CoRE Working Group	K. Hartke
Internet-Draft	Universitaet Bremen TZI
Intended status: Informational	March 07, 2011
Expires: September 08, 2011	

Accessing HTTP resources over CoAP
draft-hartke-core-coap-http-00

Abstract

The CoAP/HTTP proxying mechanism defined in this document enables Constrained Application Protocol (CoAP) clients to access a Hypertext Transfer Protocol (HTTP) resource by delegating the task of retrieving resource representations and manipulating resources to a CoAP intermediary.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

The Constrained Application Protocol (CoAP) [I-D.ietf-core-coap] allows clients to specify an arbitrary, absolute Uniform Resource Identifier (URI) [RFC3986] in requests to CoAP proxies. This document defines what it means to perform a CoAP request on a Hypertext Transfer Protocol (HTTP) URI [RFC2616].
The resulting CoAP/HTTP proxying mechanism enables CoAP clients to access HTTP resources over CoAP by delegating the task of retrieving resource representations and manipulating resources to a CoAP intermediary. This relieves the client from the burden of implementing any HTTP functionality itself.
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].

2. CoAP/HTTP Proxying

If a request contains a Proxy-URI Option with an 'http' or 'https' URI [RFC2616], then the receiving CoAP end-point (called "the proxy" henceforth) is requested to perform the operation specified by the

request method on the indicated HTTP resource and return the result to the client.
This section specifies for any CoAP request the CoAP response that the proxy should return to the client. How the proxy actually satisfies the request is an implementation detail, although the typical case is expected to be the proxy translating and forwarding the request to an HTTP origin server (see Section 3).
Since HTTP and CoAP share the basic set of request methods, performing a CoAP request on an HTTP resource is not so different from performing it on a CoAP resource. The meanings of the individual CoAP methods when performed on HTTP resources are explained below.
If the proxy is unable or unwilling to service a request with an HTTP URI, a 5.05 (Proxying Not Supported) response MUST be returned to the client. If the proxy services the request by interacting with a third party (such as the HTTP origin server) and is unable to obtain a result within a reasonable timeframe, a 5.04 (Gateway Timeout) response MUST be returned; if a result can be obtained but is not understood, a 5.02 (Bad Gateway) response MUST be returned.

2.1. GET

The GET method requests the proxy to return a representation of the HTTP resource identified by the request URI.
Upon success, a 2.00 (OK) response SHOULD be returned. The payload of the response MUST be a representation of the target HTTP resource, and the Content-Type Option be set accordingly. The response MUST indicate a Max-Age value that is no greater than the remaining time the representation can be considered fresh. If the HTTP entity has an entity tag, the proxy SHOULD include an ETag Option in the response and process ETag Options in requests as described below.
A client can influence the processing of a GET request by including the following options:

Lifetime: The request MAY include a Lifetime Option [I-D.ietf-core-observe]. This requests the proxy to notify the client of any changes to the state of the target HTTP resource. If the proxy is unable to detect changes to the resource or is unwilling to establish an observation relationship, the proxy MUST silently ignore the option. Otherwise, the proxy establishes an observation relationship as described in [I-D.ietf-core-observe] and, whenever the state of the HTTP resource changes, sends a 2.00 (OK) notification response to the client.

ETag: The request MAY include one or more ETag Options, identifying responses that the client has stored. This requests the proxy to send a 2.03 (Valid) response whenever it would send a 2.00 (OK) response with an entity tag in the requested set otherwise. (See also Section 3.2.)

Block:
The request MAY include a Block Option [I-D.ietf-core-block], specifying the block the client is interested in. This requests the proxy to send only the requested block in a 2.00 (OK) response instead of the whole representation. (See also Section 3.3.)

2.2. PUT

The PUT method requests the proxy to update or create the HTTP resource identified by the request URI with the enclosed representation.
If a new resource is created at the request URI, a 2.01 (Created) response MUST be returned to the client. If an existing resource is modified, a 2.04 (Changed) response MUST be returned to indicate successful completion of the request.
A client can influence the processing of a PUT request by including the following options:

Block: The request MAY include a Block Option, indicating a block-wise transfer as described in [I-D.ietf-core-block]. (See also Section 3.3.)

2.3. DELETE

The DELETE method requests the proxy to delete the HTTP resource identified by the request URI at the HTTP origin server.
A 2.02 (Deleted) response MUST be returned to client upon success or if the resource does not exist at the time of the request.
No further options are defined to enable the client to influence the processing of a DELETE request.

