HTTP Methods for Web Services

Paul Adamczyk, Munawar Hafiz and Ralph E. Johnson
Department of Computer Science
University of Illinois
(padamczy, mhafiz, johnson)@cs.uiuc.edu

Abstract

We present a study of the 100 most popular websites showing that many features of HTTP/1.1 (specifically, HTTP methods) are not implemented and configured correctly. We relate these results to the current use of HTTP methods by Web services and speculate about the future trends.

1. Introduction

The Hypertext Transfer Protocol (HTTP) has been the cornerstone of the Web since its inception. The current version, HTTP/1.1, includes many features meant to improve the performance of HTTP. Some of these added features are very useful, but not all the new features are used universally. Our poster describes how modern systems implement HTTP/1.1. We surveyed top 100 websites in the USA [1] to see how well they comply with the standard.

HTTP is the foundation of Web services as well. While most research focuses on SOAP-based Web services, the industry leans more towards RESTful Web services. SOAP-based Web services are defined to use POST as the only HTTP method. We do not propose to change that; instead focus on RESTful Web services that use only GET and POST methods. Other HTTP methods are defined for tasks seemingly well suited for Web services, however current Web services do not use them at all. This poster discusses the impact of the current conceptual understanding of the HTTP standard on the future of Web services, specifically RESTful Web services.

2. Background and Experimental Results

2.1. Overview of HTTP Methods

The current definition of HTTP/1.1 [2], consists of eight methods. We have divided these methods into three groups:

(1) Read-only methods (GET and HEAD) that do not modify the server. These are the only methods to which all resources must respond.
(2) Write methods (POST, PUT, and DELETE) that modify the data on the server.
(3) Supporting methods (OPTIONS, TRACE, and CONNECT) that were added in version 1.1.

2.2. Related Work

We are aware of only one experimental work that is similar to ours. PRO-COW [4] studied the compliance of Web servers with HTTP/1.1 in 1999, soon after the standard was published. This study focused on correct implementation of the mandatory and optional HTTP features, including methods and

method headers. It tested 15 most popular websites in 1999. Our study included 10 of these websites. The follow-up PRO-COW study from 2001 [5] included 150 websites, selected based on 3 different services reporting the most popular sites.

2.3. Compliance Results

Most websites do not respond to the HTTP methods correctly according to the standard. See the poster for detailed results. More detailed results are available at `http://st.cs.uiuc.edu/~padamczy/http_tests-ws.html`. These results provide good starting point for discussing how Web services should use HTTP methods.

3. Relationship to Web Services

While SOAP-based Web services are designed to use POST exclusively, the RESTful Web services should, ideally, take advantage of all HTTP methods to define better interfaces. In this section, we explain what our test results suggest about this idea.

3.1. Supporting Methods

The OPTIONS method looks suspiciously similar to UDDI. The clients could use it to dynamically discover what methods are supported by a resource and then select the most appropriate method to invoke. However, the fact that only few of the websites configure this method to return unambiguous, correct results indicates that dynamic discovery is probably not such a great idea and it's not likely to work for the RESTful Web services either. The OPTIONS method appears to be used mainly to "confuse the enemy" (*i.e.* hackers). It returns misleading results to prevent hackers from learning the capabilities of the system. Browsers never send the OPTIONS method to the server. Only hackers use it. Hence it is natural to obscure the information to achieve security through obscurity.

The TRACE method appears to be more useful, because message tracing and debugging is very difficult in distributes systems, such as the Web. Yet, our results show that only few of the websites handle this method correctly. We suspect that the reason for low compliance of both of these methods is security. Ideally, Web services should benefit from incorporating these methods, but the disappointing compliance results make us hesitant to suggest that.

Security of RESTful Web services depends on HTTPS and the CONNECT method. The SOAP-based Web services define standards for secure SOAP-based messaging (the WS-* standards), but no implementations of Web services based on these standards exist, because of their computational overhead. In contrast, CONNECT is lightweight and it provides strong security guarantee. This is why that standard is universally adopted for secure communication.

3.2. Read-only Methods

GET is the universal method of RESTful Web services. It is used to retrieve data and sometimes even to make changes to the server. For example, some Web services use GET for updating the server data:

- Bloglines API uses GET to mark unread items as read.
- Flickr API uses GET to delete a photo set.
- del.icio.us API uses GET to delete a post from the site.

But these are drastic exceptions. Web services are diligent in using GET for read requests and POST for write requests. Since these violations have been publicized, Flickr has replaced GET with POST in the API specification.

Web services do not use conditional GET, because they do not use HTTP headers. Our results show that the conditional GET is the only method whose compliance has improved significantly since the PRO-COW experiments. This indicates that caching is an important feature. We believe that conditional GET should be used by Web services, because it significantly improves performance via caching.

