Workgroup: HTTP
Internet-Draft:
draft-ietf-httpbis-proxy-status-03
Published: 9 February 2021
Intended Status: Standards Track
Expires: 13 August 2021
Authors: M. Nottingham
Fastly

P. Sikora
Google

The Proxy-Status HTTP Response Header Field

Abstract

This document defines the Proxy-Status HTTP field to convey the details of intermediary response handling, including generated errors.

Note to Readers

RFC EDITOR: please remove this section before publication

Discussion of this draft takes place on the HTTP working group mailing list (ietf-http-wg@w3.org), which is archived at https://lists.w3.org/Archives/Public/ietf-http-wg/.

Working Group information can be found at https://httpwg.org/; source code and issues list for this draft can be found at https://github.com/httpwg/http-extensions/labels/proxy-status.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 13 August 2021.

Table of Contents

1. Introduction

HTTP intermediaries -- including both forward proxies and gateways (also known as "reverse proxies") -- have become an increasingly significant part of HTTP deployments. In particular, reverse proxies and Content Delivery Networks (CDNs) form part of the critical infrastructure of many Web sites.

Typically, HTTP intermediaries forward requests towards the origin server and then forward their responses back to clients. However, if an error occurs before a response is obtained from upstream, the response is often generated by the intermediary itself.

HTTP accommodates these types of errors with a few status codes; for example, 502 Bad Gateway and 504 Gateway Timeout. However, experience has shown that more information is necessary to aid debugging and communicate what's happened to the client. Additionally, intermediaries sometimes want to convey additional information about their handling of a response, even if they did not generate it.

To enable these uses, Section 2 defines a new HTTP response field to allow intermediaries to convey details of their handling of a response, Section 2.1 enumerates the kind of information that can be conveyed, and Section 2.4 defines a set of error types for use when a proxy encounters an issue when obtaining a response for the request.

1.1. Notational Conventions

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 specification uses Structured Fields [I-D.ietf-httpbis-header-structure] to specify syntax and parsing. The terms sf-list, sf-item, sf-string, sf-token, sf-integer and key refer to the structured types defined therein.

Note that in this specification, "proxy" is used to indicate both forward and reverse proxies, otherwise known as gateways. "Next hop" indicates the connection in the direction leading to the origin server for the request.

2. The Proxy-Status HTTP Field

The Proxy-Status HTTP response field allows an intermediary to convey additional information about its handling of a response and its associated request.

It is a List [I-D.ietf-httpbis-header-structure], Section 3.1:

```
Proxy-Status   = sf-list
```

Each member of the list represents an intermediary that has handled the response. The first member of the list represents the intermediary closest to the origin server, and the last member of the list represents the intermediary closest to the user agent.

For example:

```
Proxy-Status: FooProxy, ExampleCDN
```

indicates that this response was handled first by FooProxy and then ExampleCDN.

Intermediaries determine when it is appropriate to add the Proxy-Status field to a response. Some might decide to append to it to all responses, whereas others might only do so when specifically configured to, or when the request contains a header that activates a debugging mode.

Each member of the list identifies the intermediary that inserted the value, and MUST have a type of either sf-string or sf-token. Depending on the deployment, this might be a product or service name (e.g., ExampleProxy or "Example CDN"), a hostname ("proxy-3.example.com"), an IP address, or a generated string.

Parameters on each member convey additional information about that intermediary's handling of the response and its associated request; see Section 2.1. While all of these parameters are OPTIONAL, intermediaries are encouraged to provide as much information as possible (but see Section 4 for security considerations in doing so).

When adding a value to the Proxy-Status field, intermediaries SHOULD preserve the existing members of the field, to allow debugging of the entire chain of intermediaries handling the request.

Proxy-Status MAY be sent in HTTP trailers. For example, if an intermediary is streaming a response and the upstream connection suddenly terminates, Proxy-Status can only be appended to the trailers of the outgoing message, since the headers have already been sent. As with all trailers, it might be silently discarded along the path to the user agent, so this SHOULD NOT be done unless it is not possible to send it in headers, and an intermediary MUST NOT send Proxy-Status as a trailer field unless it has also sent a corresponding Proxy-Status header field in the same message, so that the trailer value's ordering relative to other intermediaries is preserved.

Origin servers MUST NOT generate the Proxy-Status field.

