Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: September 12, 2016

M. Veillette, Ed.
Trilliant Networks Inc.
A. Pelov, Ed.
Acklio
A. Somaraju
Tridonic GmbH & Co KG
R. Turner
Landis+Gyr
A. Minaburo
Acklio

March 11, 2016

CBOR Encoding of Data Modeled with YANG
draft-veillette-core-yang-cbor-mapping-00

Abstract

This document defines encoding rules for serializing configuration data, state data, RPC input and RPC output, Action input, Action output and notifications defined within YANG modules using the Concise Binary Object Representation (CBOR) [RFC7049].

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 September 12, 2016.

Copyright Notice

Copyright (c) 2016 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. 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

The specification of the YANG 1.1 data modelling language [I-D.ietf-netmod-rfc6020bis] defines only an XML encoding for data instances, i.e. contents of configuration datastores, state data, RPC inputs and outputs, action inputs and outputs, and event notifications.

A new set of encoding rules has been defined to allow the use of the same data models in environments based on the JavaScript Object Notation (JSON) Data Interchange Format [RFC7159]. This is accomplished in the JSON Encoding of Data Modeled with YANG specification [I-D.ietf-netmod-yang-json].

The aim of this document is to define a set of encoding rules for the Concise Binary Object Representation (CBOR) [RFC7049]. The resulting encoding is more compact compared to XML and JSON and more suitable for Constrained Nodes and/or Constrained Networks as defined by [RFC7228].

2. Terminology and Notation

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].

The following terms are defined in [I-D.ietf-netmod-rfc6020bis]:

- action
- anydata
- anyxml
- data node
- data tree
- module
- notification
- RPC
- schema node
- schema tree
- submodule

The following terms are defined in [I-D.ietf-netmod-yang-json]:

- member name
- name of an identity

The following term is defined in [I-D.vanderstok-core-comi]:

- YANG hash

This specification also makes use of the following terminology:

- child: A schema node defined within a collection such as a container, a list, a case, a notification, an RPC input, an RPC output, an action input, an action output.

- delta : Difference between the SID assigned to the current schema node and the SID assigned to the parent.

- parent: The collection in which a schema node is defined.

- structured identifier or SID: Unsigned integer used to identify different YANG items.

2.1. CBOR diagnostic notation

Within this document, CBOR binary contents are represented using an equivalent textual form called CBOR diagnostic notation as defined in [RFC7049] section 6. This notation is used strictly for documentation purposes and is never used in the data serialization.

CBOR content	CBOR type	Diagnostic notation	Example	CBOR encoding
Unsigned integer	0	Decimal digits	123	18 7b
Negative integer	1	Decimal digits prefixed by a minus sign	-123	38 7a
Byte string	2	Hexadecimal value enclosed between single quotes and prefixed by an 'h'	h'f15c'	42 f15c
Text string	3	String of Unicode characters enclosed between double quotes	"txt"	63 747874
Array	4	Comma-separated list of values within square brackets	[1, 2]	82 01 02
Map	5	Comma-separated list of key : value pairs within curly braces	{ 1: 123, 2: 456 }	a2 01187b 021901c8
Boolean	7/20	false	false	f4
	7/21	true	true	f5
Null	7/22	null	null	f6
Not assigned	7/23	undefined	undefined	f7

Within this document, comments are allowed in CBOR diagnostic notation. Any characters after a Pound sign ('#') outside of a string, up to the end of the line, are treated as a comment.

3. Properties of the CBOR Encoding

This document defines CBOR encoding rules for YANG schema trees and their subtrees.

Basic schema nodes such as leaf, leaf-list, list, anydata and anyxml can be encoded standalone. In this case, only the value of this schema node is encoded in CBOR. Identification of this value need to be provided by some external means when needed.

