HTTP Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 December 2026

F. Martin
Peachymango.org

6 June 2026

HTTP Signaling of Planned IPv4 Unavailability
draft-martin-retry-over-ipv6-00

Abstract

As operators transition services to IPv6-only, planned IPv4 outages help identify remaining dependencies before permanent decommission. Such outages must be measurable, reversible, and understandable to end users. This document defines the 566 (IPv4 Unavailable) HTTP response status code and associated header fields that signal an intentional, often time-bounded IPv4 outage, instruct aware clients to retry over IPv6 after closing the IPv4 connection, and allow clients to confirm successful IPv6 recovery via an optional correlation token so operators can distinguish soft failures from hard failures in centralized logs. The mechanism supports coordinated events (for example, 6/6 IPv6 Day drills), staged enterprise rollouts, and permanent IPv6-only migration. Legacy clients that do not implement this specification treat an unrecognized 566 status code as an internal server error and MAY use the response body for human-readable guidance.

Table of Contents

1. Introduction

1.1. Why Planned IPv4 Outages

IPv6 deployment has been validated through coordinated industry events. On World IPv6 Day (8 June 2011), major content providers enabled IPv6 for 24 hours to measure brokenness in clients, networks, and middleboxes [WORLD-IPV6-DAY]. World IPv6 Launch (6 June 2012) moved many of those sites to permanently enabled IPv6 [WORLD-IPV6-LAUNCH]. Some participants retained IPv6; others reverted toward IPv4-only operation until a later 6/6 commitment. These events tested enabling IPv6; the inverse problem --- identifying what still breaks when IPv4 is intentionally unavailable --- remains under-specified at the application layer.

Operators have adopted time-bounded *planned IPv4 outages* as a complement: deliberately making IPv4 service unavailable for minutes, hours, or days to expose software, protocol, and operational gaps before irreversible decommissioning. The IETF meeting network ran an early example at IETF 71 (Philadelphia, March 2008): an IPv6-only wireless network was available throughout the week, and IPv4 on the main meeting network was disabled for roughly one hour during the Wednesday plenary on 12 March 2008 so attendees could use IPv6-only Internet access and surface client stacks, applications, and services that still depended on IPv4 [IETF71-IPV4-OUTAGE].

Governments are also publishing fixed IPv4 end dates. For example, the Czech Republic approved a plan for state administration services to stop providing IPv4 on *6 June 2032* (6/6/2032) [KONEC-IPV4-CZ]. Operators facing such deadlines need staged transition mechanisms --- including time-bounded planned outages, clear user messaging, and measurable HTTP-layer signals --- long before the final cutover date.

Network-layer IPv4 removal is a poor fit for staged drills:

* Rollback is hard --- routing, ACL, and DNS changes propagate slowly and are error-prone under pressure.
* End users lack context --- a silent timeout looks like a site outage, not an IPv4-path policy.
* Impact is unmeasured --- without an HTTP-visible signal, operators cannot count affected clients or quantify business loss (even a small percentage of requests can be material).

HTTP-layer IPv4 outages address these gaps:

* *Easy rollback* --- disable the 566 policy at the load balancer or origin without waiting for DNS TTL expiry.
* *Advance communication* --- site banners, email, and status pages can reference the same window as IPv4-Unavailable-Until.
* *Clear user messaging* --- a response body explains that IPv4 is intentionally unavailable, when service may resume, and that IPv6 (or contacting an ISP or IT department) is the remedy.
* *Operator metrics* --- count 566 responses and join them with Retry-Over-IPv6-Recovery (and optional tokens) in centralized logs to estimate soft versus hard failure rates.

1.2. Scope

HTTP is not the only application protocol on the Internet. This document addresses HTTP first because it is widely deployed, visible to end users (for example in browsers), and pervasive in enterprise environments for web applications, APIs, and microservices behind load balancers. Other protocols might adopt analogous techniques for planned IPv4 outages; defining those signals is out of scope for this document.

This specification is HTTP-version-agnostic: the status code, header fields, and client behavior apply equally to HTTP/1.1 [RFC9112], HTTP/2 [RFC9113], and HTTP/3 [RFC9114], as they are defined in terms of HTTP semantics [RFC9110]. Wire-format examples use HTTP/1.1 message syntax for readability. Implementations MUST determine whether a request was received over IPv4 at the transport layer (TCP for HTTP/1.1 and HTTP/2, QUIC for HTTP/3), not from the HTTP version alone.