Web services do not use HEAD now. HEAD is a useful method for traditional Web applications as it is used for retrieving metadata. This is an unimportant part of Web services and we believe that it is not needed for Web services.

3.3. Write Methods

The RESTful Web services literature uses PUT and DELETE methods as examples that Web services do not follow REST [3]. Existing Web services implementations do not use PUT and DELETE methods.

We have mentioned that Flickr modified the API for deleting a photo set to use POST, not GET. But according to the HTTP standard, DELETE method would be more appropriate. The default pattern for handling deletion by Web services is to send the POST method to a special Request-URI that includes the word "delete." This is not a good interpretation of the standard, because the URI identifies a resource. Including "delete" as a part of the resource identification means that the resource is a stub for a delete method on the server.

Considering how rarely PUT and DELETE are configured by websites, it is not likely that insistence on having 3 methods for modifying Web services resources will lead to a widespread acceptance of this approach. The current state of practice is likely to remain unchanged and POST will remain the only method for writing in Web services.

4. Conclusion

The results we present confirm some well-known facts. However, we present a more complete picture of which HTTP methods are important and why. Websites often configure all HTTP methods, but there is a large variety of incorrect configurations. This variety shows lack of common understanding of the concepts behind these methods. The results suggest which HTTP methods are important and should be considered for inclusion by Web services.

References

[1] Alexa Internet. Top sites United States. 2006. `http://www.alexa.com/site/ds/top_sites?cc=US&ts_mode=country&lang=none`.

[2] R. T. Fielding, J. Gettys, J. C. Mogul, H. Frystyk Nielsen, L. Masinter, P. J. Leach, and T. Berners-Lee. Hypertext Transfer Protocol — HTTP/1.1. Internet proposed standard RFC 2616, June 1999.

[3] G. Goth. Critics say Web services need a REST. *IEEE Distributed Systems Online*, 5(12):1–1, Dec. 2004.

[4] B. Krishnamurthy and M. Arlitt. PRO-COW: Protocol compliance on the Web. Technical Report 990803-05-TM, 1999.

[5] B. Krishnamurthy and M. Arlitt. PRO-COW: Protocol compliance on the Web — A longitudinal study. In *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS-01)*, 2001.

ECOWS'06
European Conference on Web Services • 4-6 December 2006, Zurich, Switzerland

HTTP Methods for Web Services

Paul Adamczyk, Munawar Hafiz and Ralph E. Johnson
(padamczy, mhafiz, johnson)@cs.uiuc.edu

Software Architecture Group
University of Illinois

Overview

- A study of the HTTP/1.1 compliance of 100 most popular US websites.
- Most HTTP methods fail to show compliance.
- We relate these results to the current use of HTTP methods by Web services.

HTTP Standard

HTTP Methods

We have divided HTTP methods into three groups

1) **Supporting Methods**: OPTIONS, TRACE and CONNECT.
2) **Read-only Methods**: GET and HEAD.
3) **Write Methods**: POST, PUT, and DELETE.

Evolution of HTTP

Description	Supported Methods	Change from previous
Initial implementation	GET	-
Before HTTP/1.0	**GET, PUT, DELETE, POST, HEAD, *CHECKIN, CHECKOUT, LINK, UNLINK, TEXTSEARCH, SEARCH, SHOWMETHOD, SPACEJUMP***	Added all methods but GET (Methods that were not in RFC 1945 are in italics)
RFC 1945 (HTTP/1.0)	Main: GET, HEAD, POST Additional: PUT, DELETE, LINK, UNLINK	Added LINK, UNLINK
RFC 2068 (HTTP/1.1) (Obsoletes RFC 1945)	GET, HEAD, POST, PUT, DELETE, OPTIONS, TRACE	Removed LINK, UNLINK
RFC 2616 (HTTP/1.1) (Obsoletes RFC 2068)	GET, HEAD, POST, PUT, DELETE, OPTIONS, TRACE, CONNECT	Added CONNECT

Interpretation of HTTP compliance

1. **Unconditional Compliance**: Implement all the "must" and "should" requirements.
2. **Conditional Compliance**: Implement all "must" requirements.
3. **Non-compliance**: Fails at least one "must" requirement.

Experimental Setup

OPTIONS, GET, TRACE and HEAD

- Implemented an HTTP client using VisualWorks 7.1 (Smalltalk).
- The client sent HTTP requests to the home page of the top 100 US websites, according to Alexa (http://www.alexa.com).

CONNECT

- Used Fiddler, an HTTP debugging proxy to log HTTP traffic.
- Manually checked the CONNECT requests and responses.

