core
Internet-Draft
Intended status: Standards Track
Expires: February 19, 2017

P. van der Stok
consultant
A. Bierman
YumaWorks, Inc.

August 18, 2016

CBOR format for YIDs
draft-vanderstok-core-cbor-yid-00

Abstract

This document describes CoAP content formats to transport CBOR encoded YANG objects which use numeric YANG object identifiers. YANG objects are serialized according to the YANG to CBOR encoding. The object identifier is composed of a module number followed by a yang name numeric identifier

Note

Discussion and suggestions for improvement are requested, and should be sent to core@ietf.org.

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 February 19, 2017.

Copyright Notice

Table of Contents

1. Introduction

The Concise Binary Object Representation (CBOR, [RFC7049]) defines a binary representation for data that can be described using an extension of the JSON data model [RFC7159].

Numeric YANG Identifiers (YID), described in [I-D.bierman-core-yid], represent YANG object names as numeric identifiers. They are structured according to a module_id, local_id pattern.

This document specifies two new content formats. The content-format, called "application/cbor+yid", follows the YANG to CBOR encoding specified in [I-D.ietf-core-yang-cbor], and the YANG name string to numeric identifier specified in [I-D.bierman-core-yid].

The content format called "application/merge-patch+cbor+yid" is an extension of the format specified in [I-D.bormann-appsawg-cbor-merge-patch]

The delta encoding specified in [I-D.ietf-core-yang-cbor], is used throughout.

The format is primarily intended to describe the numeric identifiers of the CBOR encoded YANG objects transported with CoAP [RFC7252] methods, as specified in [I-D.vanderstok-core-comi].

1.1. 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 [RFC2119].

Readers of this specification should be familiar with all the terms and concepts discussed in [RFC2578].

The following terms are defined in the NETCONF protocol [RFC6241]: client, configuration data, datastore, and server.

The following terms are defined in the YANG data modelling language [RFC6020]: container, data node, key, key leaf, leaf, leaf-list, and list.

The following terms are defined in RESTCONF protocol [I-D.ietf-netconf-restconf]: data resource, data-store resource, edit operation, query parameter, target resource, and unified data-store.

The following terms are defined in YANG ID protocol [I-D.bierman-core-yid]: YANG module schema tree, YANG ID, YANG Anchor, YANG Anchor subtree, and YANG Identifier File.

The following terms are defined in this document:

To be defined term: TODO: specification.

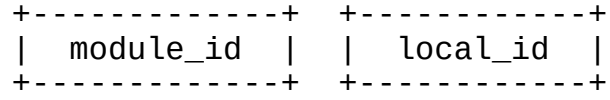
2. Design Objectives

This work is motivated by the need to minimize the size of numeric object identifiers within protocol messages representing YANG data, encoded according to the CBOR binary format. Both a single complete resource description used with GET and PUT, and partial resource descriptions used with FETCH, PATCH and iPATCH are specified.

3. YANG Identifier

The numeric YANG identifier used in this document is specified in [I-D.bierman-core-yid]. A YANG Identifier is constructed of 2 fields:

YANG Identifier Format:



The meaning of the identifiers in the above figure is:

module_id: This is a mandatory number that uniquely identifies a module with a size of "module-bits" bits.

local_id: This is a number that uniquely defines a YANG name within a module with a size of "local-bits" bits. Local identifiers can be one of two formats: YANG Hash or Manually Assigned.

3.1. YANG Identifier encoding

A YANG Identifier (YID) is constructed from module_id and local_id according to the formula:

YID = (module_id*2^local-bits)+ local_id.

The CBOR encoding uses maps (major type 5) composed of pair (map.key, map.value). Maps can be the parent of children maps. A map can have 1 or zero parents. A map without parents is called a root.

Using delta encoding (see [I-D.ietf-core-yang-cbor]), the YID is encoded in the maps according to the following rules:

If the map is a root, YID(root) = root.key.

If the map has a parent, YID(map) = map.key + YID(parent).