1.3. Technical Motivation

Many operators plan to remove or disable IPv4 while retaining IPv6 service. During migration, maintenance, or decommissioning, a client that connects over IPv4 may observe connection failures or HTTP errors even though the same origin remains available over IPv6.

IPv4-only clients cannot switch address families; they need a clear, loggable explanation that the service no longer supports IPv4 (optionally until a stated time). Dual-stack clients on networks where Happy Eyeballs [RFC8305] selects IPv4 may treat the failure as a general outage unless the server explicitly signals that IPv4 is intentionally unavailable and IPv6 should be used instead.

Application-to-application traffic carried over HTTP (for example REST-style APIs, gRPC, GraphQL, or JSON-RPC) benefits from a machine-readable signal distinct from connectivity failures on other addresses. For example, a gRPC client that tries multiple resolved addresses may surface an error from the first failing attempt, masking the fact that the meaningful signal was returned on an IPv4 connection.

1.4. Requirements Language

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.

1.5. Terminology

This document uses terms from [RFC9110]. Additional terms:

Authority: The host and port derived from the target URI.

Planned IPv4 outage: An operator-initiated period during which IPv4 HTTP service for an authority is intentionally unavailable while IPv6 service is expected to remain available.

Aware client: A client implementation that supports the mechanisms defined in this document.

Legacy client: A client that does not implement this document.

Soft failure: A client receives 566 (or transitional 503 with Retry-Over-IPv6) over IPv4 and subsequently completes the same request successfully over IPv6.

Hard failure: A client receives 566 over IPv4 but cannot successfully complete the request over IPv6.

2. Overview

When IPv4 service is intentionally unavailable for an authority, the responding entity that receives a request over IPv4 sends:

1. *566 (IPv4 Unavailable)*, or during transitional deployments *503 Service Unavailable* with the same header fields --- the IPv4 path is unavailable; the service is not a general outage if IPv6 is expected to work.
2. *Retry-Over-IPv6: ?1* --- the client should retry the same request over IPv6.
3. *IPv4-Unavailable-Until* (optional) --- when IPv4 service may be restored.
4. *Retry-Over-IPv6-Token* (optional, on the IPv4-unavailability response) and *Retry-Over-IPv6-Recovery* (on a successful IPv6 retry) --- optional telemetry so operators can correlate soft failures across load-balanced backends.

Implementations that cannot emit 566 (for example, before the status code is registered or supported by their HTTP stack) MAY send *503 Service Unavailable* instead, with *Retry-Over-IPv6: ?1* and the other response header fields defined in this document. Aware clients treat 503 with Retry-Over-IPv6: ?1 the same as 566 when deciding to retry over IPv6 (see Section 4.1 and Section 7). Operators SHOULD use 566 once their deployment supports it.

The responding entity MUST send 566 (or 503 with Retry-Over-IPv6: ?1 during transitional deployments) only when the request was received over an IPv4 transport connection on the client-facing path (see Section 8).

Clients that do not implement this specification and receive an unrecognized 566 status code MUST treat it as 500 Internal Server Error, as required by Section 15 of [RFC9110]. Operators SHOULD include a response body explaining the IPv4 outage for human readers and for logging by generic HTTP clients.

3. The 566 IPv4 Unavailable Status Code

The 566 (IPv4 Unavailable) status code indicates that the responding entity is intentionally not offering the requested service over IPv4 for this authority, while service over IPv6 is expected to be available. The client SHOULD retry the same request to the same target URI using IPv6 if IPv6 connectivity is available.

This status code applies when the responding entity received the request over IPv4. It MUST NOT be used to indicate general server overload or maintenance that affects all address families (503 Service Unavailable is appropriate for that case). It is generally inappropriate on the IPv4 loopback interface (see Section 8.1).

Intermediaries and caches MUST NOT transform a 566 response into a successful response. Caching of 566 is governed by normal HTTP cache rules [RFC9111]; operators SHOULD send appropriate Cache-Control when responses are generated dynamically based on the client-facing address family.

A 566 response SHOULD include Retry-Over-IPv6 as defined in Section 4.1. It MAY include IPv4-Unavailable-Until, a response body, and Retry-Over-IPv6-Token.

3.1. Status Code Selection

This document registers 566 (IPv4 Unavailable) in the HTTP status code range 512-599, which is currently unassigned. The code number is chosen to align with *6/6 (June 6)*, the date used for coordinated IPv6 deployment events such as World IPv6 Launch, and embeds *66* as a mnemonic for IPv6 within the 5xx server-error class. This mnemonic is for human operability only; protocol behavior does not depend on the numeric value beyond its 5xx class.