2.1. Proxy-Status Parameters

This section lists parameters that can be used on the members of the Proxy-Status field. Unrecognised parameters SHOULD be ignored.

2.1.1. error

The error parameter's value is an sf-token that is a Proxy Error Type. When present, it indicates that the proxy encountered an issue when obtaining a response.

Unless a Proxy Error Type specifies otherwise, the presences of error often, but not always, indicates that response was generated by the proxy, not the origin server or any other upstream server. For example, a proxy might attempt to correct an error, or part of a response might be forwarded before the error is encountered.

Section 2.4 lists the Proxy Error Types defined in this document; new ones can be defined using the procedure outlined in Section 2.5.

For example:

```
HTTP/1.1 504 Gateway Timeout
Proxy-Status: SomeCDN; error=connection_timeout
```

indicates that this 504 response was generated by SomeCDN, due to a connection timeout when going forward.

Or:

```
HTTP/1.1 429 Too Many Requests
Proxy-Status: SomeReverseProxy; error=http_request_error
```

indicates that this 429 Too Many Requests response was generated by the intermediary, not the origin.

When sending the error parameter, the most specific Proxy Error Type SHOULD be sent, provided that it accurately represents the error condition. If an appropriate Proxy Error Type is not defined, there are a number of generic error types (e.g., proxy_internal_error, http_protocol_error) that can be used. If they are not suitable, consider registering a new Proxy Error Type (see Section 2.5).

Each Proxy Error Type has a Recommended HTTP Status Code. When generating a HTTP response containing error, its HTTP status code SHOULD be set to the Recommended HTTP Status Code. However, there may be circumstances (e.g., for backwards compatibility with previous behaviours, a status code has already been sent) when another status code might be used.

Proxy Error Types can also define any number of extra parameters for use with that type. Their use, like all parameters, is optional. As a result, if an extra parameter is used with a Proxy Error Type for which it is not defined, it will be ignored.

2.1.2. next-hop

The next-hop parameter's value is an sf-string or sf-token that identifies the intermediary or origin server selected (and used, if contacted) for this response. It might be a hostname, IP address, or alias.

For example:

```
Proxy-Status: cdn.example.org; next-hop=backend.example.org
```

2.1.3. next-protocol

The next-protocol parameter's value indicates the ALPN protocol identifier [RFC7301] used by the intermediary to connect to the next hop. This is only applicable when that connection was actually established.

The value MUST be either an sf-token or sf-binary. If the protocol identifier is able to be expressed as an sf-token using UTF-8 encoding, that form MUST be used.

For example:

```
Proxy-Status: "proxy.example.org"; next-protocol=h2
```

2.2. received-status

The received-status parameter's value indicates the HTTP status code that the intermediary received from the next hop server.

The value MUST be an sf-integer.

For example:

```
Proxy-Status: ExampleProxy; received-status=200
```

2.2.1. details

The details parameter's value is an sf-string containing additional information not captured anywhere else. This can include implementation-specific or deployment-specific information.

For example:

```
Proxy-Status: ExampleProxy; error="http_protocol_error";
              details="Malformed response header - space before colon"
```

2.3. Defining New Proxy-Status Parameters

New Proxy-Status Parameters can be defined by registering them in the HTTP Proxy-Status Parameters registry.

Registration requests are reviewed and approved by a Designated Expert, as per [RFC8126], Section 4.5. A specification document is appreciated, but not required.

The Expert(s) should consider the following factors when evaluating requests:

- Community feedback
- If the value is sufficiently well-defined
- Generic parameters are preferred over vendor-specific, application-specific or deployment-specific values. If a generic value cannot be agreed upon in the community, the parameter's name should be correspondingly specific (e.g., with a prefix that identifies the vendor, application or deployment).
- Parameter names should not conflict with registered extra parameters in the Proxy Error Type Registry.

Registration requests should use the following template:

- Name: [a name for the Proxy-Status Parameter that matches key]

*Description: [a description of the parameter semantics and value]

*Reference: [to a specification defining this parameter]

See the registry at https://iana.org/assignments/http-proxy-status for details on where to send registration requests.

2.4. Proxy Error Types

This section lists the Proxy Error Types defined by this document. See Section 2.5 for information about defining new Proxy Error Types.

2.4.1. DNS Timeout

*Name: dns_timeout

*Description: The intermediary encountered a timeout when trying to find an IP address for the next hop hostname.