3.2. Length of CBOR major type 0 and 1

Local_ids are represented as unsigned integers (major type 0) or negative integers (major type 1). The two types in CBOR are either represented as one byte, or as multiple bytes, where the first byte indicates the length of the value. This has a consequence for the

local_id value size. The smallest possible value is 23 (5 bits), encoded as one byte. For larger local_id values, their size should be 8, 16, 32, and 64 bits, encoded in CBOR by as many bits + 8 bits.

When the local_id is based on hashes, clashes can occur. Below two tables give the probability that at least one clash occurs or at least two clashes occur for a given hash size in bits and the number of names to be hashed.

Number of names	5 bits	8 bits	16 bits	32 bits
5	3,1E-01	3,9E-02	1,5E-04	2,3E-09
10	1	1,7E-01	6,9E-04	1,1E-08
20	1	7,4E-01	2,9E-03	4,4E-08
40	1	1	1,2E-02	1,8E-07
100	1	1	7,6E-02	1,2E-06
200	1	1	3,0E-01	4,6E-06

Table 1: Probability of one or more clashes

Number of names	5 bits	8 bits	16 bits	32 bits
5	5,9E-02	9,2E-04	1,4E-08	3,3E-18
10	1	2,5E-02	3,8E-07	8,8E-17
20	1	5,0E-01	7,6E-06	1,8E-15
40	1	1	1,4E-04	3,1E-14
100	1	1	5,6E-03	1,3E-12
200	1	1	9,1E-02	1,1E-10

Table 2: Probability of more than 2 entries equal clashes

It is clear that for hash size of 5 bits, more than 5 names will certainly provoke one or more clashes. When the number of bits should be as small as possible, an assignment of numbers per hand is

favored. 16 bit hashes can safely be used when less than 50 names are present in the module. 32 bit hashes are recommended when more than a few 1000 names occur in a module.

4. CBOR format to encode YID payload

The general format in Diagnostic JSON is shown below:

```
[
{ module_id*2^module-bits: { key: value .... }},
{ module_id*2^module-bits: { key: value .... }},
.............................
]
```

This represents an array of maps, where each map in the array represents a module. When there is one module, the array is removed and a single map remains. The module value: module_id*2^module-bits helps to reduce the key value sizes when multiple root maps follow. When a single root map follows, the module value can be removed.

Two identifier forms are discussed in the two sections below:

4.1. module form encoding

The cbor-yid format is composed of an array (major type 4) of one or more maps (major type 5) called "module maps".

The array of module maps collapses to one module map when there is only one module map to transport. The module_id*2^module-bits is represented as an unsigned integer (major type 0).

The key is represented as an unsigned integer (major type 0) or negative integer (major type 1), where key = YID - module_id*2^module-bits. YID is the YANG identifier of the YANG object encoded with (key, value) pair.

4.2. single root encoding

In the special case that there is one module with one root, the cbor-yid format is composed of one map (major type 5) called "object map" as shown below.

```
{ YID: value}
```

YID is the YANG Identifier of the YANG root map encoded with the (YID, value) pair. YID = module_id*(2^module-bits) + local_id. The value of the map follows the standard CBOR to YANG encoding.

5. YID addition to merge-patch+cbor

The general format in Diagnostic JSON is hown below:

```
[
{ module_id*2^module-bits: { merge-patch+cbor+yid format... }},
{ module_id*2^module-bits: { merge-patch+cbor+yid format --- }},
.........................
]
```

The figure represents an array of module maps, where each module map in the array represents a module payload encoded in the content format application/merge-patch+cbor. When there is one module the array is removed and a single module remains.

The merge-patch+cbor+yid format follows the merge-patch+cbor encoding as specified in [I-D.bormann-appsawg-cbor-merge-patch] where each numeric YANG identifier is encoded with the map keys as described in Section 3.1.

6. YID CBOR Examples

This section show some examples based on the ietf-system module, or on ietf-interfaces module.