3.2. Example

```
HTTP/1.1 566 IPv4 Unavailable
Retry-Over-IPv6: ?1
IPv4-Unavailable-Until: Sun, 07 Jun 2026 00:00:00 GMT
Retry-Over-IPv6-Token: "a1b2c3d4e5f6"
Content-Type: application/problem+json
Content-Length: 0
```

4. Response Header Fields

4.1. Retry-Over-IPv6

The Retry-Over-IPv6 response header field indicates that the client should retry the same request over IPv6.

4.1.1. Syntax

The field value is a Boolean (see [RFC9651]):

```
Retry-Over-IPv6 = "Retry-Over-IPv6" OWS ":" OWS boolean
boolean         = "?0" / "?1"
```

On 566 responses, the value MUST be ?1.

For transitional deployments, 503 Service Unavailable responses MAY include Retry-Over-IPv6: ?1; once 566 is widely supported, operators SHOULD NOT rely on the 503 fallback.

4.1.2. Semantics

When a client receives Retry-Over-IPv6: ?1, it SHOULD retry the same request to the same target URI using IPv6 transport if IPv6 connectivity is available, but only if the response was received on an IPv4 connection. If the client already used IPv6 for that attempt, it MUST NOT retry solely because of this header field.

The header field conveys intent only. It does not guarantee that a retry over IPv6 will succeed.

This header field is a response header field as defined in Section 6.3 of [RFC9110].

4.2. IPv4-Unavailable-Until

The IPv4-Unavailable-Until response header field indicates the time after which IPv4 service for this authority may be restored.

4.2.1. Syntax

```
IPv4-Unavailable-Until = "IPv4-Unavailable-Until" OWS ":"
                         OWS HTTP-date
```

HTTP-date is defined in Section 5.6.7 of [RFC9110].

4.2.2. Semantics

For a permanent IPv6-only transition, this field MAY be omitted; permanence SHOULD be stated in the response body instead.

This field is informational for logging and client caching. It does not mean the client should wait until that time before retrying over IPv6 --- the IPv6 retry SHOULD happen promptly (subject to the client algorithm in Section 7).

IPv4-Unavailable-Until differs from Retry-After [RFC9110]: Retry-After indicates how long to wait before a follow-up request in overload or rate-limit scenarios, while IPv4-Unavailable-Until marks the end of a planned IPv4 unavailability window.

Operators MAY also send Retry-After for legacy clients that do not understand 566 or IPv4-Unavailable-Until.

4.3. Retry-Over-IPv6-Token

The Retry-Over-IPv6-Token response header field carries an opaque token that a client MAY echo on a subsequent successful IPv6 retry so operators can correlate a 566 response with a recovery in centralized logs.

4.3.1. Syntax

```
Retry-Over-IPv6-Token = "Retry-Over-IPv6-Token" OWS ":"
                        OWS quoted-string
```

quoted-string is defined in Section 5.6.4 of [RFC9110].

4.3.2. Semantics

The token is opaque to the client. The client MUST NOT interpret its internal structure.

Tokens SHOULD be short-lived (on the order of minutes, and not extending beyond IPv4-Unavailable-Until when that header is present). Deployments SHOULD use stateless tokens verifiable or loggable by any node in a load-balanced fleet without session affinity to a particular origin server.

This header is RECOMMENDED on 566 responses when operators want pairwise 566-to-recovery correlation across backends.

4.4. Legacy Client Compatibility

Legacy clients that do not implement this document might still benefit from:

* Retry-After with seconds until the outage ends.
* Cache-Control: no-store on dynamically generated outage responses.
* A response body with plain language (see Section 6).

Aware clients MUST prefer 566, Retry-Over-IPv6, and IPv4-Unavailable-Until over inferring outage semantics from the body alone.

5. Request Header Fields

5.1. Retry-Over-IPv6-Recovery

The Retry-Over-IPv6-Recovery request header field allows an aware client to confirm that a successful request over IPv6 is the retry following a 566 response (or transitional 503 with Retry-Over-IPv6: ?1) received over IPv4.

5.1.1. Syntax

```
Retry-Over-IPv6-Recovery = "Retry-Over-IPv6-Recovery" OWS ":"
                           OWS "recovered"
                           *( OWS ";" OWS recovery-param )
recovery-param           = token "=" ( token / quoted-string )
```

The only recovery parameter defined by this document is token, whose value SHOULD be copied from Retry-Over-IPv6-Token on the prior 566 response.