A collection such as container, list instance, notification, RPC input, RPC output, action input and action output is serialized using a CBOR map in which each child schema node is encoded using a key and a value. This specification supports three type of keys; SID as defined in [I-D.somaraju-core-sid], member names as defined in [I-D.ietf-netmod-yang-json] and YANG hash as defined by

[I-D.vanderstok-core-comi]. Each of these key type is encoded using a specific CBOR type which allows their interpretation during the deserialization process. The end user of this mapping specification can mandate the use of a specific key type or a specific subset of key types.

In order to minimize the size of the encoded data, the proposed mapping does not make use of any meta-information beyond those natively supported by CBOR. For instance, CBOR tags are not used for any of the proposed mapping. It is expected that entities generating and decoding CBOR contents have enough knowledge about the information processed in order to perform the expected task without the need of such extra meta-information. The CoAP Content-Format Option, or an HTTP Content-Type header field, conveys that the data is YANG-encoded CBOR in the first place.

4. Encoding of YANG Schema Node Instances

Schema node instances defined using the YANG modeling language are encoded using CBOR [RFC7049] based on the rules defined in this section. We assume that the reader is already familiar with both YANG [I-D.ietf-netmod-rfc6020bis] and CBOR [RFC7049].

4.1. The "leaf" Schema Node

Leafs MUST be encoded based on the encoding rules specified in Section 5.

4.2. The "container" Schema Node

Collections such as containers, list instances, notifications, RPC inputs, RPC outputs, action inputs and action outputs MUST be encoded using a CBOR map data item (major type 5). A map is comprised of pairs of data items, with each data item consisting of a key and a value. This specification supports three type of keys; SID as defined in [I-D.somaraju-core-sid], member names as defined in [I-D.ietf-netmod-yang-json] and YANG hash as defined by [I-D.vanderstok-core-comi].

SIDs as keys

Keys implemented using SIDs MUST be encoded using a CBOR unsigned integer (major type 0) and set to the delta value of the associated SID. Delta values are computed as follows:

o The delta value is equal to the SID of the current schema node minus the SID of the parent schema node. When no parent exists in the context of use of this container, the delta is set to the SID of the current schema node (a parent with SID equal to zero is assumed).

o Delta values may result in a negative number, clients and servers MUST support negative deltas.

Member names as keys

Keys implemented using member names MUST be encoded using a CBOR text string data item (major type 3). A namespace-qualified member name MUST be used for all members of a top-level collection, and then also whenever the namespaces of the schema node and its parent are different. In all other cases, the simple form of the member name MUST be used. Member names and namespaces are defined in [I-D.ietf-netmod-yang-json] section 4.

YANG hashes as keys

Keys implemented using YANG hashes MUST be encoded using a CBOR byte string data item (major type 2).

Values MUST be encoded using the appropriate rules defined in Section 4 and Section 5.

Definition example [RFC7317]:

```
typedef date-and-time {
  type string {
    pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}(\.\d+)?(Z|[\+\-]
             \d{2}:\d{2})';
  }
}

container system {
  leaf hostname {
    type inet:domain-name;

  container clock {
    leaf current-datetime {
      type date-and-time;
    }

    leaf boot-datetime {
      type date-and-time;
    }
  }
}
```

SIDs example

This example is encoded using the SIDs defined in [I-D.somaraju-core-sid] Appendix C.

CBOR diagnostic notation:

```
{
  1708 : {                              # clock
    2 : "2015-10-02T14:47:24Z-05:00",   # current-datetime, SID 1710
    1 : "2015-09-15T09:12:58Z-05:00"    # boot-datetime, SID 1709
  }
}
```

CBOR encoding:

```
a1                                      # map(1)
   19 06ac                              # unsigned(1708)
   a2                                   # map(2)
      02                                # unsigned(2)
      78 1a                             # text(26)
      323031352d31302d30325431343a34373a32345a2d30353a3030
      01                                # unsigned(1)
      78 1a                             # text(26)
      323031352d30392d31355430393a31323a35385a2d30353a3030
```