6.1. single module form

Consider that module ietf-system has been attributed the unique identifier 24. Assume the YANG hash size to be set to 6 bits. The YANG hash values for ’clock’, ’current-datetime’, and ’boot-datetime’ are calculated by constructing the schema node identifier for the objects, and then calculating the 5 bit murmur3 hash values (shown in parentheses) in the figure below. The YIDs are calculated by adding 24*2^6 = 0xc00, leading to:

/ietf-system:system-state/clock (0x11) YID=0xc11
/ietf-system:system-state/clock/current-datetime (0xb) YID=0xc0b
/ietf-system:system-state/clock/boot-datetime (0x8) YID=0xc08

Consider the request for the values of the clock container, as specified with JSON below.

```
REQ: GET example.com/c/system-state/clock
   (Content-Format: application/json)

RES: 2.05 Content (Content-Format: application/cbor+yid)
{ "system-state": {
      "clock" : {
         "current-datetime" : "2014-10-26T12:16:51Z",
         "boot-datetime" : "2014-10-21T03:00:00Z"
      }
   }
}
```

Assuming that there are no hash clashes in the module, knowing that only one module is concerned, using the numeric identifiers above, and using 64 bit encoding [RFC4648] for the URI, the request looks like:

```
REQ: GET example.com/c/CAR
   (Content-Format: application/cbor+yid)

RES: 2.05 Content (Content-Format: application/cbor+yid)
{ 0xc00: {
    0x11 : {
      -0x6 : "2014-10-26T12:16:51Z",
      -0x9 : "2014-10-21T03:00:00Z"
      }
   }
}
```

Which creates the corresponding CBOR code to be transported:

```
a1                                      # map(1)
   19 0c00                              # unsigned(3072)
   a1                                   # map(1)
      11                                # unsigned(17)
      a2                                # map(2)
         25                             # negative(5)
         74                             # text(20)
            323031342d31302d32365431323a31363a35315a
                                        # "2014-10-26T12:16:51Z"
         28                             # negative(8)
         74                             # text(20)
            323031342d31302d32315430333a30303a30305a
                                        # "2014-10-21T03:00:00Z"
```

The example shows that the map keys of the objects are encoded as single bytes, because they their absolute value is smaller than 23. The module identifier takes 12 bits and is encoded as two bytes.

6.2. Single root form encoding

The example of Section 6 can also be encoded in Single root form encoding by stripping the module identifier. The request in diagnostic JSON looks like:

```
REQ: GET example.com/c/CAR
   (Content-Format: application/cbor+yid)

RES: 2.05 Content (Content-Format: application/cbor+yid)
{ 0xc11 : {
      -0x6 : "2014-10-26T12:16:51Z",
      -0x9 : "2014-10-21T03:00:00Z"
      }
}
```

In this simple example, the single root form requires less bytes than the single module form. When there are multiple root maps within the module, the module form generates less code because the module is only encoded once and subtracted from the key values of the root maps.

6.3. Multi module form encoding

Suppose there are multiple modules that are requested with the FETCH method. The request is for "clock" from module ietf-system and the other for "neighbor" from module ietf-interfaces. Assume the module identifiers for ietf-system and ietf-interfaces are 24 and 7 respectively.

The YANG hash values for ’neighbor’, ’ip’, and ’link-layer-address’ are calculated by constructing the schema node identifier for the objects, calculating the 7 bit murmur3 hash values (shown in parenthesis below, followed by the module identifier calculation 7*2^8 = 0x700. This results in:

```
/ietf-interfaces:interfaces/interface/ietf-ip:ipv6/neighbor
    (0x78) YID = 0x778
/ietf-interfaces:interfaces/interface/ietf-ip:ipv6/neighbor/ip
    (0x40) YID = 0x740
/ietf-interfaces:interfaces/interface/ietf-ip:ipv6/neighbor/
    link-layer-address  (0x47) YID =0x747
```

The Fetch request and response, assuming no clashes, are shown below:

```
REQ: FETCH example.com/mg/
   (Content-Format: application/cbor+yid)
[{0x30: 0x11}, { 0x700: 0x78}]

RES: 2.05 Content (Content-Format: application/cbor+yid)
[
{ 0xc00: {
    0x11 : {
      -0x6 : "2014-10-26T12:16:51Z",
      -0x9 : "2014-10-21T03:00:00Z"
      }
   }
},
{ 0x700: {
   0x78 : {
       -0x38: "fe80::200:f8ff:fe21:67cf",
       -0x31: "00:00::10:01:23:45"
       }
    }
}
]
```

The corresponding CBOR code is shown in Appendix A.