5.1.2. Semantics

The field value recovered means: the responding entity previously returned 566 (or 503 with Retry-Over-IPv6: ?1) on an IPv4 connection for this logical request attempt, and this request is the successful retry over IPv6.

The client MUST send this header on the first successful IPv6 request that follows such a response for the same target URI. The client MUST NOT send it on every subsequent request to the authority.

The client MUST NOT send this header to unrelated origins. The header MUST NOT contain personally identifiable information.

There is no failure variant defined in this document. If the IPv6 connection attempt fails before any HTTP response is received, the client cannot report that failure in-band to the origin during a full IPv4 outage.

5.1.3. Connection Lifecycle

In typical implementations, a client does not keep the IPv4 connection open while also retrying the same request over IPv6. Maintaining both connections in parallel for one logical request increases operational complexity (connection state, cancellation, and handling of duplicate or late responses) and is therefore uncommon.

For this reason, Retry-Over-IPv6-Recovery is carried on the *IPv6* retry request. Operators MUST NOT expect recovery signaling on the IPv4 connection that received 566 (or 503 with Retry-Over-IPv6: ?1).

A typical sequence is:

1. Receive 566 (and optional Retry-Over-IPv6-Token) on IPv4.
2. Close or abandon the IPv4 connection.
3. Open a new connection over IPv6 and retry the same request.
4. On success, include Retry-Over-IPv6-Recovery on that IPv6 request.

5.1.4. Cross-Backend Logging

In load-balanced deployments, the 566 response and the recovery request often reach different origin servers. Correlation is an operator responsibility:

* Log 566 events with Retry-Over-IPv6-Token at the edge, load balancer, or origin.
* Log Retry-Over-IPv6-Recovery (and echoed token) at the same aggregation tier when possible.
* Join events off-box by token across all backend logs.

Operators SHOULD NOT assume that the origin server that emitted 566 will receive the recovery report.

Without tokens, operators MAY compare aggregate 566 counts with aggregate recovery counts over an outage window; this yields ratio estimates only, not per-session pairing.

5.1.5. Interaction with Happy Eyeballs

Implementations using the connection establishment algorithm in [RFC8305] MAY attempt IPv4 and IPv6 connections in parallel, with staggered starts. That specification assumes a destination-address preference that favors IPv6 (Section 2 of [RFC8305]): for example, a *Resolution Delay* before acting on an early A response so an AAAA response can arrive, and interleaving of address families when connection attempts begin. Implementations MAY adapt those delays when local policy differs, and Section 4 of [RFC8305] permits address sorting that reflects measured round-trip times or prior use --- in practice, some stacks therefore make a *best effort to prefer IPv6*, while others under some network conditions will *attempt IPv4 earlier or more often* than a strict IPv6-first policy would suggest.

Happy Eyeballs defines *connection-establishment* success, not application-layer HTTP success. Section 5 of [RFC8305] treats a connection attempt as successful when the transport handshake completes (generally TCP), then *SHOULD cancel* other in-flight connection attempts that have not yet succeeded. Section 9 of [RFC8305] states that Happy Eyeballs handles failures at the TCP/IP layer only; Section 9.2 explicitly notes that the application (for example, TLS or *HTTP*) may not be operational on every resolved address. *RFC 8305 does not specify that an HTTP 5xx response on one connection counts as failure for all parallel connection attempts.* A 566 (or 503 with Retry-Over-IPv6: ?1) is an HTTP response on an already-established connection; handling it --- including whether to retry over the other address family --- is *outside* the Happy Eyeballs connection-race algorithm and is left to the HTTP client or application.

Implications for this document:

* If IPv6 completes the transport handshake and delivers a successful HTTP response first, the client MAY cancel the IPv4 attempt before 566 is received. No Retry-Over-IPv6-Recovery is sent. 566 counts may under-represent total exposure --- this is often the desired outcome during an outage.
* The client MUST send Retry-Over-IPv6-Recovery only if it fully received 566 (or 503 with Retry-Over-IPv6: ?1) on an IPv4 connection for this logical request attempt.
* If IPv6 already succeeded for this logical request attempt at the HTTP layer, the client MUST NOT treat a late or abandoned IPv4 566 as requiring another IPv6 retry or recovery signal --- regardless of how Happy Eyeballs raced the underlying connections.