Comparison with Related Work

	OPTIONS		TRACE		GET		GET-un		HEAD	
	PRO-COW*	US	PRO-COW	US	PRO-COW	US	PRO-COW	US	PRO-COW	US
Unconditional	59.8	60.0	97.3	58.0	83.5	100.0	41.7	66.0	72.9	29.0
Conditional	39.4	25.0	2.5	15.0	16.1	0.0	1.2	34.0	9.4	60.0
Non-compliant	0.8	15.0	0.2	27.0	0.4	0.0	57.1	0.0	17.7	11.0

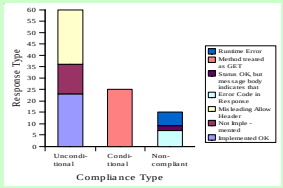
*** Ref:** B. Krishnamurthy and M. Arlitt. *PRO-COW:Protocol compliance on the Web*. Technical Report 990803-05-TM, 1999.

HTTP Compliance Test Results

Supporting Methods

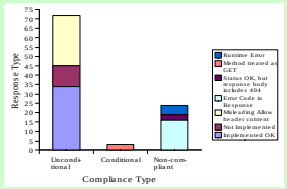
OPTIONS

	Unconditional	Conditional	Non-compliant
Implemented OK	23		
Not Implemented	13		
Misleading Allow Header	24		
Method treated as GET		25	
Error Code in Response			7
Status OK, but entity indicates that method is not supported			2
Runtime Error			6



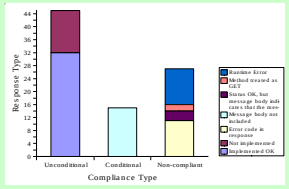
OPTIONS/*

	Unconditional	Conditional	Non-compliant
Implemented OK	34		
Not Implemented	11		
Misleading Allow Header	27		
Method treated as GET		3	
Error Code in Response			16
Status OK, but entity indicates that method is not supported			3
Runtime Error			5



TRACE

	Unconditional	Conditional	Non-compliant
Implemented OK	32		
Not Implemented	26		
Message body not included		15	
Status OK, but message body indicates that the method is not supported			3
Method treated as GET			2
Error Code in Response			11
Runtime Error			11



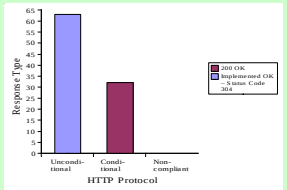
CONNECT

- CONNECT is used for setting up a secure tunnel in 60 out of 100 websites.
- 40 websites send authentication information using POST, mostly without encryption.

Read-Only Methods

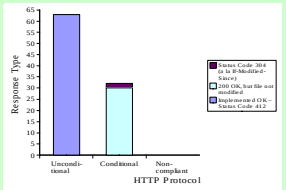
GET with If-Modified-Since Header

	Unconditional	Conditional
Implemented OK – Status code 304	63	
200 OK		32



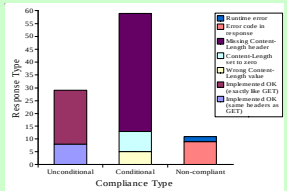
GET with If-Unmodified-Since Header

	Unconditional	Conditional
Implemented OK – Status code 412	63	
200 OK		30
Status code 304 (a la If-Modified-Since)		2



HEAD

	Unconditional	Conditional	Non-compliant
Implemented OK (same headers as GET)	8		
Implemented OK (exactly like GET)	21		
Wrong Content-Length value		5	
Content-Length set to zero		8	
Missing Content-Length header		46	
Error Code in Response			9
Runtime Error			2



Write Methods

POST, PUT and DELETE

Write methods cannot be tested uniformly. So we did not test these methods.

Relationship to Web Services

Supporting Methods

OPTIONS is meant to dynamically discover capabilities of a resource.
Used as a mechanism to "confuse the enemy".
Like UDDI, it does not work.
TRACE is meant for tracing and debugging.
It is seldom used due to security concerns.
CONNECT is a lightweight alternative to Web service standards for secure communication.
This illustrates the inefficacy of Web service standards.

Read-Only Methods

GET is used for retrieving a resource representation.
All implementations are unconditionally compliant.
RESTful Web services misuse GET for modifying server content.
Conditional GET is used to improve caching.
It is implemented correctly by most websites.
Web services should use it for caching.
HEAD is used for retrieving meta-data.
Not useful for Web services.

Write Methods

POST is used for updating data on the server.
PUT is meant for creating new resources.
DELETE is meant for deleting a resource.
In practice, POST is used for all of these tasks and more (SOAP uses POST for everything).
RESTful Web services claim that POST, PUT and DELETE must be used according to the HTTP standard.
But, the current state of practice is likely to remain unchanged.