Abstract

This document defines ASCII encoding method for Constrained Application Protocol (CoAP). CoAP has defined a binary encoding method to exchange requests and responses between constrained nodes or gateways. However, some communication modules in constrained nodes support ASCII data only. These modules are cheaper and widely deployed. To support these modules, this document describes the method to convert binary CoAP messages to base64 styled ASCII messages.

Status of this Memo

This Internet-Draft is submitted to IETF 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 14, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document.
publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
Table of Contents

1. Introduction ...................................................... 4
2. Termination .......................................................... 4
3. Base64 encoding for CoAP ........................................ 5
   3.1. Overview .................................................... 5
   3.2. Examples ..................................................... 6
4. Security Considerations ......................................... 7
5. IANA Considerations ............................................. 7
6. Acknowledgments .................................................. 8
7. References ........................................................ 8
   7.1. Normative References ....................................... 8
1. Introduction

The Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained networks. Current CoAP draft document [COAP] describes a binary encoding method for exchanging requests and response between nodes. Binary encoding is compact and fast. However, in real deployment environment, many constrained nodes use ASCII encoding only. Many ZigBee communication modules and WiFi communication modules for embedded devices allow ASCII characters only. Some modules can send a limited size (about 50 bytes) message at a time. Because these communication modules are cheaper and widely deployed, ASCII version of CoAP is required.

To support ASCII encoding, an approach to design new encoding method may be inefficient. Therefore, this document reuses Base64 encoding/decoding. Binary CoAP messages are converted to ASCII messages before transmission. ASCII encoded messages are converted to binary CoAP messages again after receiving via communication channel.

This document replaces some characters in the standard Base64 character set because of escape characters, for example, "+" character. To support fragmentation and reassembly if a communication module allows small size of message for transmitting or receiving, this document adds "#" character at end of the each messages.

Because Base64 encoding/decoding logic is very simple, it requires just small code size for it. The code for it could be written with about 120 lines source code.

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 [RFC2119].
3. Base64 encoding for CoAP

3.1. Overview

A CoAP node MAY consist of the CPU module and the communication module logically or physically. Some communication modules do not support binary encoded messages. To use the communication modules which do not support binary messages, the CPU module in CoAP node SHOULD encode a binary coded CoAP message to an ASCII based text message before passing the message to the communication module. After the communication module passes the message to the CPU module, the CPU module SHOULD decode the ASCII based text message to the binary CoAP message. A CoAP node can use a normal CoAP protocol software stack with this translation.

In this document, we use Base64 encoding[BASE64]. Base64 is a binary-to-text encoding scheme that represent binary data in an ASCII string by a radix-64 representation. There are some variations in Base64 characters. For example, MIME’s Base64 uses A-Z, a-z, 0-9, ‘+’, and ‘/’. And, ‘=’ is used as a padding character. The number of encoded bytes per original bytes is approximately 4/3 (33% overhead). In this document, we use Base64url encoding that ‘-’(minus) character is used instead of ‘+’(plus) and ‘_’(underline) character is used instead of ‘/’ (slash). This is a URL and file name safe version of Base64 variations. And, some communication module may translate ‘+’ as an escape character. Therefore, the ‘+’(plus) character SHOULD NOT be used for the communication.

In this document, ‘#’ is used as the end of message. Therefore, the CPU module in CoAP node SHOULD add ‘#’ character at the end of Base64 encoded message. It helps the message fragmentation and reassembly. The sender side MAY send the partial encoded messages. The receiver side SHOULD reassemble the messages until ‘#’ is received. The sender MAY send some white space characters, i.e., carriage return, line feed, space, or tab, between fragmented messages. The receiver SHOULD ignore all characters besides Base64url character set and ‘=’ character.

The following figure shows how ASCII encoding method for Constrained Application Protocol works. The Base64 encoding/decoding software module runs independently from CoAP protocol stack.
3.2. Examples

In the appendix A in CoAP draft-18, a confirmable request and piggybacked response message example is described.

Client        Server

|                |                |
|                | +-----<---------
|                | | Header: GET (T=CON, Code=0.01, MID=0x7d34) |
|                | | GET   | Uri-Path: "temperature" |
|                | | <-----+]
|                | | 2.05 | Payload: "22.3 C"

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 1 | 0 | 0 | GET=1 | MID=0x7d34 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 11 | 11 | "temperature" (11 B) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 1 | 2 | 0 | 2.05=69 | MID=0x7d34 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 1 1 1 1 1 1 | "22.3 C" (6 B) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Original binary encoded CoAP Request:
40 01 7d 34 bb 74 65 6d 70 65 72 61 74 75 72 65

Original binary encoded CoAP Response:
60 45 7d 34 ff 32 32 2e 33 43

These messages are converted to the Base64 encoded messages as the follows.

Text encoded CoAP Request:
QAF9NLt0ZW1wZXJhdHVyZQ==#

This message is written in hex string as the follows.
51 41 46 39 4e 4c 70 30 5a 57 31 77 5a 58 4a 68 64
48 56 79 5a 51 3d 3d 23

Text encoded CoAP Response:
YEV9NP8yMi4zIEM=#

This message is written in hex string as the follows.
59 45 56 39 4e 50 38 79 4d 69 34 7a 49 45 4d 3d 23

4. Security Considerations

TODO - fill in

5. IANA Considerations

TODO - fill in
6. Acknowledgments

TODO - fill in

7. References

7.1. Normative References


Author’s Addresses

Seok-Kap Ko
ETRI
176-11 Cheomdan Gwagi-ro, Buk-gu, Gwangju, 500-480, Korea
Phone: +82-62-970-6677
Email: softgear@etri.re.kr

Ilkyun Park
ETRI
176-11 Cheomdan Gwagi-ro, Buk-gu, Gwangju, 500-480, Korea
Phone: +82-62-970-6651
Email: ikpark@etri.re.kr

Seung-Hun Oh
ETRI
176-11 Cheomdan Gwagi-ro, Buk-gu, Gwangju, 500-480, Korea
Phone: +82-62-970-6655
Email: osh93@etri.re.kr

Byung-Tak Lee
ETRI
176-11 Cheomdan Gwagi-ro, Buk-gu, Gwangju, 500-480, Korea
Phone: +82-62-970-6624
Email: bytelee@etri.re.kr