* An aware client that receives 566 only on IPv4 and has not yet succeeded over IPv6 MUST apply the IPv6 retry requirements in Section 7.2; that behavior is an HTTP-layer extension beyond [RFC8305].

Operators interpreting 566 and recovery metrics during planned outages SHOULD account for Happy Eyeballs transport racing and for the fact that HTTP status codes are not part of the RFC 8305 success definition.

5.1.6. Example

```
GET /api/v1/resource HTTP/1.1
Host: example.com
Retry-Over-IPv6-Recovery: recovered; token="a1b2c3d4e5f6"
```

6. Response Body

Responses with 566 SHOULD include a body explaining the planned IPv4 outage for legacy clients and human readers.

Operators SHOULD make the body as clear as possible for non-technical readers. The body SHOULD NOT assume that the reader understands IPv4, IPv6, or the difference between them. The body SHOULD NOT assume that the reader can resolve the problem themselves (for example, by changing browser or device settings). The body SHOULD briefly explain, in plain language, that the Internet is transitioning to a newer protocol generation (IPv6) and that this service may not be reachable over the older generation (IPv4) on the reader’s network path. The body SHOULD give the reader concrete information they can pass to their Internet service provider (ISP) or organization IT department --- for example, that the site may require IPv6 but their system or network does not appear to support it --- and SHOULD ask them to investigate why IPv6 is not working. When IPv4-Unavailable-Until is present, the body SHOULD state when service over the older connection may resume in plain language.

The following plain-text example is suitable for Content-Type: text/plain or as the text content of an HTML page as Content-Type: text/html:

```
|  This site is not available on your current Internet connection.
|
|  The Internet is moving to a newer protocol generation called IPv6.
|  This service is not reachable over the older generation (IPv4) on
|  your network.  You probably cannot fix this yourself.
|
```

```
|  Contact your Internet provider or your organization's IT help desk
|  and say: "I cannot reach this site --- it may require IPv6, but my
|  system does not seem to work with IPv6."  Ask them why IPv6 is not
|  working for you and whether they can enable it.
|
|  If this is a planned outage, service over the older connection may
|  resume after 7 June 2026, 00:00 UTC.
```

For machine-readable errors, deployments MAY use Problem Details [RFC9457], for example:

```
{
  "type": "about:blank",
  "title": "IPv4 Unavailable",
  "status": 566,
  "detail": "IPv4 unavailable until 2026-06-07T00:00:00Z.",
  "retryOverIPv6": true,
  "ipv4UnavailableUntil": "2026-06-07T00:00:00Z"
}
```

The detail field in Problem Details is primarily for developers and aware clients; deployments SHOULD still provide a separate human-oriented body (plain text or HTML) with the guidance above when the response may be shown to end users.

7. Client Requirements

7.1. Processing 566

When a client receives 566 (or 503 with Retry-Over-IPv6: ?1):

1. If the client knows the response arrived on an IPv4 connection, it SHOULD proceed with an IPv6 retry as below.
2. If the address family is unknown, it MAY retry over IPv6 once.
3. If a successful *HTTP* response for this logical request attempt was already delivered over IPv6 (including when Happy Eyeballs [RFC8305] raced the underlying connections; see Section 5.1.5), the client MUST NOT perform another retry or send Retry-Over-IPv6-Recovery based on a late IPv4 response.

7.2. IPv6 Retry

The client SHOULD close or abandon the IPv4 connection before retrying over IPv6, consistent with the lifecycle described in Section 5.1. The retry MUST use the same method, target URI, and authority. The client SHOULD force address-family selection to IPv6 for this retry. The client MUST NOT change the host, scheme, or port solely because of 566 or Retry-Over-IPv6.

7.3. Idempotent Methods

Aware clients that receive 566 (or transitional 503 with Retry-Over-IPv6: ?1) SHOULD retry the same method, target URI, and body over IPv6 (see Section 7.2). For safe methods [RFC9110], such a retry is generally acceptable. For non-idempotent methods such as POST, the same retry can cause duplicate processing --- for example, a duplicate payment, order, or database insert. Responding entities and operators SHOULD follow the guidance in Section 8.2 on when not to send 566 for such requests.

7.4. Loop Prevention

The client MUST NOT perform more than one IPv4-to-IPv6 retry per logical request attempt triggered by 566 or Retry-Over-IPv6.

After receiving 566, the client SHOULD prefer IPv6 for subsequent connections to the authority until IPv4-Unavailable-Until (if present) or for a default period (for example, 10 minutes).

