage-specific processing use cases:

- Specialized cache policies and access controls based on complex evaluations of request headers and URIs.
- Request Transformations such as HTTP header modifications or URI rewrites.
- Response Transformations such as suppression of or additions to origin response status codes or headers.
- Modification of cached content prior to sending to client.
- Generating of Synthetic Responses.



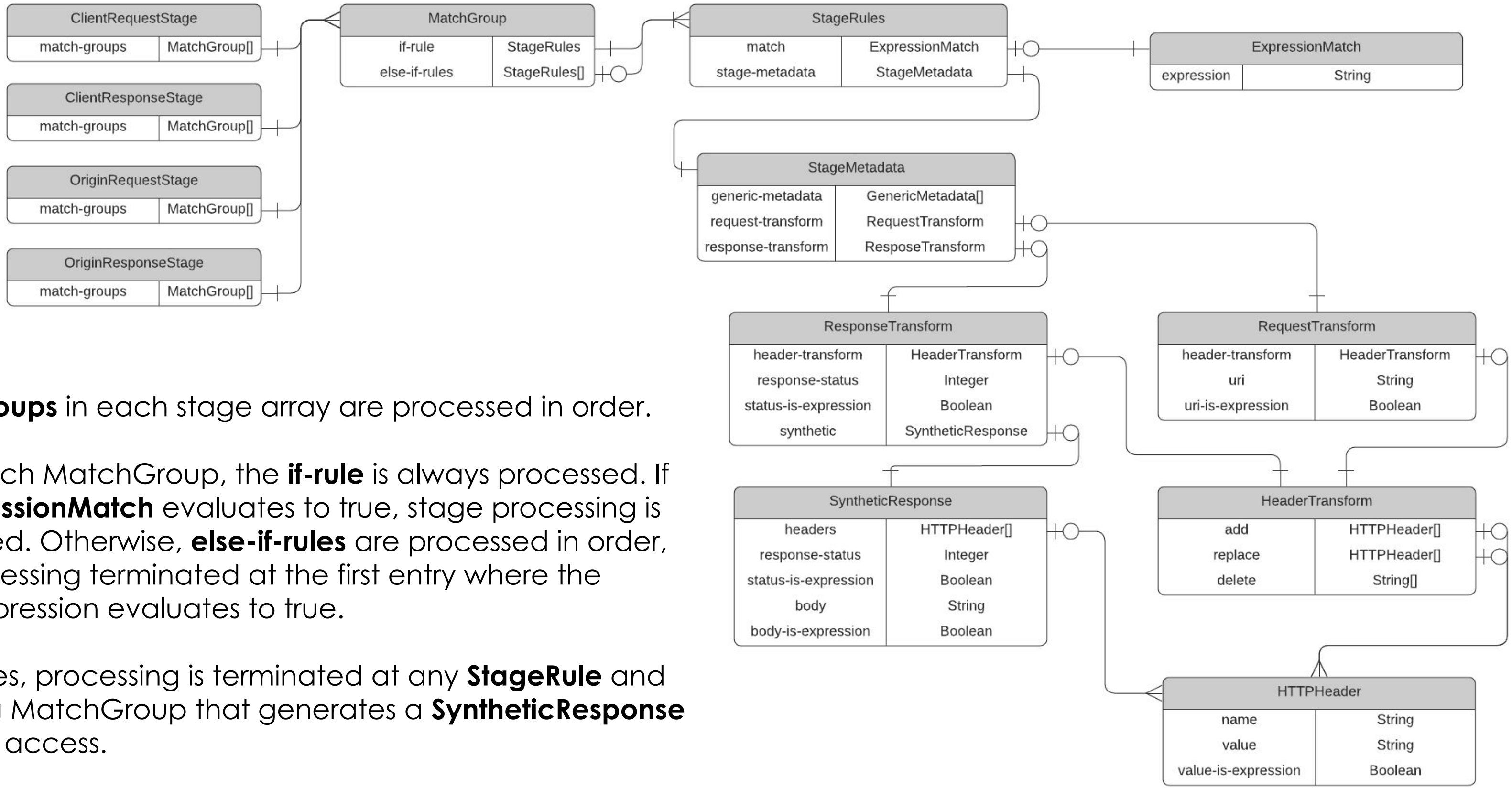
clientRequest - Rules run on the inbound client request prior to further processing.

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

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

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

Processing Stages Object Model



ClientRequestStage

match-groups	MatchGroup[]

ClientResponseStage

match-groups	MatchGroup[]

OriginRequestStage

match-groups	MatchGroup[]

OriginResponseStage

match-groups	MatchGroup[]

MatchGroup

if-rule	StageRules
else-if-rules	StageRules[]

StageRules

match	ExpressionMatch
stage-metadata	StageMetadata

ExpressionMatch

expression	String

StageMetadata

generic-metadata	GenericMetadata[]
request-transform	RequestTransform
response-transform	ResposeTransform

ResponseTransform

header-transform	HeaderTransform
response-status	Integer
status-is-expression	Boolean
synthetic	SyntheticResponse

RequestTransform

header-transform	HeaderTransform
uri	String
uri-is-expression	Boolean

SyntheticResponse

headers	HTTPHeader[]
response-status	Integer
status-is-expression	Boolean
body	String
body-is-expression	Boolean

HeaderTransform

add	HTTPHeader[]
replace	HTTPHeader[]
delete	String[]

HTTPHeader

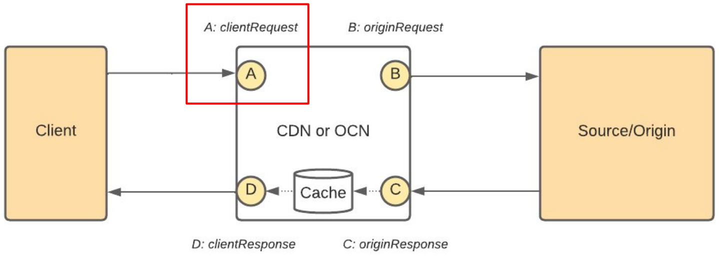
name	String
value	String
value-is-expression	Boolean

MatchGroups in each stage array are processed in order.

