Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 13, 2026

A. Jurkovikj

12 May 2026

Semantic Validators for HTTP
draft-jurkovikj-http-semantic-validator-00

Abstract

This document defines the Semantic-ETag HTTP response header field and two associated conditional request header fields. Unlike the standard ETag field, which identifies a selected representation, Semantic-ETag identifies a server-defined semantic equivalence class for the underlying resource state. This enables origin servers to support representation-independent validation and optimistic concurrency control across different representations, such as HTML, JSON, and Markdown views of the same logical resource.

This document does not update or replace the semantics of the ETag header field defined in RFC 9110.

1. Introduction

The Hypertext Transfer Protocol (HTTP) [RFC9110] defines entity tags (ETags) as validators for selected representations. ETags are used for cache validation, conditional requests, and optimistic concurrency control. RFC 9110 distinguishes between strong and weak validators, but both are scoped to representation metadata rather than to an application-defined logical resource state shared across multiple representations.

Many resources are exposed through several representations. For example, a resource might have a browser-facing HTML representation, an API-facing JSON representation, and an editable Markdown representation. A client might read one representation and later

update another. Standard representation validators are not sufficient to express that those representations were derived from the same underlying resource state.

This document defines Semantic-ETag, a response header field that identifies a server-defined semantic equivalence class for the resource state. It also defines If-Semantic-Match and If-Semantic-None-Match for conditional requests against that semantic validator.

Semantic-ETag is intended for origin-server validation. HTTP caches and intermediaries that do not explicitly implement this specification will not use Semantic-ETag for cache revalidation.

2. Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses the Augmented Backus-Naur Form (ABNF) notation of [RFC5234] and the list extension defined in Section 5.6.1 of [RFC9110]. It imports the "opaque-tag" rule from Section 8.8.3 of [RFC9110].

Canonical Resource State (CRS): A server-defined deterministic model of the logical resource state used to decide semantic equivalence. The CRS excludes representation-specific artifacts such as templates or insignificant serialization details, but includes all state that affects the resource’s logical meaning, authorization, or integrity for the relevant equivalence domain.

Semantic equivalence domain: The scope within which a server considers two representations to have the same CRS. The domain might be a single resource across media types, a tenant-scoped resource, an authorization-scoped view, or another server-defined boundary.

Semantic validator: The opaque value carried in Semantic-ETag that identifies the current CRS within a semantic equivalence domain.

3. The Semantic-ETag Header Field

The Semantic-ETag response header field provides the current semantic validator for the selected representation’s resource state.

```
Semantic-ETag = opaque-tag
```

The weak prefix ("W/") MUST NOT be used in Semantic-ETag. A recipient MUST NOT treat a weak entity-tag value as matching a Semantic-ETag.

Semantic-ETag is representation metadata. A server MAY include both ETag and Semantic-ETag in the same response. The two fields are independent: ETag validates the selected representation, while Semantic-ETag validates the server-defined CRS for the applicable semantic equivalence domain.

3.1. Generation and Equivalence

A server that generates Semantic-ETag MUST construct its CRS in a deterministic and reproducible manner for the applicable semantic equivalence domain.

Semantic-ETag values are opaque to clients. A server MAY derive the value from a collision-resistant hash of the CRS, an HMAC, a revision identifier, a version vector, or another server-controlled mechanism, provided that the value changes whenever the CRS changes in a way that matters for semantic equivalence.

If a server provides Semantic-ETag for multiple representations in the same semantic equivalence domain, the opaque string MUST be identical when the server considers their CRS identical, even if their standard ETag values differ.

This specification does not define a universal CRS algorithm that different implementations can use to compute identical validators for the same real-world object. Interoperability is defined at the HTTP field and conditional request level; semantic equivalence is defined by the origin server.

3.2. CRS Construction Guidance

CRS construction is application-specific. Servers are encouraged to use deterministic normalization techniques when the underlying data model has a standard canonical form. For example, if the CRS is represented as JSON, a server can apply the JSON Canonicalization Scheme (JCS) [RFC8785] before deriving a validator.

Servers MUST include all state that affects authorization decisions, integrity, visible logical content, or mutation safety for the semantic equivalence domain. Servers MUST NOT share a Semantic-ETag across representations or authorization contexts that do not have the same CRS.

4. Conditional Requests with Semantic Validators

This document defines two request header fields for conditional requests against Semantic-ETag.

4.1. If-Semantic-Match

The If-Semantic-Match request header field makes the request method conditional on the origin server’s current semantic validator matching one of the provided validators.

If-Semantic-Match = "*" / 1#opaque-tag

If the field value is "*", the condition evaluates to true if the target resource has a current CRS in the applicable semantic equivalence domain, and false otherwise.

If the field value is a list of validators, the condition evaluates to true if the current Semantic-ETag matches one of the provided validators. Otherwise, the condition evaluates to false.

If the condition evaluates to false, the origin server MUST NOT perform the requested method. The origin server MUST respond with a 412 (Precondition Failed) status code.

4.2. If-Semantic-None-Match

The If-Semantic-None-Match request header field makes the request method conditional on the origin server’s current semantic validator not matching any of the provided validators.

If-Semantic-None-Match = "*" / 1#opaque-tag

If the field value is "*", the condition evaluates to false if the target resource has a current CRS in the applicable semantic

equivalence domain, and true otherwise.