2.4. POST

The POST method requests the proxy to have the representation enclosed in the request be processed by the HTTP origin server. The actual function performed by the POST method is determined by the origin server and dependent on the resource identified by the request URI.
If the action performed by the POST method does not result in a resource that can be identified by a URI, a 2.04 (Changed) response MUST be returned to the client. If a resource has been created on the origin server, a 2.01 (Created) response MUST be returned.
A client can influence the processing of a PUT request by including the following options:

Block: The request MAY include a Block Option, indicating a block-wise transfer as described in [I-D.ietf-core-block]. (See also Section 3.3.)

3. Implementation Considerations

The CoAP/HTTP proxying requirements described in Section 2 deliberately make no restrictions on how a proxy implementation actually satisfies a request to produce the expected response. Obviously, these requirements would not be very useful if they could not be implemented. Also, some of the proxies will run on nodes that exhibit some constraints and may use networks that also are constrained.
In general, an implementation will translate and forward CoAP requests to the HTTP origin server and translate back HTTP responses to CoAP responses, typically employing a certain amount of caching to make this translation more efficient. This non-normative section gives some hints for implementing the translation.

3.1. Payloads and Media Types

CoAP supports only a subset of media types. A proxy should convert payloads and approximate content-types as closely as possible. For example, if a HTTP server returns a resource representation in "text/plain; charset=iso-8859-1" format, the proxy should convert the payload to "text/plain; charset=utf-8" format. If conversion is not possible, the proxy can specify a media type of "application/octet-stream".

3.2. Max-Age and ETag Options

The proxy can determine the Max-Age Option for responses to GET requests by calculating the freshness lifetime (see Section 13.2.4 of [RFC2616]) of the HTTP resource representation retrieved. The Max-Age Option for responses to POST, PUT or DELETE requests should always be set to 0.
The proxy can assign entity tags to responses it sends to a client. These can be generated locally, if the proxy employs a cache, or be derived from the ETag header field in a response from the HTTP origin server, in which case the proxy can optimize future requests to the HTTP by using Conditional Requests. Note that CoAP does not support weak entity tags.

3.3. Block-wise Transfers

The proxy can optimize the processing of many requests. For example, when a GET request includes a Block Option, the proxy can use HTTP Range Requests to retrieve only the requested block(s) of the HTTP resource from the origin server instead of retrieving the whole resource.
Section 2 makes no restrictions on whether the proxy attempts to re-assemble all block-wise PUT requests into a single HTTP PUT request or attempts to push each request forward by making use of the HTTP Content-Range header field (which is not widely implemented for PUT outside WebDAV). In either case, the handling chosen MUST be reflected in the More bit sent back in the CoAP response as specified in [I-D.ietf-core-block].

