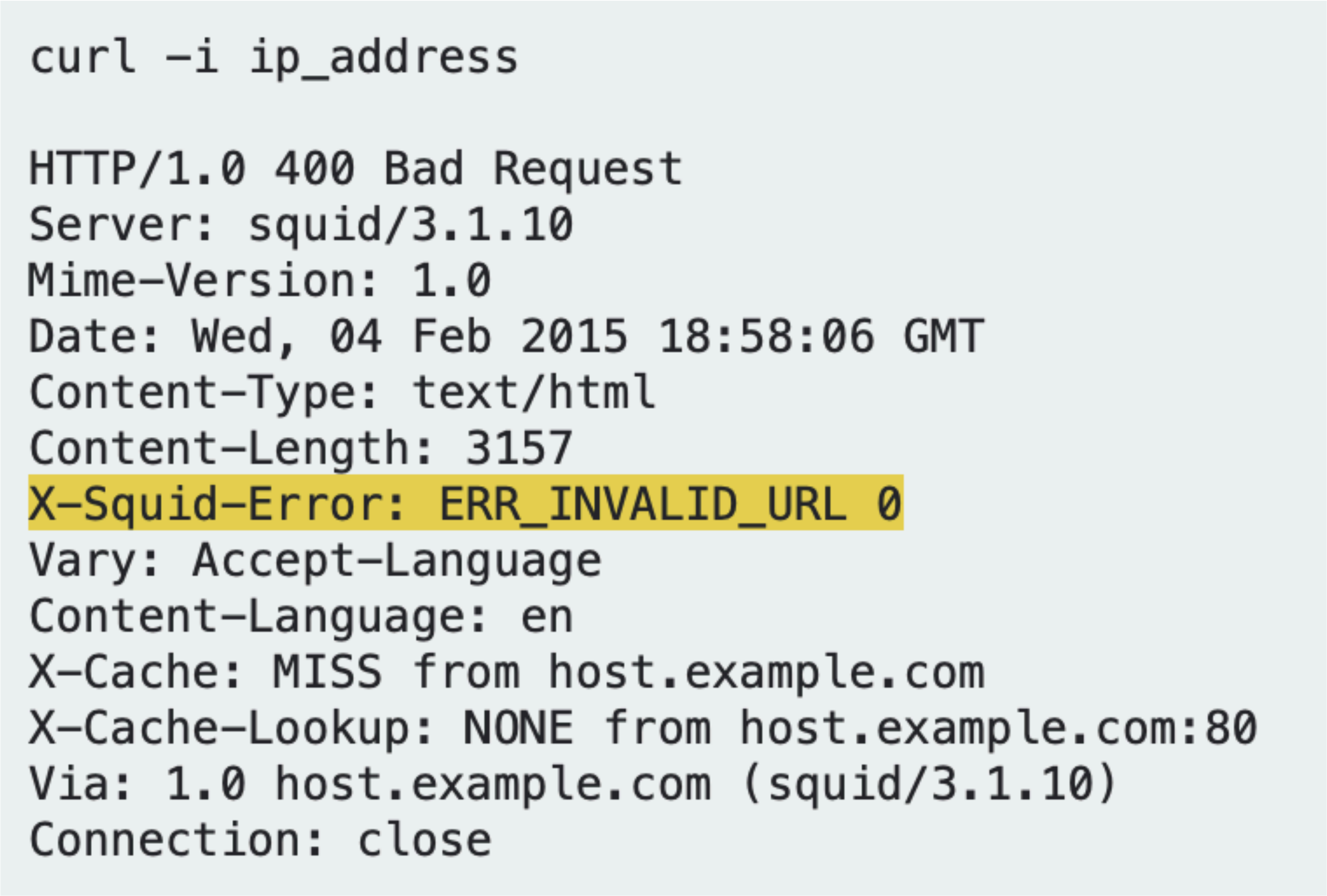