If the IPv6 retry fails with connectivity errors, the client SHOULD apply backoff before further attempts and SHOULD NOT fall back to IPv4 while IPv4-Unavailable-Until is in the future.

7.5. IPv4-Only Clients

Clients without IPv6 connectivity cannot retry over IPv6. They SHOULD surface IPv4-Unavailable-Until (if present) and the response body to the user or calling application for logging and support tickets.

7.6. Recovery Signaling

On the first successful IPv6 request following a fully received 566 over IPv4, the client SHOULD send Retry-Over-IPv6-Recovery: recovered and SHOULD echo Retry-Over-IPv6-Token in the token parameter when a token was provided.

7.7. NAT64 and Translation

Clients on translated IPv4 paths (for example NAT64/464XLAT) might not be able to initiate a native IPv6 retry even when dual-stack is reported at the API layer. Implementations SHOULD present the response body explanation to the user; operators SHOULD not assume all "IPv4" clients can switch address families.

8. Server and Operational Considerations

8.1. When to Send 566

The responding entity SHOULD send 566 when:

* IPv4 HTTP service for the authority is intentionally unavailable;
* IPv6 service for the requested resource is expected to be available; and
* The request was received over IPv4 on the client-facing path; and
* For non-idempotent methods, duplicate processing of an IPv6 retry is acceptable or prevented (see Section 8.2).

The responding entity MAY omit 566 (and the transitional 503 with Retry-Over-IPv6) for requests received on the IPv4 loopback interface --- for example, when the client-facing connection uses addresses in 127.0.0.0/8 such as 127.0.0.1. Routable IPv4 service may be disabled during a planned outage while loopback remains available for local health checks, monitoring, and administration; those clients do not need a signal to retry over IPv6.

Operators MAY run staged rollouts: short canary outages (for example, one minute), longer windows (hours or a full day aligned with 6/6), and eventually permanent IPv6-only service.

8.2. Idempotent Methods and Duplicate Processing

As described in Section 7.3, aware clients SHOULD retry after 566, including for non-idempotent methods --- which can cause duplicate processing. Clients cannot generally determine whether a given application or resource tolerates duplicate processing. Responding entities MUST NOT assume that end-user clients will suppress IPv6 retries for non-idempotent methods.

A 566 response does *not* guarantee that the first request had no effect. Duplicate risk arises when:

* *566 is generated at an edge or load balancer* while an origin server already started or completed processing the request on the IPv4 path.
* *Policy races during rollout* --- IPv4-unavailability policy may be enabled or disabled while requests are in flight.
* *Late IPv4 responses versus an IPv6 retry* --- when Happy Eyeballs [RFC8305] or a prior IPv6 attempt is in play, a client may retry or complete work without deduplication at the application layer (see Section 5.1.5).

Because of this uncertainty, the responding entity SHOULD NOT send 566 (or 503 with Retry-Over-IPv6: ?1) for non-idempotent methods such as POST when an IPv6 retry of the same request would cause

unacceptable duplicate side effects, unless the application provides deduplication (for example, an idempotency key), a request identifier, or another mechanism that makes the retry safe. Where duplicate processing is unacceptable and no such mechanism exists, *omitting 566 MAY be preferable* to signaling a retry the client cannot safely evaluate.

Operators SHOULD prefer applying 566 to idempotent methods during outage tests. APIs that must remain available for non-idempotent methods through a planned IPv4 outage SHOULD document and implement application-level deduplication or other safe-retry semantics explicitly.

8.3. Measuring Outage Impact

Operators SHOULD instrument at the edge or load balancer, aggregating all backends:

Metric	Source
566 count	566 responses logged with optional token
Recovery count	Requests carrying Retry-Over-IPv6-Recovery
Paired recoveries	Off-box join on matching token values
Unrecovered 566	566 count - paired recoveries (estimated hard fail and legacy clients)

Table 1

Hard-failure counts are estimates: clients with no IPv6 path cannot send recovery signals in-band.

The responding entity SHOULD log recovery headers but MUST NOT alter the response based on them.

8.4. CDN and Reverse Proxy Deployment

When an edge terminates client IPv4 and connects to an origin over IPv6, the *edge* sends 566 to the client when IPv4 to the edge is disabled --- not necessarily the origin application. The entity that generates 566 MUST know the client-facing address family.

8.5. Token Generation

Token format and validation are deployment-specific. Tokens SHOULD be unguessable, short-lived, and loggable without affinity to the issuing server.