*Extra Parameters: None.

*Recommended HTTP status code: 504

2.4.2. DNS Error

*Name: dns_error

*Description: The intermediary encountered a DNS error when trying to find an IP address for the next hop hostname.

*Extra Parameters:

-rcode: A sf-string conveying the DNS RCODE that indicates the error type. See [RFC8499], Section 3.

*Recommended HTTP status code: 502

2.4.3. Destination Not Found

*Name: destination_not_found

*Description: The intermediary cannot determine the appropriate next hop to use for this request; for example, it may not be configured. Note that this error is specific to gateways, which typically require specific configuration to identify the "backend" server; forward proxies use in-band information to identify the origin server.

*Extra Parameters: None.

*Recommended HTTP status code: 500

2.4.4. Destination Unavailable

*Name: destination_unavailable

*Description: The intermediary considers the next hop to be unavailable; e.g., recent attempts to communicate with it may have failed, or a health check may indicate that it is down.

*Extra Parameters: None.

*Recommended HTTP status code: 503

2.4.5. Destination IP Prohibited

*Name: destination_ip_prohibited

*Description: The intermediary is configured to prohibit connections to the next hop IP address.

*Extra Parameters: None.

*Recommended HTTP status code: 502

2.4.6. Destination IP Unroutable

*Name: destination_ip_unroutable

*Description: The intermediary cannot find a route to the next hop IP address.

*Extra Parameters: None.

*Recommended HTTP status code: 502

2.4.7. Connection Refused

*Name: connection_refused

*Description: The intermediary's connection to the next hop was refused.

*Extra Parameters: None.

*Recommended HTTP status code: 502

2.4.8. Connection Terminated

*Name: connection_terminated

*Description: The intermediary's connection to the next hop was closed before complete response was received.

*Extra Parameters: None.

*Recommended HTTP status code: 502

*Notes: Responses with this error type might not have been generated by the intermediary.

2.4.9. Connection Timeout

*Name: connection_timeout

*Description: The intermediary's attempt to open a connection to the next hop timed out.

*Extra Parameters: None.

*Recommended HTTP status code: 504

2.4.10. Connection Read Timeout

*Name: connection_read_timeout

*Description: The intermediary was expecting data on a connection (e.g., part of a response), but did not receive any new data in a configured time limit.

*Extra Parameters: None.

*Recommended HTTP status code: 504

*Notes: Responses with this error type might not have been generated by the intermediary.

2.4.11. Connection Write Timeout

*Name: connection_write_timeout

*Description: The intermediary was attempting to write data to a connection, but was not able to (e.g., because its buffers were full).

*Extra Parameters: None.

*Recommended HTTP status code: 504

*Notes: Responses with this error type might not have been generated by the intermediary.

2.4.12. Connection Limit Reached

*Name: connection_limit_reached

*Description: The intermediary is configured to limit the number of connections it has to the next hop, and that limit has been passed.

*Extra Parameters: None.

*Recommended HTTP status code: 503

2.4.13. TLS Protocol Error

*Name: tls_protocol_error

*Description: The intermediary encountered a TLS error when communicating with the next hop, either during handshake or afterwards.

*Extra Parameters: None.

*Recommended HTTP status code: 502

*Notes: Responses with this error type might not have been generated by the intermediary.

Note that additional information about the error can be recorded in the details parameter (as is the case for all errors).

2.4.14. TLS Certificate Error

*Name: tls_certificate_error

*Description: The intermediary encountered an error when verifying the certificate presented by the next hop.

*Extra Parameters: None.

*Recommended HTTP status code: 502

Note that additional information about the error can be recorded in the details parameter (as is the case for all errors).

2.4.15. TLS Alert Received

*Name: tls_alert_received

*Description: The intermediary received a TLS alert from the next hop.

*Extra Parameters:

  -alert-message: an sf-token containing the applicable description string from the TLS Alerts registry.

*Recommended HTTP status code: 502

*Notes: Responses with this error type might not have been generated by the intermediary.

2.4.16. HTTP Request Error

*Name: http_request_error

*Description: The intermediary is generating a client (4xx) response on the origin's behalf. Applicable status codes include (but are not limited to) 400, 403, 405, 406, 408, 411, 413, 414, 415, 416, 417, 429.

*Extra Parameters:

  -status-code: an sf-integer containing the generated status code.