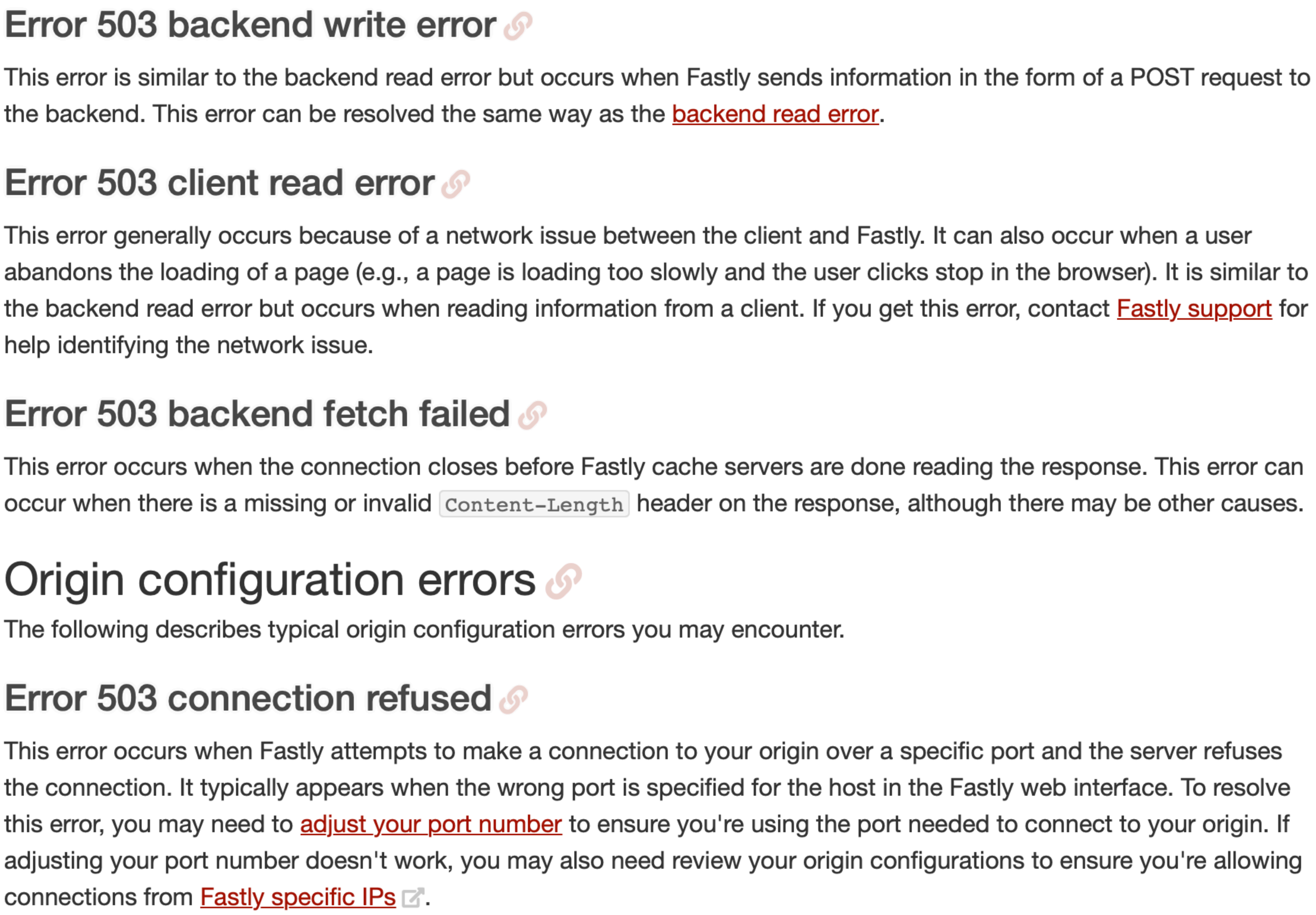
Proxy-Status

