core
Internet-Draft
Intended status: Informational
Expires: September 9, 2015

P. van der Stok
Consultant
A. Sehgal
Jacobs University

March 8, 2015

Patch Method for Constrained Application Protocol (CoAP)
draft-vanderstok-core-patch-00

Abstract

Several applications (for example see [I-D.vanderstok-core-comi]) which extend the Constrained Application Protocol [RFC7252] (CoAP) need to perform partial resource modifications. The existing CoAP PUT method only allows a complete replacement of a resource. This proposal adds a new CoAP method, PATCH, to modify an existing CoAP resource partially.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This specification defines the new Constrained Application Protocol (CoAP) [RFC7252] method, PATCH, which is used to apply partial modifications to a resource.

PATCH is also specified for HTTP in [RFC5789]. Most of the motivation for PATCH described in [RFC5789] also applies here.

The PUT method exists to overwrite a resource with completely new contents, and cannot be used to perform partial changes. When using PUT for partial changes, proxies and caches, and even clients and servers, may get confused as to the result of the operation. PATCH was mentioned in an early design stage of CoAP but was deemed unnecessarily complicated. With the arrival of the Constrained Management Interface (CoMI) protocol, [I-D.vanderstok-core-comi], the need to do partial changes to resources specified with YANG becomes more acute. Applications might wish to make to changes to parts of a YANG data resource, and transferring all data associated with a YANG data resource unnecessarily burdens the constrained communication medium.

This document relies on knowledge of the PATCH specification for HTTP [RFC5789]. This document provides extracts from [RFC5789] to make independent reading possible.

1.1. Requirements Language

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 [RFC2119].

1.2. Terminology and Acronyms

This document uses terminology defined in [RFC5789] and [RFC7252].

2. Patch Method

The PATCH method requests that a set of changes described in the request payload is applied to the resource identified by the Request-URI. The set of changes is represented in a format identified by a media type. If the Request-URI does not point to an existing resource, the server MAY create a new resource with that URI, resulting in a 2.01 (Created) Response Code. Restrictions to a PATCH can be made by including the If-Match or If-None-Match options in the request (see Section 5.10.8.1 and 5.10.8.2 of [RFC7252]). If the resource could not be created or modified, then an appropriate Error Response Code SHOULD be sent.

The difference between the PUT and PATCH requests is extensively documented in [RFC5789].

PATCH is not safe but idempotent conformant to CoAP PUT specified in [RFC7252], Section 5.8.3.

PATCH can use confirmable (CON) or Non-confirmable (NON) CoAP requests. It is recommended to use the CON version of the PATCH command.

A PATCH request is idempotent to prevent bad outcomes from collisions between two PATCH requests on the same resource in a similar time frame. These collisions can be detected with the MessageId and the source end-point provided by the CoAP protocol (see section 4.5 of [RFC7252].

The server MUST apply the entire set of changes atomically and never provide a partially modified representation to a concurrently executed GET request. Given the constrained nature of the servers, most servers will only execute CoAP requests consecutively, thus preventing a concurrent partial overlapping of request modifications. In general, modifications MUST NOT be executed when an error occurs or only a partial execution is possible. The atomicity requirement holds for all directly affected (sub)resources. See "Response

Codes", Section 2.2, for details on status codes and possible error conditions.

If the request passes through a cache and the Request-URI identifies one or more currently cached responses, those responses SHOULD be treated as being stale. A cached PATCH response can only be used to respond to subsequent GET requests; it MUST NOT be used to respond to other methods (in particular, PATCH).

There is no guarantee that a resource can be modified with PATCH. Servers are required to support a subset of the content formats as specified in sections 12.3 and 5.10.3 of [RFC7252]. Servers MUST ensure that a received PATCH payload is appropriate for the type of resource identified by the Request-URI.

Clients MUST choose to use PATCH rather than PUT when the request affects (sub)resources of a given resource.

2.1. A Simple PATCH Example

```
                      REQ: PATCH
                           coap://www.example.com/object/sub1
                           payload with changes
                      RET:
                           CoAP 2.04 Changed
```

This example illustrates use of a hypothetical PATCH on the sub resource /object/sub1 of the existing resource "object". The 2.04 (Changed) response code is conforms with the CoAP PUT method.