6.4. Multi form encoding with clashes

As in Section 6.3, multiple modules that are requested with the FETCH method. The request is for "clock" from module ietf-system and the other for "neighbor" from module ietf-interfaces. Assume the module identifiers for ietf-system and ietf-interfaces are 24 and 7 respectively.

Assume the YANG hash size is still set to 6 bits. Assume a clash occurred during the hashing of the identifier, /ietf-system:system-state/clock/current-datetime of module ietf-system. Another identifier in ietf-system also has hash value 0xb. The rehashed identifier for "clock" is set to 0x21 with bit 6 set to 1.

The new YANG hash values for ’clock’, ’current-datetime’, and ’boot-datetime’ are shown within parentheses in the figure below:

```
/ietf-system:system-state/clock (0x11) YID = 0xc11
/ietf-system:system-state/clock/current-datetime ( 0x21) YID = 0xc21
/ietf-system:system-state/clock/boot-datetime (0x8) YID = 0xc08
```

The Fetch request, including a rehash value, with answer is shown below:

```
REQ: FETCH example.com/mg/
   (Content-Format: application/cbor+yid)
[{0xc00: 0x11}, { 0x700: 0x78}]

RES: 2.05 Content (Content-Format: application/cbor+yid)
[
{ 0xc00: {
      0x11 : {
        0x10 : "2014-10-26T12:16:51Z",
        -0x9 : "2014-10-21T03:00:00Z"
      }
   }
},
{ 0x700: {
   0x78 : {
       -0x38:  "fe80::200:f8ff:fe21:67cf",
       -0x31:  "00:00::10:01:23:45"
       }
    }
}
]
```

The identifier 0xb of current-datetime has been replaced with the rehash value 0x21.

The corresponding CBOR code is shown in Appendix A.

7. IANA Considerations

The Internet media type [RFC6838] for CBOR YANG Identifier are:

1.

* application/cbor+yid.
* Type name: application
* Subtype name: cbor+yid
* Required parameters: None
* Optional parameters: None

* Encoding considerations: Resources that use the "application/cbor+yid" media type are required to conform to the "application/cbor" media type and are therefore subject to the same encoding considerations specified in Section 7.3 of [RFC7049].

* Security considerations: As defined in this specification

* Published specification: This specification.

* Applications that use this media type: None currently known.

2.

* application/merge-patch+cbor+yid.

* Type name: application

* Subtype name: merge-patch+cbor+yid

* Required parameters: None

* Optional parameters: None

* Encoding considerations: Resources that use the "application/merge-patch+cbor+yid" media type are required to conform to the "application/cbor" media type and are therefore subject to the same encoding considerations specified in Section 7.3 of [RFC7049].

* Security considerations: As defined in this specification

* Published specification: This specification.

* Applications that use this media type: None currently known.

Additional information:

Magic number(s): N/A

File extension(s): N/A

Macintosh file type code(s): N/A

Person and email address to contact for further information: IESG

Intended usage: COMMON

Restrictions on usage: None

Author: Peter van der Stok < consultancy@vanderstok.org >

Change controller: IESG

8. Security considerations

Security considerations to be done.

9. Acknowledgements

We are very grateful to Bert Greevenbosch who suggested the use of hashes and specified the use of murmur3.

10. Changelog

xxx

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <http://www.rfc-editor.org/info/rfc6838>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <http://www.rfc-editor.org/info/rfc7049>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <http://www.rfc-editor.org/info/rfc7252>.

[I-D.ietf-core-yang-cbor] Veillette, M., Pelov, A., Somaraju, A., Turner, R., and A. Minaburo, "CBOR Encoding of Data Modeled with YANG", draft-ietf-core-yang-cbor-02 (work in progress), July 2016.