IETF104



```
curl -i ip_address

HTTP/1.0 400 Bad Request
Server: squid/3.1.10
Mime-Version: 1.0
Date: Wed, 04 Feb 2015 18:58:06 GMT
Content-Type: text/html
Content-Length: 3157
X-Squid-Error: ERR_INVALID_URL 0
Vary: Accept-Language
Content-Language: en
X-Cache: MISS from host.example.com
X-Cache-Lookup: NONE from host.example.com:80
Via: 1.0 host.example.com (squid/3.1.10)
Connection: close
```



Error 503 backend write error

This error is similar to the backend read error but occurs when Fastly sends information in the form of a POST request to the backend. This error can be resolved the same way as the backend read error.

Error 503 client read error

This error generally occurs because of a network issue between the client and Fastly. It can also occur when a user abandons the loading of a page (e.g., a page is loading too slowly and the user clicks stop in the browser). It is similar to the backend read error but occurs when reading information from a client. If you get this error, contact Fastly support for help identifying the network issue.

Error 503 backend fetch failed

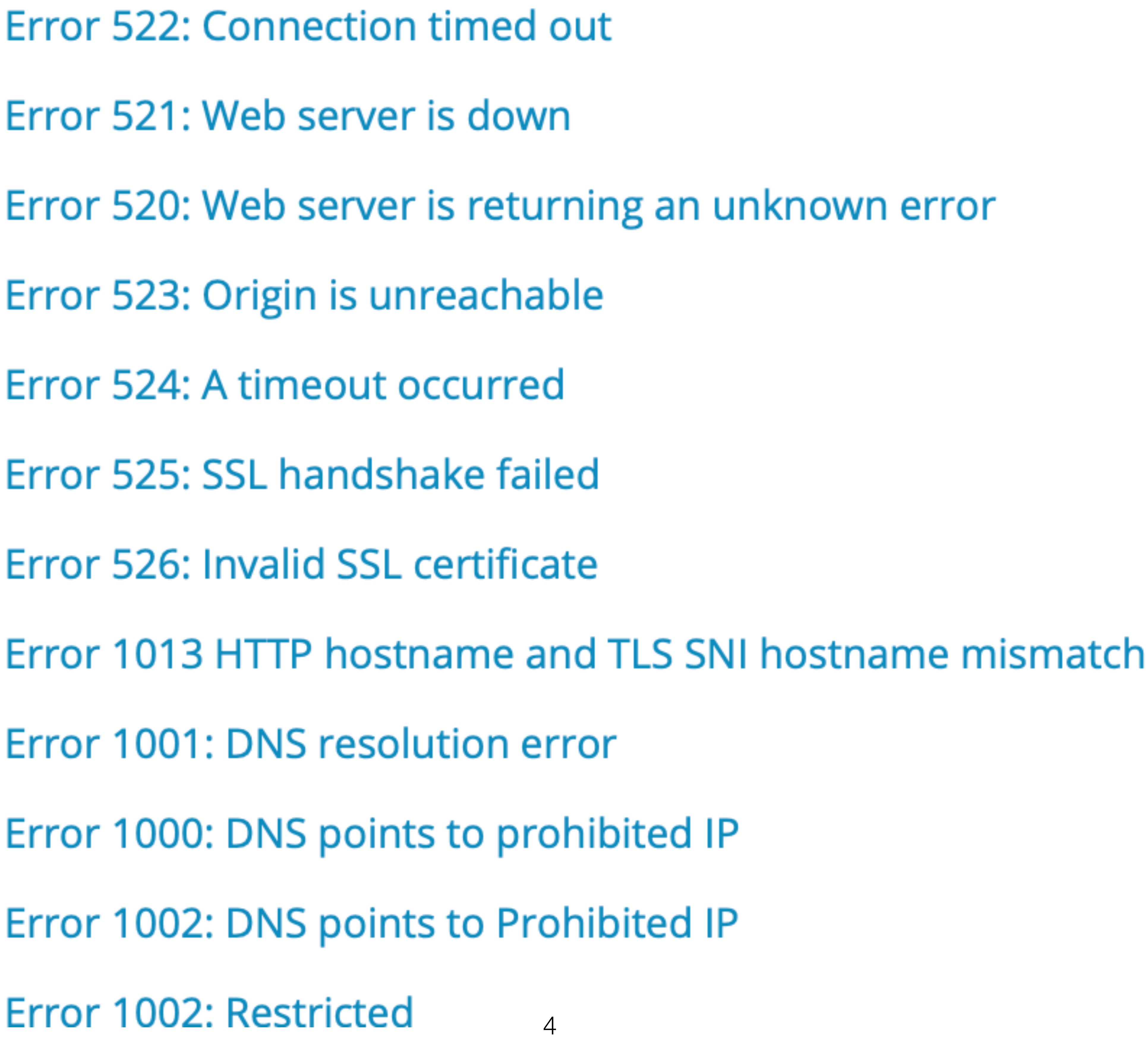
This error occurs when the connection closes before Fastly cache servers are done reading the response. This error can occur when there is a missing or invalid `Content-Length` header on the response, although there may be other causes.

Origin configuration errors

The following describes typical origin configuration errors you may encounter.