2.2. Response Codes

PATCH for CoAP adopts the response codes as specified in sections 5.9 and 12.1.2 of [RFC7252].

2.3. Option Numbers

PATCH for CoAP adopts the option numbers as specified in sections 5.10 and 12.2 of [RFC7252].

2.4. Securing PATCH

PATCH is secured following the CoAP recommendations as specified in section 9 of [RFC7252]. When more appropriate security techniques are standardized for CoAP, PATCH can also be secured by those new techniques.

3. Error Handling

A PATCH request may fail under certain known conditions. These situations should be dealt with as expressed below.

Malformed PATCH payload: If a server determines that the payload provided with a PATCH request is not properly formatted, it can return a 4.00 (Bad Request) CoAP error. The definition of a malformed payload depends upon the CoAP Content-Format specified with the request.

Unsupported PATCH payload: In case a client sends payload that is inappropriate for the resource identified by the Request-URI, the server can return a 4.15 (Unsupported Content-Format) CoAP error. The server can determine if the payload is supported by checking the CoAP Content-Format specified with the request.

Unprocessable request: This situation occurs when the payload of a PATCH request is determined as valid, i.e. well-formed and supported, however, the server is unable to or incapable of processing the request. The server can return a X.XX CoAP error. Such a scenario might include situations when:

* the server has insufficient computing resources to complete the request successfully,

* the resource specified in the request becomes invalid by applying the payload,

* modifying a resource leads to a conflicting state.

In case there are more specific errors that provide more insight into the problem, then those should be used.

Resource not found: The 4.04 (Not Found) error should be returned in case the payload of a PATCH request cannot be applied to a non-existent resource.

Failed precondition: In case the client uses the conditional If-Match or If-None-Match option to define a precondition for the PATCH request, and that precondition fails, then the server can return the 4.12 (Precondition Failed) CoAP error.

Request too large: If the payload of the PATCH request is larger than a CoAP server can process, then it can return the 4.13 (Request Entity Too Large) CoAP error.

Conflicting modification: In situations when a server detects possible conflicting modifications and no precondition is defined in the requests, the server can return a X.XX CoAP status.

Conflicting state: If the modification specified by a PATCH request cannot be applied to a resource in its current state, or causes the resource to enter an inconsistent state the server can return the X.XX CoAP status. Such a situation might be encountered when a structural modification is applied to a configuration data-store, but the structures being modified do not exist or lead the device into an inconsistent state if the modifications are made.

Concurrent modification: Resource constrained devices might need to process requests in the order they are received. In case requests are received concurrently to modify the same resource but they cannot be queued, the server can return a X.XX CoAP status.

It is possible that other error situations, not mentioned here, are encountered by a CoAP server while processing the PATCH request. In these situations other appropriate CoAP status codes can also be returned.

4. Security Considerations

This section analyses the possible threats to the CoAP PATCH protocol. It is meant to inform protocol and application developers about the security limitations of CoAP PATCH as described in this document. The security consideration of section 15 of [RFC2616], section 11 of [RFC7252], and section 5 of [RFC5789] also apply.

The security considerations for PATCH are nearly identical to the security considerations for PUT ([RFC7252]). Whatever mechanisms are used for PUT can be used for PATCH as well.

5. IANA Considerations

The entry with name PATCH in the sub-registry, "CoAP Method Codes", is 0.05. the addition will follow the "IETF Review or IESG Approval" procedure as described in [RFC5226].

6. Acknowledgements

This document reflects discussions and remarks from several individuals including (in alphabetical order):

7. Change log

When published as a RFC, this section needs to be removed.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2616] 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.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, September 2004.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5789] Dusseault, L. and J. Snell, "PATCH Method for HTTP", RFC 5789, March 2010.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, June 2014.

8.2. Informative References

[I-D.vanderstok-core-comi] Stok, P., Greevenbosch, B., Bierman, A., Schoenwaelder, J., and A. Sehgal, "CoAP Management Interface", draft-vanderstok-core-comi-06 (work in progress), February 2015.

Authors' Addresses

Peter van der Stok
Consultant

Email: consultancy@vanderstok.org

Anuj Sehgal
Jacobs University
Campus Ring 1
Bremen 28759
Germany

Email: s.anuj@jacobs-university.de