  -status-phrase: an sf-string containing the generated status phrase.

*Recommended HTTP status code: The applicable 4xx status code

*Notes: This type helps distinguish between responses generated by intermediaries from those generated by the origin.

2.4.17. HTTP Request Denied

*Name: http_request_denied

*Description: The intermediary rejected the HTTP request based on its configuration and/or policy settings. The request wasn't forwarded to the next hop.

*Extra Parameters: None.

*Recommended HTTP status code: 403

2.4.18. HTTP Incomplete Response

*Name: http_response_incomplete

*Description: The intermediary received an incomplete response to the request from the next hop.

*Extra Parameters: None.

*Recommended HTTP status code: 502

*Notes: Responses with this error type might not have been generated by the intermediary.

2.4.19. HTTP Response Header Section Too Large

*Name: http_response_header_section_size

*Description: The intermediary received a response to the request whose header section was considered too large.

*Extra Parameters:

- -header-section-size: an sf-integer indicating how large the headers received were. Note that they might not be complete; i.e., the intermediary may have discarded or refused additional data.

*Recommended HTTP status code: 502

*Notes: Responses with this error type might not have been generated by the intermediary.

2.4.20. HTTP Response Header Too Large

*Name: http_response_header_size

*Description: The intermediary received a response to the request containing an individual header line that was considered too large.

*Extra Parameters:

- -header-name: an sf-string indicating the name of the header that triggered the error.

- -header-size: an sf-integer indicating the size of the header that triggered the error.

*Recommended HTTP status code: 502

*Notes: Responses with this error type might not have been generated by the intermediary.

2.4.21. HTTP Response Body Too Large

*Name: http_response_body_size

*Description: The intermediary received a response to the request whose body was considered too large.

*Extra Parameters:

  -body-size: an sf-integer indicating how large the body received was. Note that it may not have been complete; i.e., the intermediary may have discarded or refused additional data.

*Recommended HTTP status code: 502

*Notes: Responses with this error type might not have been generated by the intermediary.

2.4.22. HTTP Response Trailer Section Too Large

*Name: http_response_trailer_section_size

*Description: The intermediary received a response to the request whose trailer section was considered too large.

*Extra Parameters:

  -trailer-section-size: an sf-integer indicating how large the trailers received were. Note that they might not be complete; i.e., the intermediary may have discarded or refused additional data.

*Recommended HTTP status code: 502

*Notes: Responses with this error type might not have been generated by the intermediary.

2.4.23. HTTP Response Trailer Too Large

*Name: http_response_trailer_size

*Description: The intermediary received a response to the request containing an individual trailer line that was considered too large.

*Extra Parameters:

  -trailer-name: an sf-string indicating the name of the trailer that triggered the error.

  -trailer-size: an sf-integer indicating the size of the trailer that triggered the error.

*Recommended HTTP status code: 502

*Notes: Responses with this error type might not have been generated by the intermediary.

2.4.24. HTTP Response Transfer-Coding Error

*Name: http_response_transfer_coding

*Description: The intermediary encountered an error decoding the transfer-coding of the response.

*Extra Parameters:

  -coding: an sf-token containing the specific coding that caused the error.

*Recommended HTTP status code: 502

*Notes: Responses with this error type might not have been generated by the intermediary.

2.4.25. HTTP Response Content-Coding Error

*Name: http_response_content_coding

*Description: The intermediary encountered an error decoding the content-coding of the response.

*Extra Parameters:

  -coding: an sf-token containing the specific coding that caused the error.

*Recommended HTTP status code: 502

*Notes: Responses with this error type might not have been generated by the intermediary.

2.4.26. HTTP Response Timeout

*Name: http_response_timeout

*Description: The intermediary reached a configured time limit waiting for the complete response.

*Extra Parameters: None.

*Recommended HTTP status code: 504

*Notes: Responses with this error type might not have been generated by the intermediary.

2.4.27. HTTP Upgrade Failed

*Name: http_upgrade_failed

*Description: The HTTP Upgrade between the intermediary and the next hop failed.

*Extra Parameters: None.

*Recommended HTTP status code: 502

2.4.28. HTTP Protocol Error

*Name: http_protocol_error

*Description: The intermediary encountered a HTTP protocol error when communicating with the next hop. This error should only be used when a more specific one is not defined.

