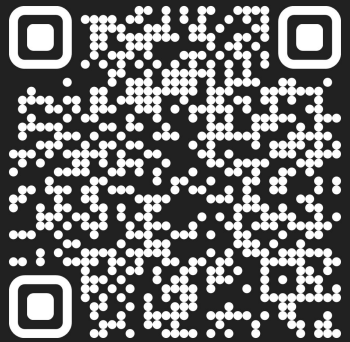
Do you speak HTTP?

What’s my and your HTTP implementation missing?

Mohammed Al Sahaf



In the beginning… there were the reports

User reported Redbot complaints:

- 304 response missing required headers
- Partial response missing required headers
- Vary is missing
- ETag doesn’t change between negotiated responses
- If-None-Match conditional request returned the full content unchanged

Fixed, but…

Q: What if they sneak back?

Q: What else are we missing?

In the beginning… there were the reports

User reported Redbot complaints:

- 304 response missing required headers
- Partial response missing required headers
- Vary is missing
- ETag doesn’t change between negotiated responses
- If-None-Match conditional request returned the full content unchanged

Fixed, but…

Q: What if they sneak back?

A: Tests!!

Q: What else are we missing?

In the beginning… there were the reports

User reported Redbot complaints:

- 304 response missing required headers
- Partial response missing required headers
- Vary is missing
- ETag doesn’t change between negotiated responses
- If-None-Match conditional request returned the full content unchanged

Fixed, but…

Q: What if they sneak back?

A: Tests!!

Q: What else are we missing?

A: ???

Then… there are the questions

What if HTTP/1.1 in-band controls disagree with the data? E.g. Content-Length header value mismatch with response body size

How do user agents handle it?

What do users/developers expect?

Users Expectation

response body > content-length	
Response	**Count / %**
Ignore header; read fully	4 votes; 21.1%
Read till content-length value	6 votes; 31.6%
Abort/reject	9 votes; 47.4%

response body < 0	
Response	**Count / %**
Ignore value	7 votes; 33.3%
Reject/abort	14 votes; 66.7%

response body < content-length	
Response	**Count / %**
Ignore header; read 3	7 votes; 38.9%
Pad; with what?	0 votes; 0%
Reject/abort	9 votes; 50%
Other; comment	2 votes; 11.1%, none of them elaborated

Scenario: response body < content-length

	HTTP/1.1	HTTP/2
curl	`curl: (18) transfer closed with 2 bytes remaining to read`	(92) HTTP/2 stream 1 was not closed cleanly: PROTOCOL_ERROR (err 1)
	response payload displayed	response payload displayed
Firefox	`NS_ERROR_PARTIAL_TRANSFER`	
	response payload displayed	displays the full payload without reporting any errors
Chrome	`(failed)net::ERR_CONTENT_LENGTH_MISMATCH`	
	nothing displayed	displays the full payload without reporting any errors

Meet the latest HTTP client: fancy-http-client.sh

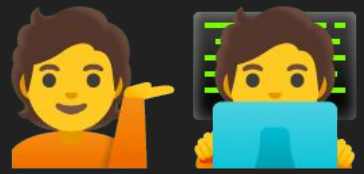
```
#!/usr/bin/env bash

URL=$1

# Extract the host and port
HOST_PORT=${URL#*://}
HOST_PORT=${HOST_PORT%%/*}

# Extract the host
HOST=${HOST_PORT%:*}

# Extract the port, if present
PORT=${HOST_PORT#*:}
if [ "$PORT" == "$HOST_PORT" ]; then
 PORT="80"
fi

# Extract the path
URL_PATH=${URL#*/}
# assume / if PATH is empty
if [ "$URL_PATH" == "$URL" ]; then
   URL_PATH="/"
fi

ip=$(dig +short "$HOST")
echo "GET /$URL_PATH" | nc "$ip" "$PORT"
```

Server:

```
caddy file-server \
    --domain localhost \
    --browse \
    --listen :80
```

Ignores section “4.2.1.” of RFC 9110, specifically the section:

> A sender MUST NOT generate an "http" URI with an empty host identifier. A recipient that processes such a URI reference MUST reject it as invalid.



Well, who's to say what's an HTTP server/client?!

We govern the RFCs!

