Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 30, 2014

J. Arwe
S. Speicher
IBM
E. Wilde
EMC Corporation

July 29, 2013

The Accept-Post HTTP Header
draft-wilde-accept-post-00

Abstract

This specification defines a new HTTP response header field Accept-Post, which indicates server support for specific media types for entity bodies in HTTP POST requests.

Note to Readers

This draft should be discussed on the apps-discuss mailing list [9].

Online access to all versions and files is available on github [10].

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 http://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 January 30, 2014.

Copyright Notice

Table of Contents

1. Introduction

This specification defines a new HTTP response header field Accept-Post, which indicates server support for specific media types for entity bodies in HTTP POST requests. This header field is comparable to the Accept-Patch response header field specified together with the HTTP PATCH method [4] (notice, however, that while Accept-Patch is defined to only list specific media types, Accept-Post reuses the "media range" concept of HTTP's Accept header and thus allows media type wildcards as well).

2. Terminology

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. The Accept-Post Response Header Field

This specification introduces a new response header field Accept-Post used to specify the document formats accepted by the server in HTTP POST requests. Accept-Post SHOULD appear in the OPTIONS response for any resource that supports the use of the POST method. The presence of the Accept-Post header in response to any method is an implicit indication that POST is allowed on the resource identified by the Request-URI. The presence of a specific document format in this header indicates that that specific format is allowed on the resource identified by the Request-URI.

The syntax for Accept-Post headers, using the ABNF syntax defined in Section 2.1 of RFC 2616 [2], is given by the following definition.
Accept-Post = "Accept-Post" ":" #(media-range [accept-params])

The Accept-Post header specifies a media range as defined by HTTP [2]. The media range specifies a type of representation that can be POSTed to the Request-URI.

The app:accept element is similar to the HTTP Accept request header field [2]. Media type parameters are allowed within Accept-Post, but Accept-Post has no notion of preference - "accept-params" or "q" arguments, as specified in Section 14.1 of [2], are not significant.

4. IANA Considerations

4.1. The Accept-Post Response Header

The Accept-Post response header should be added to the permanent registry of message header fields (see [3]).

5. Examples

Accept-Post extends the way in which interaction information can be exposed in HTTP itself. The following sections contain some examples how this can be used in concrete HTTP-based services.

5.1. Linked Data Platform

The Linked Data Platform (LDP) [5] describes a set of best practices and simple approach for a read-write Linked Data architecture, based on HTTP access to Web resources that describe their state using the RDF data model. LDP defines LDP Containers (LDPC) and LDP Resources (LDPR). Adding new LDPRs to an LDPC is done by sending an HTTP POST request to the LDPC. An LDPC can constrain the media types it is accepting for these POST requests, and MUST expose its support for accepted media types via Accept-Post.

In fact, the Accept-Post header was initially developed within the W3C's LDP Working Group (LDPWG), see Appendix A for acknowledgements. It was then decided that the header itself might be useful in other contexts as well, and thus should be specified in a standalone document.

5.2. Atom Publishing Protocol

The Atom Publishing Protocol (AtomPub) [6] defines a model of interacting with collections and members, based on representations using the Atom [7] syntax. AtomPub allows clients to create new collection members by using HTTP POST, with the request being sent to the collection URI. AtomPub servers can limit the media types they accept in these POST requests, and the accepted media types are listed in an "AtomPub service document".

The Accept-Post header field does allow an AtomPub server to advertise its support for specific media types in interactions with the collection resource, without the need for a client to locate the service document and interact with it. This increases the visibility of the "POST to Create" model of AtomPub, and makes it easier for clients to find out about the capabilities of a specific collection.

While the AtomPub protocol cannot be changed retroactively, this additional way of exposing interaction guidance could make it easier

for clients to interact with AtomPub services that do support the Accept-Post header field. For those that do not support Accept-Post, clients would still have to rely on using the information contained in the service document (including the sometimes tricky issue of how to locate the service document for a given collection).

5.3. Additional Information in Error Responses

If a client POSTs an unsupported POST document, it is possible for the server to use Accept-Post to indicate the supported media types. These can be specified using a 415 (Unsupported Media Type) response when the client sends a POST document format that the server does not support for the resource identified by the Request-URI. Such a response then MAY include an Accept-Post response header notify the client what POST document media types are supported.

6. Implementation Status

Note to RFC Editor: Please remove this section before publication.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 6982 [8]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 6982, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

...

7. Security Considerations

The Accept-Post header may expose information that a server would prefer to not publish. In such a case, a server can simply stop

exposing the header, in which case HTTP interactions would be back to the level of standard HTTP (i.e., with no indication what kind of media types a resource accepts in POST requests).

8. Open Issues

Note to RFC Editor: Please remove this section before publication.

- Accept-Post currently uses the "media range" concept of HTTP's Accept header field. An alternative would be only support fully specified media types, which is what the Accept-Patch header field is doing. This latter solution is more constrained, and fails to address some uses cases, such as AtomPub's way of exposing collection support for POST requests.

9. References

9.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

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

[3] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, September 2004.

9.2. Non-Normative References

[4] Dusseault, L. and J. Snell, "PATCH Method for HTTP", RFC 5789, March 2010.

[5] Speicher, S. and J. Arwe, "Linked Data Platform 1.0", World Wide Web Consortium WD WD-ldp-20130307, March 2013, <http://www.w3.org/TR/2013/WD-ldp-20130307>.

[6] Gregorio, J. and B. de hOra, "The Atom Publishing Protocol", RFC 5023, October 2007.

[7] Nottingham, M., Ed. and R. Sayre, Ed., "The Atom Syndication Format", RFC 4287, December 2005.

[8] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code:

The Implementation Status Section", RFC 6982, July 2013.

URIs

[9] <https://www.ietf.org/mailman/listinfo/apps-discuss>

[10] <https://github.com/dret/I-D/tree/master/accept-post>

Appendix A. Acknowledgements

This work has been done in the context of the W3C Linked Data Platform Working Group (LDPWG) [5]. Thanks for comments and suggestions provided by the working group as a whole.

Authors' Addresses

John Arwe
IBM

Email: johnarwe@us.ibm.com

Steve Speicher
IBM

Email: sspeiche@us.ibm.com

Erik Wilde
EMC Corporation
6801 Koll Center Parkway
Pleasanton, CA 94566
U.S.A.

Phone: +1-925-6006244
Email: erik.wilde@emc.com
URI: http://dret.net/netdret/