*Extra Parameters: None.

*Recommended HTTP status code: 502

*Notes: Responses with this error type might not have been generated by the intermediary.

Note that additional information about the error can be recorded in the details parameter (as is the case for all errors).

2.4.29. Proxy Internal Response

*Name: proxy_internal_response

*Description: The intermediary generated the response locally, without attempting to connect to the next hop (e.g. in response to a request to a debug endpoint terminated at the intermediary).

*Extra Parameters: None.

*Recommended HTTP status code:

2.4.30. Proxy Internal Error

*Name: proxy_internal_error

*Description: The intermediary encountered an internal error unrelated to the origin.

*Extra Parameters: None

*Recommended HTTP status code: 500

Note that additional information about the error can be recorded in the details parameter (as is the case for all errors).

2.4.31. Proxy Configuration Error

*Name: proxy_configuration_error

*Description: The intermediary encountered an error regarding its configuration.

*Extra Parameters: None

*Recommended HTTP status code: 500

Note that additional information about the error can be recorded in the details parameter (as is the case for all errors).

2.4.32. Proxy Loop Detected

*Name: proxy_loop_detected

*Description: The intermediary tried to forward the request to itself, or a loop has been detected using different means (e.g. [RFC8586]).

*Extra Parameters: None.

*Recommended HTTP status code: 502

2.5. Defining New Proxy Error Types

New Proxy Error Types can be defined by registering them in the HTTP Proxy Error Types registry.

Registration requests are reviewed and approved by a Designated Expert, as per [RFC8126], Section 4.5. A specification document is appreciated, but not required.

The Expert(s) should consider the following factors when evaluating requests:

*Community feedback

*If the value is sufficiently well-defined

*Generic types are preferred over vendor-specific, application-specific or deployment-specific values. If a generic value cannot

be agreed upon in the community, the types's name should be correspondingly specific (e.g., with a prefix that identifies the vendor, application or deployment).

* Extra Parameters should not conflict with registered Proxy-Status parameters.

Registration requests should use the following template:

* Name: [a name for the Proxy Error Type that matches sf-token]

* Description: [a description of the conditions that generate the Proxy Error Type]

* Extra Parameters: [zero or more optional parameters, along with their allowable type(s)]

* Recommended HTTP status code: [the appropriate HTTP status code for this entry]

* Notes: [optional]

If the Proxy Error Type might occur in responses that are not generated by the intermediary -- for example, when the error is detected during response content processing and a Proxy-Status trailer field is appended -- that SHOULD be explained in the Notes.

See the registry at https://iana.org/assignments/http-proxy-status for details on where to send registration requests.

3. IANA Considerations

Upon publication, please create the HTTP Proxy-Status Parameters registry and the HTTP Proxy Error Types registry at https://iana.org/assignments/http-proxy-statuses and populate them with the types defined in Section 2.1 and Section 2.4 respectively; see Section 2.3 and Section 2.5 for its associated procedures.

4. Security Considerations

One of the primary security concerns when using Proxy-Status is leaking information that might aid an attacker. For example, information about the intermediary's configuration and back-end topology can be exposed.

As a result, care needs to be taken when deciding to generate a Proxy-Status field. Note that intermediaries are not required to generate a Proxy-Status field in any response, and can conditionally generate them based upon request attributes (e.g., authentication tokens, IP address).

Likewise, generation of all parameters is optional.

5. References

5.1. 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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <https://www.rfc-editor.org/info/rfc8499>.

[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>.

[I-D.ietf-httpbis-header-structure] Nottingham, M. and P. Kamp, "Structured Field Values for HTTP", Work in Progress, Internet-Draft, draft-ietf-httpbis-header-structure-19, 3 June 2020, <http://www.ietf.org/internet-drafts/draft-ietf-httpbis-header-structure-19.txt>.

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/info/rfc7301>.

5.2. Informative References

[RFC8586] Ludin, S., Nottingham, M., and N. Sullivan, "Loop Detection in Content Delivery Networks (CDNs)", RFC 8586, DOI 10.17487/RFC8586, April 2019, <https://www.rfc-editor.org/info/rfc8586>.

Authors' Addresses

Mark Nottingham
Fastly
Prahran VIC
Australia

Email: mnot@mnot.net
URI: https://www.mnot.net/

Piotr Sikora
Google

Email: piotrsikora@google.com