8.6. Transitional Fallback

Deployments that cannot emit 566 MAY use 503 Service Unavailable with Retry-Over-IPv6: ?1 and IPv4-Unavailable-Until until 566 support is available.

9. Application Protocol Considerations

This section is informative.

9.1. HTTP Versions

No change to the on-the-wire status code or header field definitions is required across HTTP versions. Deployment considerations differ mainly in how connections are managed:

* *HTTP/1.1* --- A 566 response typically applies to one request on a single TCP connection. The client closes that IPv4 connection before retrying over IPv6, as described in Section 5.1.3.
* *HTTP/2* --- 566 is a connection-level signal for that TCP connection. A client SHOULD close the IPv4 HTTP/2 connection (affecting all streams on it) before opening an IPv6 connection for the retry. Servers SHOULD emit 566 on every IPv4 HTTP/2 connection that receives a request during an outage, not only on the first stream.
* *HTTP/3* --- The same semantics apply on a QUIC connection to the authority. HTTP/3 is a separate transport from HTTP/1.1 or HTTP/2 over TCP; a client MAY hold concurrent connections of different HTTP versions and address families. A 566 received on an IPv4 QUIC connection does not automatically invalidate an existing IPv6 HTTP/3 connection, but the client MUST still apply Section 7.2 when the logical request attempt that received 566 has not yet succeeded over IPv6.

Clients that discover HTTP/3 via Alt-Svc or similar mechanisms on an IPv4 connection still need to evaluate 566 and Retry-Over-IPv6 before treating the request as a general failure. Operators SHOULD configure load balancers and origins to emit the same signaling on all HTTP versions they expose.

9.2. gRPC and Other HTTP APIs

gRPC maps HTTP 566 to UNAVAILABLE, the same as 503. gRPC implementations SHOULD inspect Retry-Over-IPv6 on the HTTP response before aggregating multi-address connection errors, so that an IPv4 policy signal is not confused with IPv6 connectivity failure.

Suggested error text for logs: "IPv4 unavailable until <date>; retry over IPv6."

Retry policies SHOULD retry over IPv6 when Retry-Over-IPv6: ?1 is present, not blindly retry the same address list.

10. Deployment Models

This section compares HTTP-layer signaling with other transition techniques.

Method	Limitation for staged outages
DNS-only (withdraw A records)	Hard rollback; poor application errors; difficult time-bounded windows
Network ACL or routing	Complex rollback; timeouts instead of policy signals; weak metrics
Happy Eyeballs alone [RFC8305]	Implicit; may misattribute IPv4 policy as IPv6 brokenness
Site banner only	Applications and APIs do not see banners; no automatic IPv6 retry
HTTP 566 + headers (this document)	Reversible at LB; structured retry; measurable soft/hard fail

Table 2

HTTP-layer signaling complements DNS and network changes, especially when A records remain or when the client already connected over IPv4.

11. Security Considerations

An attacker who can inject or modify HTTP responses could attempt to influence client connection behavior by adding Retry-Over-IPv6 or related header fields. Implementations SHOULD only honor these fields on authenticated transport connections to the intended authority.

Misuse could cause clients to prefer IPv6 paths that are slower, unavailable, or subject to different policy than the original IPv4 path. Operators SHOULD monitor IPv6 reachability before signaling clients to retry over IPv6.

Recovery headers and tokens are operational telemetry, not authentication. Deployments SHOULD rate-limit and treat forged recovery signals as untrusted hints.

The token carries no meaning to the client. An operator MAY nonetheless generate tokens that the operator can validate when processing logs, so that random or forged recovery reports can be discarded. For example, a deployment might combine a site identifier (such as the authority’s domain name) with a random nonce and protect the value with a keyed authenticator (such as an HMAC) using a secret shared across the load-balanced fleet. Such validation is for operational filtering only; clients MUST NOT interpret token structure, and token validation does not authenticate the client or the recovery signal.

566 responses that depend on the client-facing address family SHOULD use Cache-Control: private, no-store when appropriate to avoid cache poisoning.

This mechanism does not by itself provide confidentiality or integrity for retried requests. Any security properties depend on the underlying transport and application protocol in use.

12. IANA Considerations

IANA is requested to make the following registrations.