[I-D.bormann-appsawg-cbor-merge-patch] Bormann, C. and P. Stok, "CBOR Merge Patch", draft-bormann-appsawg-cbor-merge-patch-00 (work in progress), March 2016.

[I-D.bierman-core-yid] Bierman, A. and P. Stok, "Numeric YANG Identifiers", draft-bierman-core-yid-00 (work in progress), August 2016.

11.2. Informative References

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, DOI 10.17487/RFC2578, April 1999, <http://www.rfc-editor.org/info/rfc2578>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <http://www.rfc-editor.org/info/rfc4648>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[I-D.ietf-netconf-restconf] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-16 (work in progress), August 2016.

[I-D.vanderstok-core-comi] Stok, P. and A. Bierman, "CoAP Management Interface", draft-vanderstok-core-comi-09 (work in progress), March 2016.

Appendix A. CBOR examples

The CBOR code belonging to Section 6.2 is shown below:

```
a1                                      # map(1)
   19 0c11                              # unsigned(3089)
   a2                                   # map(2)
      25                                # negative(5)
      74                                # text(20)
         323031342d31302d32365431323a31363a35315a
                                        # "2014-10-26T12:16:51Z"
      28                                # negative(8)
      74                                # text(20)
         323031342d31302d32315430333a30303a30305a
                                        # "2014-10-21T03:00:00Z"
```

The CBOR code belonging to Section 6.3 is shown below:

```
82                                      # array(2)
   a1                                   # map(1)
      19 0c00                           # unsigned(3072)
      a1                                # map(1)
         11                             # unsigned(17)
         a2                             # map(2)
            25                          # negative(5)
            74                          # text(20)
               323031342d31302d32365431323a31363a35315a
                                        # "2014-10-26T12:16:51Z"
            28                          # negative(8)
            74                          # text(20)
               323031342d31302d32315430333a30303a30305a
                                        # "2014-10-21T03:00:00Z"
   a1                                   # map(1)
      19 0700                           # unsigned(1792)
      a1                                # map(1)
         18 78                          # unsigned(120)
         a2                             # map(2)
            38 37                       # negative(55)
            78 18                       # text(24)
               666538303a3a3230303a663866663a666532313a36376366
                                  # "fe80::200:f8ff:fe21:67cf"
            38 30                       # negative(48)
            72                          # text(18)
               30303a30303a3a31303a30313a32333a3435
                                        # "00:00::10:01:23:45"
```

Because the hash codes for the object identifiers of the ietf-interface module are 1 byte long, the corresponding CBOR encoding takes 2 bytes.

The CBOR code belonging to Section 6.4 is shown below:

```
   82                                      # array(2)
      a1                                   # map(1)
         19 0c00                           # unsigned(3072)
         a1                                # map(1)
            11                             # unsigned(17)
            a2                             # map(2)
               10                          # unsigned(16)
               74                          # text(20)
                  323031342d31302d32365431323a31363a35315a
                                           # "2014-10-26T12:16:51Z"
               28                          # negative(8)
               74                          # text(20)
                  323031342d31302d32315430333a30303a30305a
                                           # "2014-10-21T03:00:00Z"
      a1                                   # map(1)
         19 0700                           # unsigned(1792)
         a1                                # map(1)
            18 78                          # unsigned(120)
            a2                             # map(2)
               38 37                       # negative(55)
               78 18                       # text(24)
                  666538303a3a3230303a663866663a666532313a36376366
                                       # "fe80::200:f8ff:fe21:67cf"
               38 30                       # negative(48)
               72                          # text(18)
                  30303a30303a3a31303a30313a32333a3435
                                           # "00:00::10:01:23:45"
```

The byte 0b, numeric identifier of current-datetime, has been replaced by the one byte 20 due to the hash clash.

Authors' Addresses

Peter van der Stok
consultant

Phone: +31-492474673 (Netherlands), +33-966015248 (France)
Email: consultancy@vanderstok.org
URI: www.vanderstok.org

Andy Bierman
YumaWorks, Inc.
685 Cochran St.
Suite #160
Simi Valley, CA 93065
USA

Email: andy@yumaworks.com