Member names example

CBOR diagnostic notation:

```
{
  "ietf-system:clock" : {
    "current-datetime" : "2015-10-02T14:47:24Z-05:00",
    "boot-datetime" : "2015-09-15T09:12:58Z-05:00"
  }
}
```

CBOR encoding:

```
   a1                                          # map(1)
      71                                       # text(17)
         696574662d73797374656d3a636c6f636b    # "ietf-system:clock"
      a2                                       # map(2)
         70                                    # text(16)
            63757272656e742d6461746574696d65   # "current-datetime"
         78 1a                                 # text(26)
            323031352d31302d30325431343a34373a32345a2d30353a3030
         6d                                    # text(13)
            626f6f742d6461746574696d65         # "boot-datetime"
         78 1a                                 # text(26)
            323031352d30392d31355430393a31323a35385a2d30353a3030
```

YANG Hashes example

CBOR diagnostic notation:

```
   {
     h’334c67d9’ : {                                # clock
       h’047c468b’ : "2015-10-02T14:47:24Z-05:00",  # current-datetime
       h’2fe1a167’ : "2015-09-15T09:12:58Z-05:00"   # boot-datetime
     }
   }
```

CBOR encoding:

```
   a1                                      # map(1)
      44                                   # bytes(4)
         334c67d9
      a2                                   # map(2)
         44                                # bytes(4)
            047c468b
         78 1a                             # text(26)
            323031352d31302d30325431343a34373a32345a2d30353a3030
         44                                # bytes(4)
            2fe1a167
         78 1a                             # text(26)
            323031352d30392d31355430393a31323a35385a2d30353a3030
```

4.3. The "leaf-list" Schema Node

A leaf-list MUST be encoded using a CBOR array data item (major type 4). Each entry of this array MUST be encoded using the rules defined by the YANG type specified.

Definition example [RFC7317]:

```
typedef domain-name {
  type string {
    length "1..253";
    pattern '((([a-zA-Z0-9_]([a-zA-Z0-9\-_]){0,61})?[a-zA-Z0-9].)
             *([a-zA-Z0-9_]([a-zA-Z0-9\-_]){0,61})?[a-zA-Z0-9]\.?
             )|\.';
  }
}

leaf-list search {
  type domain-name;
  ordered-by user;
}
```

CBOR diagnostic notation: ["ietf.org", "ieee.org"]

CBOR encoding: 82 68 696574662e6f7267 68 696565652e6f7267

4.4. The "list" Schema Node

A list MUST be encoded using a CBOR array data item (major type 4). Each entry of this array is encoded using a CBOR map data item (major type 5) based on the same rules as a YANG container, see Section 4.2.

Definition example [RFC7317]:

```
   list server {
     key name;

     leaf name {
       type string;
     }
     choice transport {
       case udp {
         container udp {
           leaf address {
             type host;
             mandatory true;
           }
           leaf port {
             type port-number;
           }
         }
       }
     }
     leaf association-type {
       type enumeration {
         enum server;
         enum peer;
         enum pool;
       }
       default server;
     }
     leaf iburst {
       type boolean;
       default false;
     }
     leaf prefer {
       type boolean;
       default false;
     }
   }
```

SIDs example

SIDs used in this example are defined in [I-D.somaraju-core-sid] Appendix C. It is important to note that the protocol or method using this mapping may carry a parent SID or may have the knowledge of this parent SID based on its context. In these cases, delta encoding can be performed based on this parent SID which minimizes the size of the encoded data.