Error 503 connection refused

This error occurs when Fastly attempts to make a connection to your origin over a specific port and the server refuses the connection. It typically appears when the wrong port is specified for the host in the Fastly web interface. To resolve this error, you may need to adjust your port number to ensure you're using the port needed to connect to your origin. If adjusting your port number doesn't work, you may also need review your origin configurations to ensure you're allowing connections from Fastly specific IPs.



Error 522: Connection timed out

Error 521: Web server is down

Error 520: Web server is returning an unknown error

Error 523: Origin is unreachable

Error 524: A timeout occurred

Error 525: SSL handshake failed

Error 526: Invalid SSL certificate

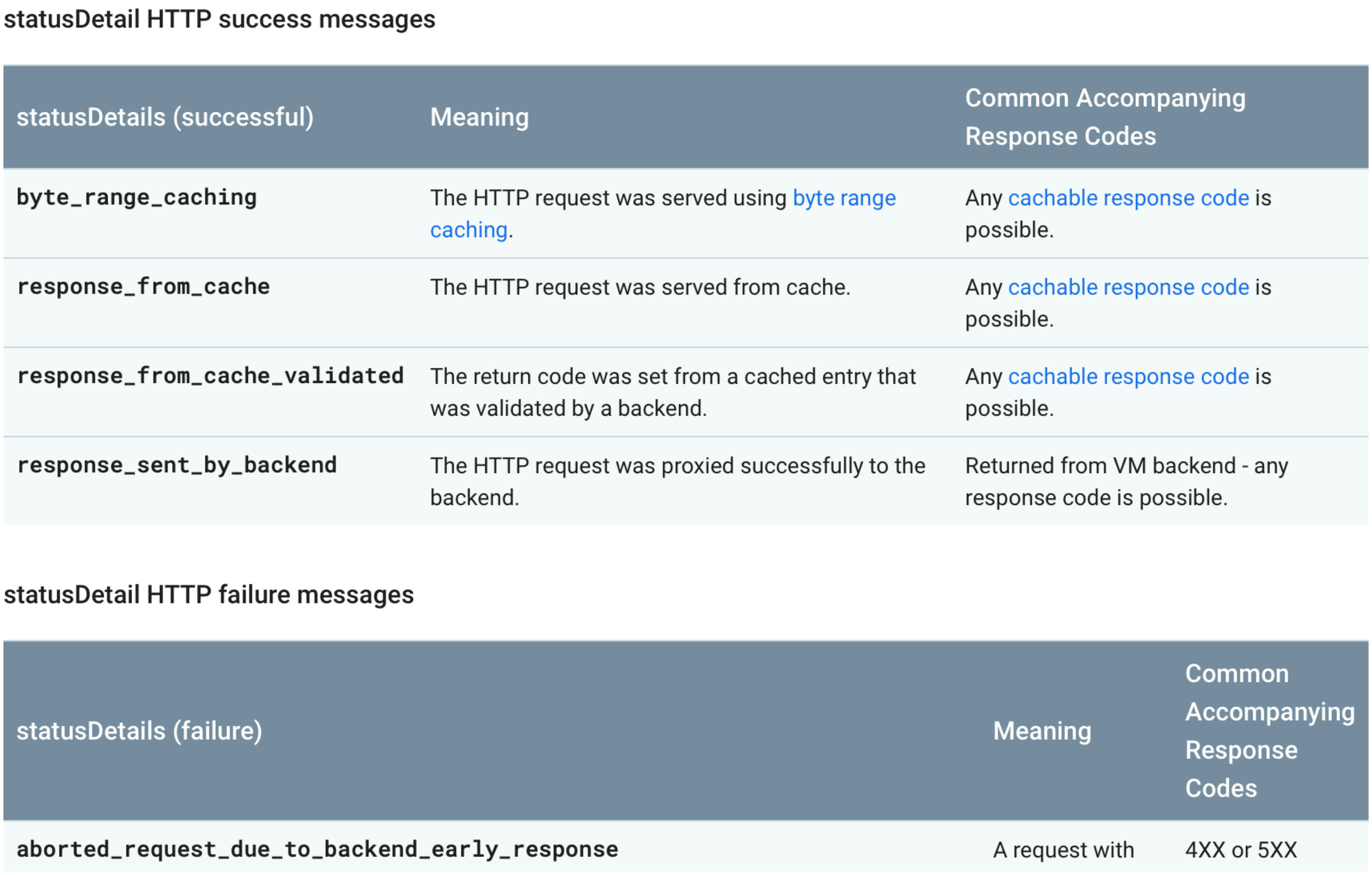
Error 1013 HTTP hostname and TLS SNI hostname mismatch

