Source Access Control

Chris Lemmons on behalf of Glenn Goldstein

IETF CDNI WG

2025-07-20

Scope

- Defines how dCDNs access origins
- Provides extended object as alternative to `MI.SourceMetadata` from RFC 8006
- Describes mechanisms for multi-source access control, load balancing, failover, and authentication

MI.SourceMetadataExtended

- New fields:
  - `source-detention`: Put bad origins in the naughty corner.
  - `load-balance`: Define how load is balanced.

MI.SourceExtended

- New fields:
  - `origin-host`: Host header.
  - `sni-host`: The SNI to use upstream.
  - MEL Enablers:
    - `endpoint-is-expression`
    - `origin-host-is-expression`
    - `sni-host-is-expression`
  - `webroot`: Prepended to the path requested upstream.
  - `follow-redirects`: Return or follow 3xx redirects.

MI.SourceExtended (cont.)

- More new fields
  - Connection failure control:
    - `timeout-ms`: Simple connection timeout.
    - `connection-control`: Advanced connection control.
  - Failover control:
    - `failover-errors`: Simple list of status codes that trigger failover.
    - `http-code-failover`: Advanced failover control.
  - `endpoint-detention`: Put bad endpoints in the naughty corner.

MI.SourceConnectionControl

More detailed than `timeout-ms`.

- Connection Setup:
  - `connection-setup-timeout-ms`: How long to wait.
  - `connection-setup-timeout-ms-actions`: What to do.
- First Byte Read:
  - `first-byte-read-timeout-ms`: How long to wait.
  - `first-byte-read-timeout-ms-actions`: What to do.
- Ongoing Byte Read:
  - `byte-read-timeout-ms`: How long to wait.
  - `byte-read-timeout-ms-actions`: What to do.

MI.SourceConnectionControl (cont.)

- `max-connection-retries-per-source`: How many retries.
- `resume-from-last-byte-of-previous-source`: Resume or start over on source error.
- `resume-from-last-byte-of-previous-endpoint`: Resume or start over on endpoint error.

MI.Source(ByteRead)?TimeoutActions

- `retries`: How many retries.
- `error-state`: The MEL-enabled error message to return.
- `resume-from-last-byte`: (ByteRead-only) Start over or not.

MI.SourceConnectionRetries

- `max-retries-per-source`: How many times to retry sources.
- `retries-per-endpoint`: How many times to retry this endpoint.

MI.HTTPCodeFailover

More detailed than `failover-errors`.

- `max-reforwards-per-source`: How many reforwards to allow.
- `http-code-failover-actions`: What to do.

MI.HTTPCodeFailoverActions

- `http-codes`: Which codes are considered errors.
- `reforwards`: How to do a reforward.
- `error-state`: The MEL-enabled error message to return.

MI.HTTPCodeReforwards

- `max-reforwards-per-source`: How many source reforwards to allow.
- `reforwards-per-endpoint`: How many endpoint reforwards to allow.

MI.EndpointDetention

- `connection-setup-fail-trigger`: The trigger for connection setup errors.
- `read-timeout-trigger`: The trigger for read timeout errors.
- `http-error-code-trigger`: The trigger for HTTP errors.
- `detention-seconds`: How long to keep it in the naughty corner.

MI.HTTPErrorCodeTrigger

A wrapper around an `EndpointDetentionTrigger`.

- `trigger`: The inner trigger.
- `error-codes`: A list of codes that trigger this.

MI.EndpointDetentionTrigger

A wrapper around `EndpointRepeatingFailures`.

- `trigger-value`: The inner object.

MI.EndpointRepeatingFailures

- `event-count`: How many errors in the window fire the trigger.
- `time-window-millsec`: How long is the window.
- `fail-event-percent-threshold`: What percentage of errors in the window fire the trigger.

MI.SourceDetention

- `detention-full-behavior`: What to when everybody is in the naughty corner.
  - `serve-if-stale-available`: Serve things stale.
  - `synthetic-response`: Create something (see Processing Stages for details).
- `detention-reset-behavior`: If everyone is in the naughty corner, when can they be brought out early.
  - `reset-endpoints`: Remove a specific subset of endpoints.
  - `reset-all-endpoints`: Remove all endpoints.

MI.LoadBalanceMetadata

- `balance-algorithm`: How do we balance. Pick one of:
  - random
  - content-hash
  - ip-hash
- `balance-weights`: Relative weights.
- `balance-path-pattern`: A regex that mutates the path for use with `content-hash`.

MI.HeaderAuth

Shared secret authentication with the origin.

- `header-name`: Name of the header.
- `header-value`: Value to send. (Use Protected Secrets if you like security.)

MI.AWSv4Auth

Use AWSv4 authentication with the origin.

- `access-key-id`
- `secret-access-key`
- `aws-region`
- `aws-service`
- `host-name`

Summary

- Fine-grained control of error conditions upstream.
- Fine-grained control of authentication upstream.
- Plenty of stubs and places that need to be filled out.

CfA

- Is this work the working group wishes to do?
- Is this work ready for the working group?
- Call for Adoption?