draft-ietf-cdni-edge-control-metadata-02

IETF-121

8th November 2024

Slide Title

- Abstract
- Changes in version 02 recap
- Version 03
- Next steps
- Background slides from version 2

Abstract

This specification defines configuration metadata objects related to controlling edge access to resources via content delivery networks (CDNs)

- Configuring Cross-Origin Resource Sharing (CORS) access rules and the dynamic generation of CORS headers is a key feature of typical configurations
- Defining response body compression rules
- Defining client connection timeouts.

Changes in version 02 recap

- Removed references to Open Caching when not needed
- Added references to the latest WHATWG Fetch specification for CORS
- Section 3
  - Separate requirements and object definition
  - Requirements section has been rewritten for more clear concepts on CORS protocol and references to WHATWG Fetch specification
  - Defined valid values of CORS response headers according to the WHATWG Fetch specification for those
- Section 4
  - Removed reference to Metadata Expression Language (MEL)
- Section 5
  - Changed example timeout to 3000 (ms)
- Section 6 is removed
- Added Payload types subsections

Solving some questions

Others Separate examples in a different section

- We have separate examples when there are more than one.
- When there is only one, we have maintained the example in the same section as the object definition, as can be seen in some others CDN-I documents. Would it really be necessary to separate them in that case?

version 03

- Version 03 of the document will be uploaded for:
  - Referencing MI.PatternMath from RFC8006 instead or adding the literal specification of it in the draft, as an outcome of IETF-120
- Any other outcome of IETF-121

Next Steps

- There has not been new comments to the document in the mailing list
  - Version 03 will be uploaded after the meeting with the outcome of this meeting
- Do the group feel we can proceed with WC Last Call?

Version 02 slides backup

Solving some questions recap

Section 3. MI.CrossoriginPolicy

it is unclear if "Origin" refers to the uCDN relative to the dCDN, or whether itrefers to the dCDN relative to the UA. For clarity, please use UA, dCDN,and uCDN to reference the entities, to remove any ambiguity

- When the document talks about Origin, is always referring to an HTTP request header named “Origin”, according to WHATWG Fetch specification. It is not related to the “origin” concept used in CDNs.
- We tried to explain this difference in the new version

Solving some questions recap

Section 3. MI.CrossoriginPolicy

we should not be creating custom pattern matching schemes. We use standard regular expression forms in our other drafts - unless we have a really goodreason not to, we should do the same here

- The definition of “pattern” in this document is the same one used in RFC8006 for MI.PatternMatch (Section 4.1.5). Should we reference it directly?
-
- When the document talks about Origin, is always referring to an HTTP request header named “Origin”, according to WHATWG Fetch specification. It is not related to the “origin” concept used in CDNs.
- We tried to explain this difference in the new version

Solving some questions recap

Section 3. MI.CrossoriginPolicy

section 3.1: are both properties really required? The first example in 3.2 seems redundant? if wildcard-return is true, why include an allow-list? and conversely, if I supply an allow-list with something other than "", does that not already imply wildcard-return is false?*

- More text explaining this is included in the document.
- *allow-origins* declares the list of valid values to match with the value of the received “Origin” header in the request.
- Only in the case that it matches one of the pattern in the list, the dCDN can include CORS response headers, and only in this case, include the Access-Control-Allow-Origin response header.
- If so, the value of this response header can be
  - a wildcard, or
  - to reflect the value of the received Origin header in the request.
- So , “*wildcard*” property only applies as a result of the validation of *allow-origins*

Solving some questions recap

Section 5. MI.ClientConnectionControl

how does this optimize performance? it's not terribly dynamic.

- Different Content Provider's applications can have different restrictions or requirements in terms of timeout for receiving the content.
- Permitting to configure this timeout parameter as part of the metadata permits dCDNs to be more application specific rather than applying a common dCDN global configuration

Solving some questions recap

Section 5. MI.ClientConnectionControl

how does it accommodate device-specific constraints? the metadata is not device-specific?

- Combined with MI.PathMatch (RFC8006) or eventually using MI.ProcessingStages with User-Agent request header this GenericMetadata could be used to define timeouts device-specific

Thank you