INTERNET-DRAFT
Document: draft-nottingham-http-roles-00.txt
Expires: 1 March 2000

M. Nottingham
1 September 1999

Server-Side Roles in the HTTP

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts. 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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Please send comments to Mark Nottingham <mnot@pobox.com>

1. Abstract

Web servers are becoming more complex, and as a result are losing the full benefits of HTTP protocol compliance.

This applicability statement defines classifications of Web server sub-components and clarifies their responsibilities in implementing HTTP/1.1 protocol features, with a discussion of the motivations for doing so.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [1].

3. Introduction

Although many Web servers available claim HTTP/1.1 [2] compliance, the majority do not issue compliant responses for every object served. This is especially noticeable for the performance-enhancing

provisions of the protocol, such as persistent connections, transfer encoding, and object validation [3].

This behavior is most evident in objects which are created by non-traditional means [4]. Historically, the primary method of managing Web content was by storing it in a filesystem as discrete files. Web servers were designed to serve static files as efficiently as possible, while still having important (and readily available) metadata such as Last-Modified (for validation) and Content-length (for persistent connections) available.

Dynamically generated content (such as CGI) was thought of as a small portion of the total traffic, and relatively inconsequential when optimizing performance.

This is no longer necessarily the case. CGI and other specialized methods are being used to manage entire Web sites. The filesystem, with its easily generated metadata, is being replaced by a software layer that presents a much more opaque profile to the server.

In common practice, this lack of clear responsibility between server components results in partial or no support for key protocol features, leaving compliance in the hands of the content publisher.

Consequently, the performance benefits of HTTP/1.1 are unrealized for an increasing portion of Web traffic. Research has established that they are both able to be realized and feasible to implement [5].

Additionally, future protocol extensions may not be useable for a large number of entities generated by a server, for the same reasons.

4. Goals

The goals of this document are to:

1. Define the roles and responsibilities of server-side components, with respect to implementing HTTP/1.1 protocol features.

2. Promote interoperability of Web servers and increase the protocol compliance of objects served, without to regard to how they are generated.

3. Prevent the burdening of content publishers with the responsibility for protocol compliance by shifting responsibility to the server, making the HTTP implementation transparent.

5. Terminology

HTTP/1.1 defines two primary roles implicitly:

client
: A program that establishes connections for the purpose of sending requests.

server
: An application program that accepts connections in order to service requests by sending back responses. Any given program may be capable of being both a client and a server; our use of these terms refers only to the role being performed by the program for a particular connection, rather than to the program's capabilities in general. Likewise, any server may act as an origin server, proxy, gateway, or tunnel, switching behavior based on the nature of each request.

In this document, server means the component of the software that actually responds to the request. Also, we only consider generalized servers; that is, those that are capable of serving content for a number of purposes. Single-purpose (e.g., embedded) servers are out of scope.

Additional roles, such as proxy, gateway and tunnel, are defined, but are considered out of scope for this proposal.

The following terms are new:

content
: The intended entity-body payload. It may or may not be actually transferred in a response.

content generator
: A means by which requests are mapped to content, which is optionally transformed, generated or otherwise prepared, and then passed back to the server, which then sends it to the client. The generator may be, but is not necessarily, external to the server.

publisher
: A person or persons who make content available on the server, possibly through a content generator.

synthetic validator
: A validator generated by a server, without knowledge of the process used to produce the content it describes.

6. Server Role and Responsibilities

Implementation of HTTP protocol features should, in general, be the responsibility of the server. A server SHOULD NOT expect a content generator, content management system or publisher to handle them, except where it does not have sufficient information available to do so.

Core features of interest are listed below; this is not meant to be exhaustive, but only an indication of those most critical and obvious.

6.1 Persistent Connections

Servers MUST support a persistent connection if the content generator supplies a Content-Length header. If it is not available, the server SHOULD attempt to buffer the response in order to generate one, although this MAY be circumvented if:
* the server does not have resources (i.e., memory) to do so, or
* the object is very large, and overall latency becomes unacceptable, or
* the time required to generate the object adds unacceptable latency

Because many clients begin rendering HTML as soon as it is available, servers MAY reduce size and time thresholds for text/html responses.

There MUST be a method by which content generators can specify that content is not to be buffered; this MAY be performed by a pseudo-HTTP header that is consumed by the server.

Servers MUST serve chunked encoding responses for all objects, if:
* Content-Length is unavailable and impractical to generate and
* the client advertises itself as HTTP/1.1 capable, or
* the client includes 'chunked' in a TE header

6.2 Validators and Validation

Servers MUST pass through any validator (e.g., Etag or Last-Modified date) sent by a Content Generator, without modification.

Conditional request validation MUST be handled by regenerating the object, comparing the validators, and returning an appropriate response. Servers MAY use an alternate method of regenerating the validator without generating the entire object, if the validator

produced is equivalent.

6.2.1 Synthetic Validators

Servers SHOULD generate synthetic validators for objects that do not any associated, when possible. However, this MUST NOT be done when the generated object contains a Cache-Control: no-store header.

Synthetic validators MUST be strong, and MUST have a Cache-Control: no-cache and Expires time in the past associated, unless a freshness period is explicitly assigned.

One possible method of synthetic validator generation would be to chunk-encode or buffer the response (as outlined above) and append a Trailer containing the MD5 hash of the body as a validator.

6.3 Partial Content

Servers MUST respond to partial-content requests (i.e., those with Range headers) appropriately, by using the Content Generator to produce the response, and then sending back the requested range(s).

6.4 Transfer-Encoding

Servers SHOULD reply to requests that allow transfer encoding of objects (i.e., TE header present) with appropriate encoding, in a fashion transparent to the content generator.

6.5 HEAD Requests

HEAD requests SHOULD be responded to by using the content generator to produce a GET response, omitting the body before returning it to the client.

6.5 Publisher Access

Servers SHOULD provide a suitable method of setting per-object and global metadata (such as Expires times and Cache-Control headers) to users. If WebDAV [6] is implemented, it MAY be the mechanism used.

7. Content Generator Role and Responsibilities

Content generators SHOULD provide as much information as possible to the server, to facilitate efficient operation. This includes Content-Length, Etag and/or Last-Modified validators, and freshness information, as possible.

Content generators SHOULD NOT specify unique URLs for multiple instances of identical content, if avoidable. When this does happen predictably, the generator SHOULD include a Cache-Control: no-store

header.

User-specific or transaction-based content SHOULD have an associated Cache-Control: no-store header, unless there is a possibility of reuse of the resource (by one or more clients), and side effects of the transaction are not important.

Content generators SHOULD allow publishers to specify that buffering is not to be used for specific content.

8. Security Considerations

There are no security implications specific to this document; see the HTTP/1.1 security section for general information.

9. References

1 Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997
2 Fielding, R. et al., "Hypertext Transfer Protocol - HTTP/1.1", RFC2616, June, 1999
3 Krishnamurthy, B., Arlitt, M., "PRO-COW: Protocol Compliance on the Web", AT&T Labs - Research Technical Report # 990803-05-TM
4 Nottingham, M., "Web Server Capabilities", http://www.mnot.net/papers/capabilities.html
5 Wills, C., Mikhailov, M., "Examining the Cacheability of User-Requested Web Resources", 4th Web Caching Workshop, San Diego CA, March 1999
6 Whitehead, E. et al., "HTTP Extensions for Distributed Authoring - WEBDAV", RFC2518, February 1999

10. Acknowledgments

The author would like to thank Mike Ciavarella for his constructive comments.

11. Author's Addresses

Mark Nottingham
Email: mnot@pobox.com

Full Copyright Statement