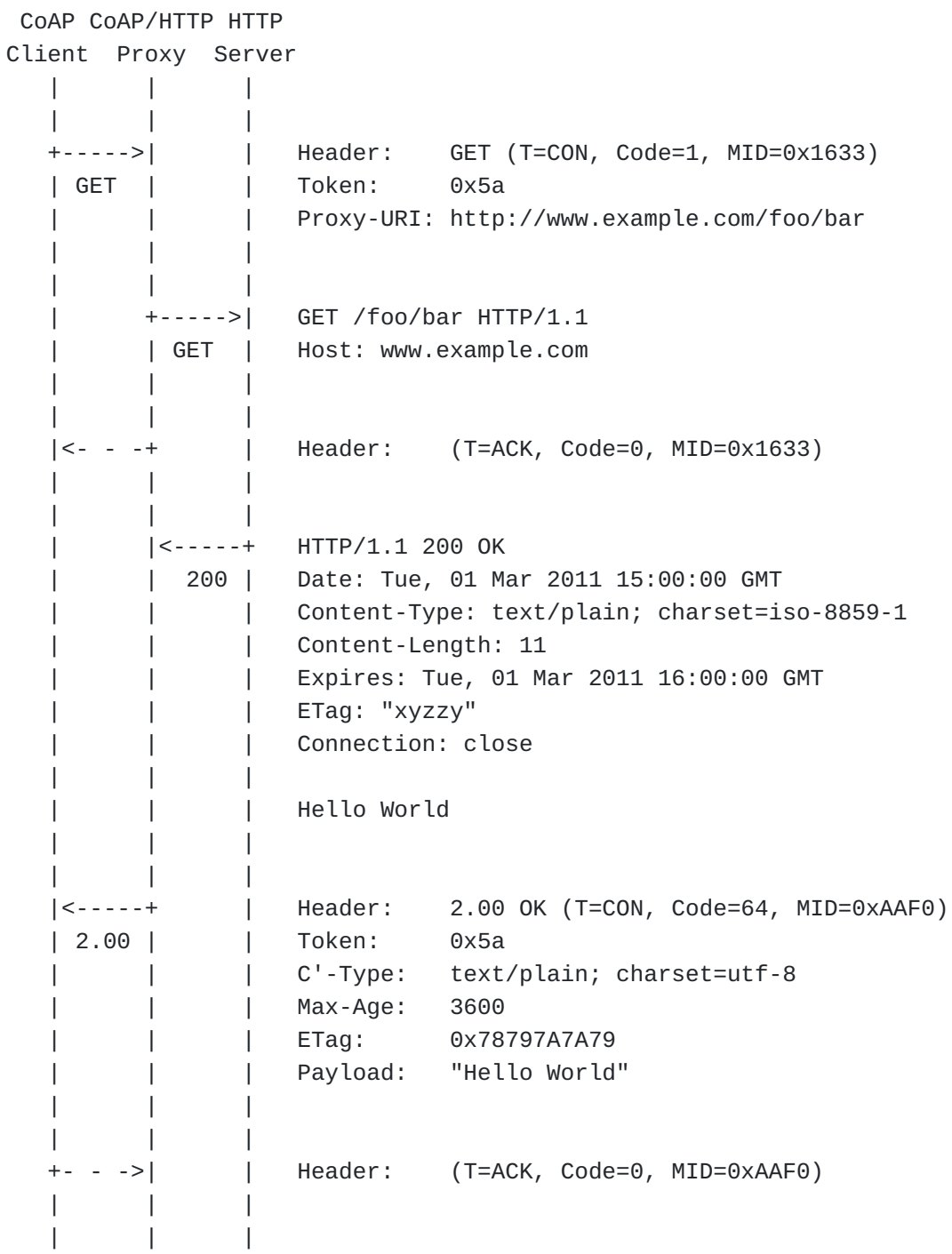
For POST, similar considerations as with PUT apply, except that HTTP Content-Range is even less widely implemented with HTTP POST.