Error 1001: DNS resolution error

Error 1000: DNS points to prohibited IP

Error 1002: DNS points to Prohibited IP

Error 1002: Restricted



statusDetail HTTP success messages

statusDetails (successful)	Meaning	Common Accompanying Response Codes
`byte_range_caching`	The HTTP request was served using byte range caching.	Any cachable response code is possible.
`response_from_cache`	The HTTP request was served from cache.	Any cachable response code is possible.
`response_from_cache_validated`	The return code was set from a cached entry that was validated by a backend.	Any cachable response code is possible.
`response_sent_by_backend`	The HTTP request was proxied successfully to the backend.	Returned from VM backend - any response code is possible.

statusDetail HTTP failure messages

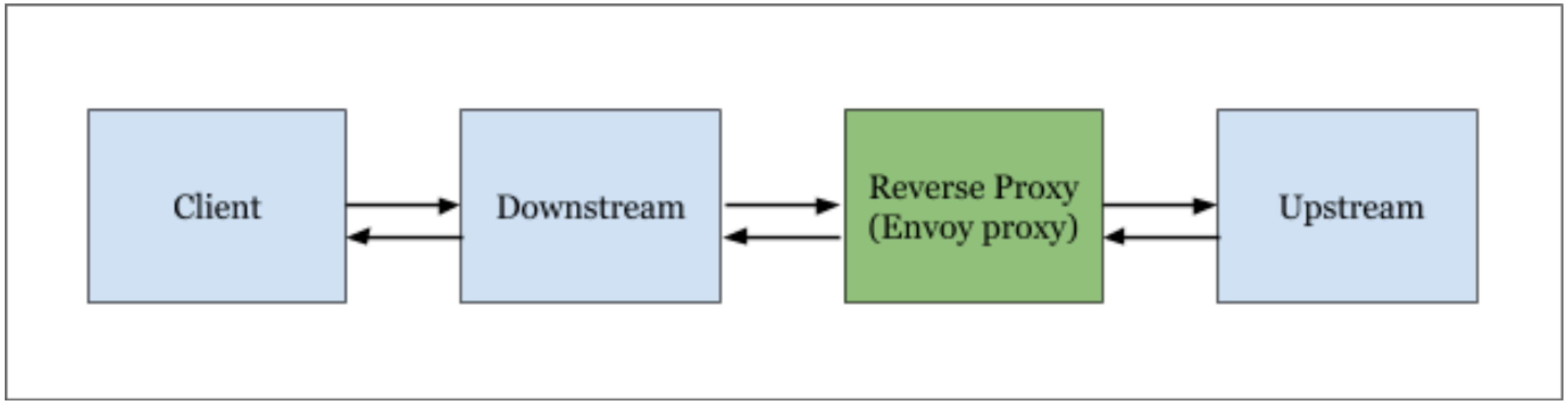
statusDetails (failure)	Meaning	Common Accompanying Response Codes
`aborted_request_due_to_backend_early_response`	A request with	4XX or 5XX