Within each MatchGroup, the **if-rule** is always processed. If the **ExpressionMatch** evaluates to true, stage processing is terminated. Otherwise, **else-if-rules** are processed in order, with processing terminated at the first entry where the MatchExpression evaluates to true.

In all cases, processing is terminated at any **StageRule** and enclosing MatchGroup that generates a **SyntheticResponse** or denies access.

ClientRequest Stage Example



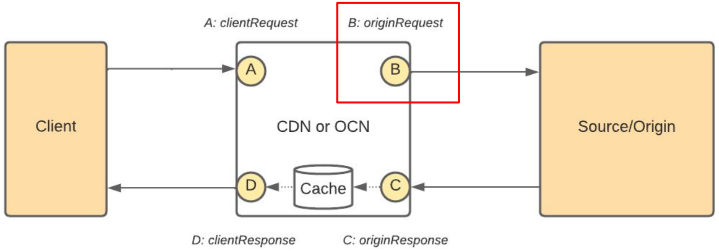
- Example examines the user agent of inbound HTTP request from client.

- A **MEL Match Expression** is used to test if the user agent is from a mobile device. If so, a synthetic response is generated with a 405 status code and custom headers.

- A **MEL Value Expression** is used to synthesise response body text.

```
{
  "generic-metadata-type": "MI.ClientRequestStage",
  "generic-metadata-value": {
    "match-groups": [
      {
        "if-rule": {
          "match": {
            "expression": "req.h.user-agent *= '*Mobile*'"
          },
          "stage-metadata": {
            "generic-metadata": [
              {
                "generic-metadata-type": "MI.SyntheticResponse",
                "generic-metadata-value": {
                  "headers": [
                    {
                      "name": "content-type",
                      "value": "text/plain"
                    },
                    {
                      "name": "X-custom-response-header",
                      "value": "some static value"
                    }
                  ],
                  "response-status": "405",
                  "response-body": "'Sorry, Access to resource ' .
                                    req.uri . ' not allowed'",
                  "body-is-expression": true
                }
              }
            ]
          }
        }
      }
    ]
  }
}
```

OriginRequest Stage Example

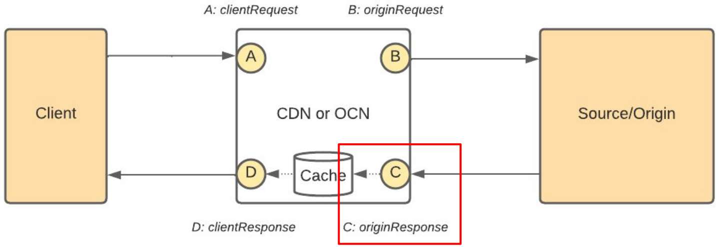


- Example uses a **HeaderTransform** to modify request headers on requests made to the origin. The absence of a match expression means that the if-rule is always applied.

- The HeaderTransform illustrates use of the **add** and **delete** properties to add and remove HTTP headers.

```
{
  "generic-metadata-type": "MI.OriginRequestStage",
  "generic-metadata-value": {
    "match-groups": [
      {
        "if-rule": {
          "stage-metadata": {
            "generic-metadata": [
              {
                "generic-metadata-type": "MI.HeaderTransform",
                "generic-metadata-value": {
                  "add": [
                    {
                      "name": "X-custom-header1",
                      "value": "header-value 1"
                    },
                    {
                      "name": "X-custom-header2",
                      "value": "header-value 2"
                    }
                  ],
                  "delete": [
                    "Authorization",
                    "Accept-Language"
                  ]
                }
              }
            ]
          }
        }
      }
    ]
  }
}
```

OriginResponse Stage Examples



- Example uses if/else construct to apply different Cache Policies depending on the HTTP response code received from origin.
- Can be expanded to contain as many **else-if-rules** blocks as needed.

```
{
  "generic-metadata-type": "MI.OriginResponseStage",
  "generic-metadata-value": {
    "match-groups": [
      {
        "if-rule": {
          "match": {
            "expression": "resp.status == 200"
          },
          "stage-metadata": {
            "generic-metadata": [
              {
                "generic-metadata-type": "MI.CachePolicy",
                "generic-metadata-value": {} < success policy/ttl >
              }
            ]
          }
        },
        "else-if-rules": [
          {
            "match": {
              "expression": "resp.status == 503 or resp.status == 504"
            },
            "stage-metadata": {
              "generic-metadata": [
                {
                  "generic-metadata-type": "MI.CachePolicy",
                  "generic-metadata-value": {}
                }
              ]
            }
          }
        ]
      }
    ]
  }
}
```

Conclusion

Based on the contents of this presentation, Can the CDNI working group accept these two new documents as a Working Group Draft?

- draft-power-metadata-expression-language-00
- draft-goldstein-processing-stages-metadata-00