12.1. HTTP Status Code

In the "Hypertext Transfer Protocol (HTTP) Status Code Registry" (https://www.iana.org/assignments/http-status-codes/ (https://www.iana.org/assignments/http-status-codes/)):

Value	Description	Reference
566	IPv4 Unavailable	This document

Table 3

12.2. HTTP Field Names

In the "Hypertext Transfer Protocol (HTTP) Field Name Registry" (https://www.iana.org/assignments/http-fields/ (https://www.iana.org/assignments/http-fields/)):

Field Name	Status	Struct	Reference
Retry-Over-IPv6	permanent	-	This document
IPv4-Unavailable-Until	permanent	-	This document
Retry-Over-IPv6-Token	permanent	-	This document
Retry-Over-IPv6-Recovery	permanent	-	This document

Table 4

13. Examples

This section is informative.

13.1. Dual-Stack Browser

A browser receives:

```
HTTP/1.1 566 IPv4 Unavailable
Retry-Over-IPv6: ?1
IPv4-Unavailable-Until: Sun, 07 Jun 2026 00:00:00 GMT
Content-Length: 0
```

It closes the IPv4 connection, retries over IPv6, and completes the page load without displaying an error page.

13.2. Legacy Browser with HTML Body

```
HTTP/1.1 566 IPv4 Unavailable
Retry-After: 86400
Content-Type: text/html; charset=utf-8

<html><body><p>This site is not available on your current
Internet connection.</p><p>The Internet is moving to a newer protocol
generation called IPv6. This service is not reachable over the older
generation (IPv4) on your network. You probably cannot fix this
yourself.</p><p>Contact your Internet provider or your organization’s
IT help desk and say:
&amp;quot;I cannot reach this site --- it may require IPv6, but my system
does not seem to work with IPv6.&amp;quot; Ask them
why IPv6 is not working for you and whether they can enable it.</p>
<p>If this is a planned outage, service over the older connection may
resume after 7 June 2026, 00:00 UTC.</p></body></html>
```

13.3. Cross-Backend Recovery

Backend A (IPv4 path) returns:

```
HTTP/1.1 566 IPv4 Unavailable
Retry-Over-IPv6: ?1
Retry-Over-IPv6-Token: "abc123"
Content-Length: 0
```

The client retries over IPv6; backend B receives:

```
GET /index.html HTTP/1.1
Host: example.com
Retry-Over-IPv6-Recovery: recovered; token="abc123"
```

An edge log pipeline joins both events on token=abc123.

14. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8305] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency", RFC 8305, DOI 10.17487/RFC8305, December 2017, <https://www.rfc-editor.org/info/rfc8305>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[RFC9112] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP/1.1", STD 99, RFC 9112, DOI 10.17487/RFC9112, June 2022, <https://www.rfc-editor.org/info/rfc9112>.

[RFC9113] Thomson, M., Ed. and C. Benfield, Ed., "HTTP/2", RFC 9113, DOI 10.17487/RFC9113, June 2022, <https://www.rfc-editor.org/info/rfc9113>.

[RFC9114] Bishop, M., Ed., "HTTP/3", RFC 9114, DOI 10.17487/RFC9114, June 2022, <https://www.rfc-editor.org/info/rfc9114>.

[RFC9651] Nottingham, M. and P. Kamp, "Structured Field Values for HTTP", RFC 9651, DOI 10.17487/RFC9651, September 2024, <https://www.rfc-editor.org/info/rfc9651>.

15. Informative References

[IETF71-IPV4-OUTAGE]
Internet Society, "IETF 71 IPv4 Outage", 2008, <https://web.archive.org/web/20111016062408/wiki.tools.isoc.org/IETF71_IPv4_Outage>.

[KONEC-IPV4-CZ]
CZ.NIC, "Konec IPv4 - Czech Republic IPv4 End Date", <https://konecipv4.cz/en/>.

[RFC9111] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Caching", STD 98, RFC 9111, DOI 10.17487/RFC9111, June 2022, <https://www.rfc-editor.org/info/rfc9111>.

[RFC9457] Nottingham, M., Wilde, E., and S. Dalal, "Problem Details for HTTP APIs", RFC 9457, DOI 10.17487/RFC9457, July 2023, <https://www.rfc-editor.org/info/rfc9457>.

[WORLD-IPV6-DAY]
Internet Society, "World IPv6 Launch FAQ", <https://www.worldipv6launch.org/faq/>.

[WORLD-IPV6-LAUNCH]
Internet Society, "World IPv6 Launch", <https://www.worldipv6launch.org/>.

Author’s Address

Franck Martin
Peachymango.org
Email: franck@peachymango.org