IETF

HTTP

Category	RFC #	RFC Title
Core	9110	HTTP Semantics
	9111	HTTP Caching
	9112	HTTP/1.1
	9113	HTTP/2
	9114	HTTP/3
	7541	HPACK: Header Compression for HTTP/2
	9204	QPACK Field Compression for HTTP/3
	9651	Structured Field Values for HTTP
	9205	Building Protocols with HTTP

Category	RFC #	RFC Title
Core	9110	HTTP Semantics
	9111	HTTP Caching
	9112	HTTP/1.1
	9113	HTTP/2
	9114	HTTP/3
	7541	HPACK: Header Compression for HTTP/2
	9204	QPACK Field Compression for HTTP/3
	9651	Structured Field Values for HTTP
	9205	Building Protocols with HTTP

9 RFCs

RFC 9110: HTTP Semantics

Requirements Keywords Count										
MUST	MUST NOT	REQUIRED	SHALL	SHALL NOT	SHOULD	SHOULD NOT	RECOMMENDED	NOT RECOMMENDED	MAY	OPTIONAL
135	75	0?	0	0	98	32	1	0	119?	1?

RFCs are Great, but

Written in natural language, ambiguous

Walls of text

Monolingual (English)

Interpreted by the HumanVM™

Reinterpreted every time

Category	RFC #	RFC Title	Word Count
Core	9110	HTTP Semantics	65,000
	9111	HTTP Caching	11,000
	9112	HTTP/1.1	15,000
	9113	HTTP/2	26,000
	9114	HTTP/3	21,000
	7541	HPACK: Header Compression for HTTP/2	13,000
	9204	QPACK Field Compression for HTTP/3	12,000
	9651	Structured Field Values for HTTP	10,000
	9205	Building Protocols with HTTP	9,000

Proposal

To develop an HTTP spec compliance test framework and suite to test an HTTP actor (agent or server) compliance to the collection of relevant HTTP RFCs and is blessed by the HTTP WG

Subject raised in the mailing list on 2024-05-26:

https://lists.w3.org/Archives/Public/ietf-http-wg/2024AprJun/0056.html

Acknowledgement:

Dalibor Karlović

GitHub: /dkarlovi

“Would creating this ‘Hurl-based HTTP spec test suite’ be better as a standalone project which Caddy (and others, hopefully) can take advantage of and, hopefully, maintain?”

Goals

To provide a

diagnostic

test suite

Non-Goals

To provide a rubber stamp for HTTP servers or clients as bragging rights

Why Bother?

Why Bother?

- Assurance of compatibility
- Assists in optimization efforts
- Unifies the expectations of the community
- Removes ambiguity
- Feedback loop

Suggested Approach

Declarative language

Phased:

1. Server
2. Client

Suggested Approach

Declarative language

Phased:

1. Server
2. Client

Easier?

“Testing clients is very difficult because contrary to servers which just have to respond to sollicitations, someone has to act on the client to run the desired tests, so the approach is different (and different between various clients)” ~ Willy Tarreau

Challenges

- Tooling
- DSL definition

Poul-Henning Kamp adds

- Two-way protocols “explode combinatorically in very few exchanges”
- Time definition
- Semantics

compliance-client:

HEAD /

server-under-test:

301 to /justkidding

compliance-client:

HEAD /justkidding

server-under-test:

301 to /

Prior Art - https://github.com/httpwg/wiki/wiki/HTTP-Testing-Resources

Tool Name	Title	License
h2spec	HTTP/2 framing layer tests	Open Source
cache-tests.fyi	Caching tests	Open Source
Co-Advisor	Intermediary (including caching) tests	Proprietary
REDbot	HTTP resource linter	Open Source
httplint	HTTP message linter	Open Source
VTest	Intermediary tests	Open Source
Content-Disposition Tests	Browser tests for Content-Disposition handling	Open Source
Structured Header Tests	parsing and serialisation tests	Open Source
WebServer Tester	Server tests with CLI and Web UI	Open Source
Daft	Framework for testing	Open Source
Web Platform Tests	Specifically, Fetch	Open Source

Prior Art - https://github.com/httpwg/wiki/wiki/HTTP-Testing-Resources

Tool Name	Title	License
h2spec	HTTP/2 framing layer tests	Open Source
cache-tests.fyi	Caching tests	Open Source
Co-Advisor	Intermediary (including caching) tests	Proprietary
REDbot	HTTP resource linter	Open Source
httplint	HTTP message linter	Open Source
VTest	Intermediary tests	Open Source
Content-Disposition Tests	Browser tests for Content-Disposition handling	Open Source
Structured Header Tests	parsing and serialisation tests	Open Source
WebServer Tester	Server tests with CLI and Web UI	Open Source
Daft	Framework for testing	Open Source
Web Platform Tests	Specifically, Fetch	Open Source

Next Steps

- Concurrence
- Process & bureaucracy
- Project management
  - Identifying stakeholders
  - Design
  - Phasing
  - Task tracker
  - Project page/repository
- Work!