If the field value is a list of validators, the condition evaluates to false if the current Semantic-ETag matches one of the provided validators. Otherwise, the condition evaluates to true.

If the condition evaluates to false for a GET or HEAD request, the origin server MUST NOT perform the requested method and MUST respond with a 304 (Not Modified) status code. The origin server SHOULD include the current Semantic-ETag field in the 304 response when one is available.

If the condition evaluates to false for any other request method, the origin server MUST NOT perform the requested method and MUST respond with a 412 (Precondition Failed) status code.

4.3. Evaluation with Standard Preconditions

If a request contains both standard HTTP conditional request header fields and semantic conditional request header fields, an origin server that implements this specification MUST evaluate all applicable preconditions. If any required precondition evaluates to false, the server MUST NOT perform the requested method.

Standard HTTP preconditions retain their meaning and precedence as defined by [RFC9110]. Semantic preconditions are evaluated by the origin server after representation-scoped preconditions that can fail the request have been evaluated, and before the requested method is performed.

When both If-Semantic-Match and If-Semantic-None-Match are present, the origin server MUST evaluate both semantic preconditions. The request proceeds only if both evaluate to true.

If a semantic precondition fails and a standard precondition has not already determined a response status, the response status is:

* 304 (Not Modified), when If-Semantic-None-Match fails for GET or HEAD.

* 412 (Precondition Failed), for other semantic precondition failures.

When returning 412 (Precondition Failed), the origin server SHOULD include the current Semantic-ETag field when one is available. It MAY also include ETag if a selected representation validator is available and useful to the client.

5. Interaction with Caches, Content Negotiation, and Range Requests

Semantic-ETag does not replace ETag and does not define a cache key. Intermediaries MUST NOT use Semantic-ETag as a cache key for stored responses unless they explicitly implement this specification and all relevant HTTP caching requirements.

A cache that does not implement this specification will treat Semantic-ETag as an ordinary extension field. Such a cache will not use If-Semantic-Match or If-Semantic-None-Match for cache revalidation. Clients that require semantic validation therefore SHOULD expect these preconditions to be evaluated by the origin server.

Semantic-ETag does not itself create a Vary dimension. However, servers MUST NOT use the same Semantic-ETag across content negotiation variants, tenants, authorization contexts, or other boundaries unless those variants are in the same semantic

equivalence domain.

Because Semantic-ETag guarantees semantic equivalence but not byte-for-byte identity, it MUST NOT be used to evaluate If-Range preconditions. Servers MUST evaluate If-Range using only validators permitted by [RFC9110] for range requests.

6. Security Considerations

6.1. Overbroad Normalization

If CRS construction omits state that affects authorization, integrity, or mutation safety, a client might incorrectly treat stale or unauthorized state as current. Servers MUST include such state in the CRS or use a narrower semantic equivalence domain.

6.2. Cross-Representation Confusion

If two representations are generated from divergent internal models, they MUST NOT share a Semantic-ETag. Sharing a semantic validator in that case can cause unsafe cross-representation updates.

6.3. Validator Correlation

Identical Semantic-ETag values can reveal that two representations, tenants, or authorization-scoped views share the same underlying CRS. If that correlation is sensitive, servers MUST use separate semantic equivalence domains or privacy-preserving validator construction.

6.4. Validator Guessing and Tracking

Predictable Semantic-ETag values can expose information about resource update frequency or internal revision state. Servers SHOULD use validator construction mechanisms that do not reveal sensitive internal state when validators are exposed to untrusted clients.

6.5. Cache Safety

This specification defines Semantic-ETag as a separate field and does not alter ETag parsing. This separation reduces the risk that legacy caches confuse semantic equivalence with representation identity. Nevertheless, servers SHOULD test deployments with intermediaries that might normalize, remove, or mishandle unknown fields.

7. IANA Considerations

IANA is requested to register the following new field names in the "Hypertext Transfer Protocol (HTTP) Field Name Registry":

Field Name: Semantic-ETag
Status: Permanent
Change Controller: IETF
Reference: This document
Notes: This field uses the opaque-tag syntax and is not a Structured Field.

Field Name: If-Semantic-Match
Status: Permanent
Change Controller: IETF
Reference: This document
Notes: This field uses the opaque-tag syntax and is not a Structured Field.

Field Name: If-Semantic-None-Match
Status: Permanent

Change Controller: IETF
Reference: This document
Notes: This field uses the opaque-tag syntax and is not a Structured Field.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5234] Crocker, D., Ed., and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, June 2022.

8.2. Informative References

[RFC8785] Rundgren, A., Jordan, B., and S. Erdtman, "JSON Canonicalization Scheme (JCS)", RFC 8785, June 2020.

[I-D.jurkovikj-httpapi-agentic-state] Jurkovikj, A., "HTTP Profile for Synchronized Resource State in Agentic Workflows", Work in Progress, Internet-Draft, draft-jurkovikj-httpapi-agentic-state-01.

[I-D.jurkovikj-collab-tunnel] Jurkovikj, A., "The Collaboration Content Transfer (TCT) Protocol", Work in Progress, Internet-Draft, draft-jurkovikj-collab-tunnel-02.

Author’s Address

Antun Jurkovikj
Email: antunjurkovic@gmail.com