3.4. HTTP Status Codes 1xx and 3xx

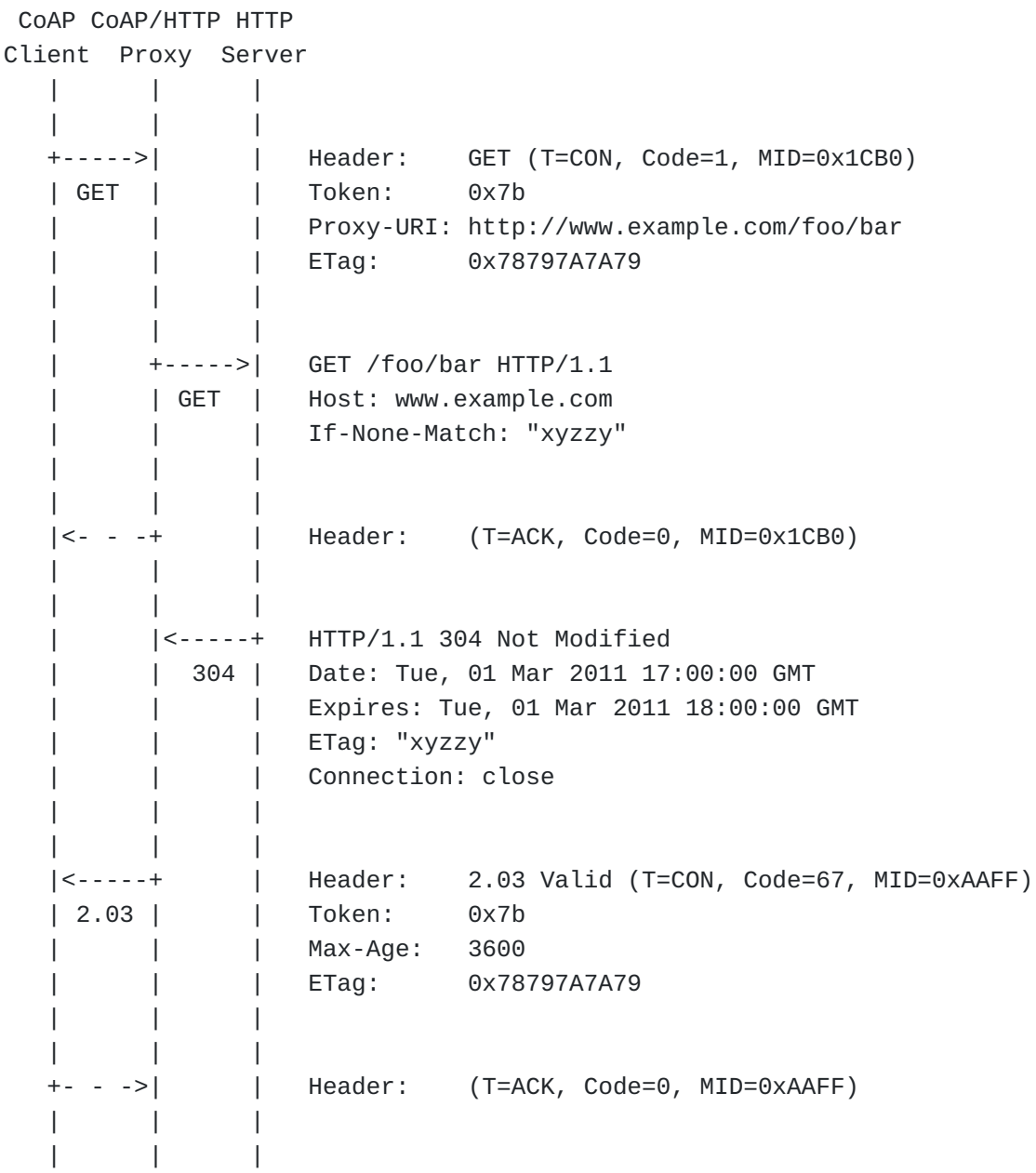
CoAP does not have provisional responses (HTTP Status Codes 1xx) or responses indicating that further action needs to be taken (HTTP Status Codes 3xx). When a proxy receives such a response from the HTTP server, the response should cause the proxy to complete the request, for example, by following redirects. If the proxy is unable or unwilling to do so, it can return a 5.02 (Bad Gateway) error.

3.5. Example

The example in Figure 1 shows a possible implementation of a basic CoAP GET request with an HTTP URI as the value of a Proxy-URI Option. The proxy retrieves a representation of the target resource from the HTTP origin server. It converts the payload to UTF-8 charset, calculates the Max-Age Option from the Expires header field, and derives an entity-tag from the ETag header field.



The example in Figure 2 builds on the previous example and shows a possible implementation of a GET request that includes a previously returned ETag Option. The proxy makes a Conditional Request to the HTTP origin server by including an If-None-Match header field in the HTTP GET request. The CoAP response indicates that the response stored by the client is fresh. It includes a Max-Age Option calculated from the HTTP response's Expires header field.



4. Security Considerations

When CoAP NoSec mode is used, a CoAP to HTTP proxy provides a way for a UDP sender to initiate an HTTP transfer, possibly using a fake source address, making the original request untraceable in general. (I.e. the usual minimal protection afforded to HTTP proxies by the TCP three-way handshake does not apply.) CoAP/HTTP proxies MUST be used in a way that they don't attract attackers that want to exploit this, e.g., by requiring communication security and/or by running them in a protected network.

The security considerations of section 15.7 of RFC 2616 apply and MUST be heeded. (TODO: Add further text about the perils of proxies, e.g., cache poisoning.)

5. IANA Considerations

This document does not require IANA action.

6. Acknowledgements

Some text for Section 3 and Section 4 was contributed by Carsten Bormann.

7. 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.
[RFC3986]	Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
[I-D.ietf-core-coap]	Shelby, Z, Hartke, K, Bormann, C and B Frank, "Constrained Application Protocol (CoAP)", Internet-Draft draft-ietf-core-coap-04, January 2011.
[I-D.ietf-core-block]	Shelby, Z and C Bormann, "Blockwise transfers in CoAP", Internet-Draft draft-ietf-core-block-01, January 2011.
[I-D.ietf-core-observe]	Hartke, K and Z Shelby, "Observing Resources in CoAP", Internet-Draft draft-ietf-core-observe-01, February 2011.

Author's Address

Klaus Hartke Hartke Universitaet Bremen TZI Postfach 330440 Bremen, D-28359 Germany Phone: +49-421-218-63905 EMail: hartke@tzi.org