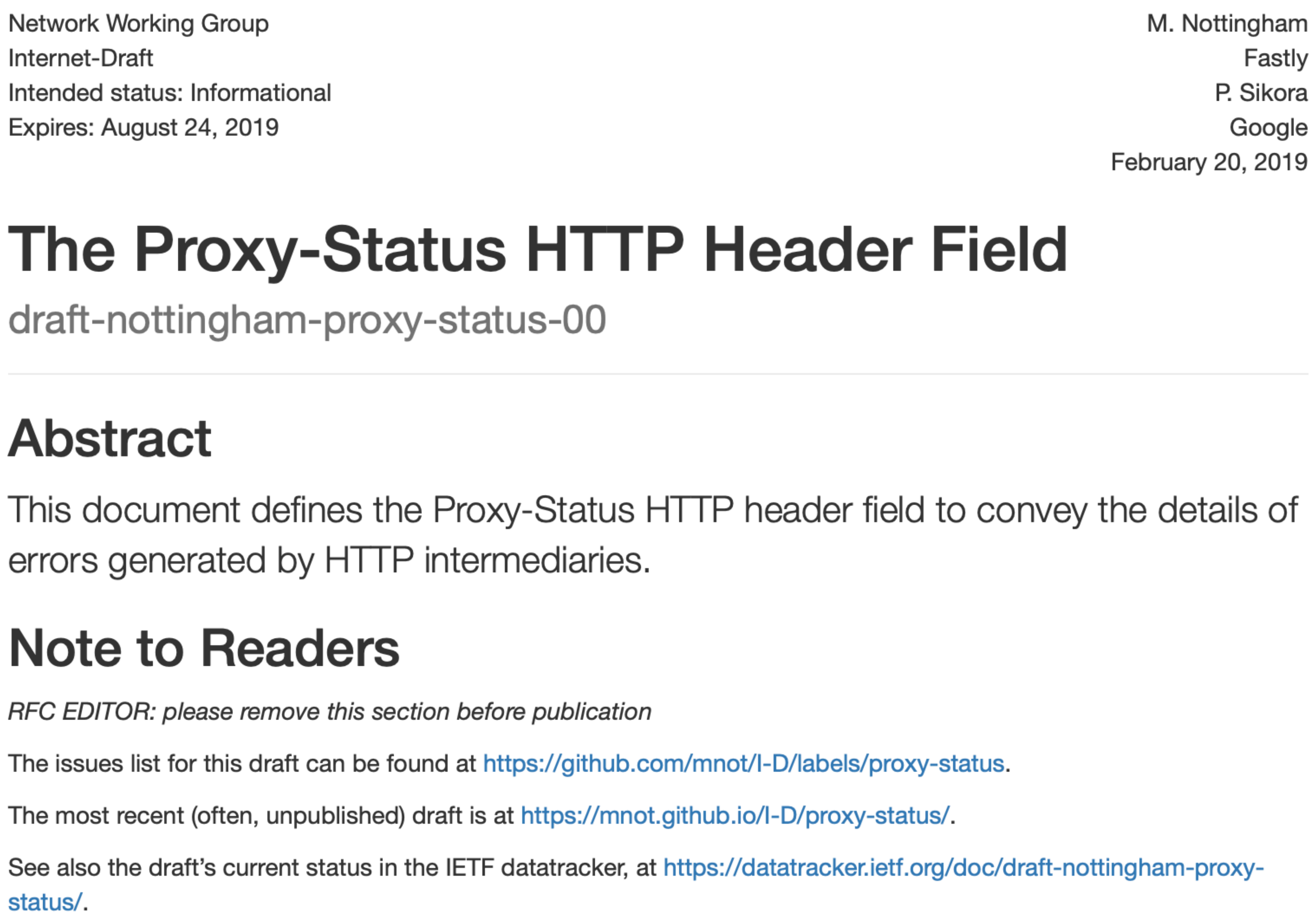
Reverse Proxy Response Propagation to downstream

Author : meshram@google.com
Contributors : piotrsikora@google.com

Objective

At present, many reverse proxies do not provide the ability to propagate a detailed response from upstream to downstream. This capability can be extremely useful in scenarios where a reverse proxy is introduced to solve cross cutting concerns and downstream wants to get details about what exactly happened at upstream. The downstream might also like to take some action based on the response it received or propagate a detailed response to the downstream for further action. In this design we will consider the use case of Envoy proxy.





Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 24, 2019

M. Nottingham
Fastly
P. Sikora
Google

February 20, 2019

The Proxy-Status HTTP Header Field

draft-nottingham-proxy-status-00

Abstract

This document defines the Proxy-Status HTTP header field to convey the details of errors generated by HTTP intermediaries.

Note to Readers

RFC EDITOR: please remove this section before publication

The issues list for this draft can be found at https://github.com/mnot/I-D/labels/proxy-status.

The most recent (often, unpublished) draft is at https://mnot.github.io/I-D/proxy-status/.

See also the draft’s current status in the IETF datatracker, at https://datatracker.ietf.org/doc/draft-nottingham-proxy-status/.

```
HTTP/1.1 504 Gateway Timeout
Proxy-Status: connection_timeout; proxy=SomeCDN; origin=abc; tries=3
```

```
HTTP/1.1 200 OK
Proxy-Status: http_response_status; proxy=SomeCDN; origin=abc
```

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

Interesting?

Adopt?