CBOR diagnostic notation:

```
[
  {
    1746 : "NRC TIC server",          # name
    1748 : {                          # udp
      1 : "tic.nrc.ca",               # address, SID 1749
      2 : 123                         # port, SID 1750
    },
    1744 : 0,                         # association-type
    1745 : false,                     # iburst
    1747 : true                       # prefer
  },
  {
    1746 : "NRC TAC server",          # name
    1748 : {                          # udp
      1 : "tac.nrc.ca"                # address, SID 1749
    }
  }
]
```

CBOR encoding:

```
82                                    # array(2)
   a5                                 # map(5)
      19 06d2                         # unsigned(1746)
      6e                              # text(14)
         4e524320544943207365727665722 # "NRC TIC server"
      19 06d4                         # unsigned(1748)
      a2                              # map(2)
         01                           # unsigned(1)
         6a                           # text(10)
            7469632e6e72632e6361      # "tic.nrc.ca"
         02                           # unsigned(2)
         18 7b                        # unsigned(123)
      19 06d0                         # unsigned(1744)
      00                              # unsigned(0)
      19 06d1                         # unsigned(1745)
      f4                              # primitive(20)
      19 06d3                         # unsigned(1747)
      f5                              # primitive(21)
   a2                                 # map(2)
      19 06d2                         # unsigned(1746)
      6e                              # text(14)
         4e524320544143207365727665722 # "NRC TAC server"
      19 06d4                         # unsigned(1748)
      a1                              # map(1)
         01                           # unsigned(1)
         6a                           # text(10)
            7461632e6e72632e6361      # "tac.nrc.ca"
```

Member names example

CBOR diagnostic notation:

```
[
  {
    "ietf-system:name" : "NRC TIC server",
    "ietf-system:udp" : {
      "address" : "tic.nrc.ca",
      "port" : 123
    },
    "ietf-system:association-type" : 0,
    "ietf-system:iburst" : false,
    "ietf-system:prefer" : true
  },
  {
    "ietf-system:name" : "NRC TAC server",
    "ietf-system:udp" : {
      "address" : "tac.nrc.ca"
    }
  }
]
```

CBOR encoding:

```
82                                           # array(2)
   a5                                        # map(5)
      70                                     # text(16)
         696574662d73797374656d3a6e616d65     # "ietf-system:name"
      6e                                     # text(14)
         4e524320544943207365727665722       # "NRC TIC server"
      6f                                     # text(15)
         696574662d73797374656d3a756470       # "ietf-system:udp"
      a2                                     # map(2)
         67                                  # text(7)
            61646472657373                   # "address"
         6a                                  # text(10)
            7469632e6e72632e6361             # "tic.nrc.ca"
         64                                  # text(4)
            706f7274                         # "port"
         18 7b                               # unsigned(123)
      78 1c                                  # text(28)
         696574662d73797374656d3a6173736f63696174696f6e2d74797065
      00                                     # unsigned(0)
      72                                     # text(18)
         696574662d73797374656d3a696275727374 # "ietf-system:iburst"
      f4                                     # primitive(20)
      72                                     # text(18)
         696574662d73797374656d3a707265666572 # "ietf-system:prefer"
      f5                                     # primitive(21)
   a2                                        # map(2)
      70                                     # text(16)
         696574662d73797374656d3a6e616d65     # "ietf-system:name"
      6e                                     # text(14)
         4e524320544143207365727665722       # "NRC TAC server"
      6f                                     # text(15)
         696574662d73797374656d3a756470       # "ietf-system:udp"
      a1                                     # map(1)
         67                                  # text(7)
            61646472657373                   # "address"
         6a                                  # text(10)
            7461632e6e72632e6361             # "tac.nrc.ca"
```

YANG hashes example

CBOR diagnostic notation:

```
   [
     {
       h'06c32032' : "NRC TIC server",           # name
       h'11889c84' : {                           # udp
         h'3158c529' : "tic.nrc.ca",             # address
         h'34492d05' : 123                       # port
       },
       h'2c2c2ccf' : 0,                          # association-type
       h'1058dc5d' : false,                      # iburst
       h'390e346a' : true                        # prefer
     },
     {
       h'06c32032' : "NRC TAC server",           # name
       h'11889c84' : {                           # udp
         h'3158c529' : "tac.nrc.ca"              # address
       }
     }
   ]
```

CBOR encoding:

```
   82                                   # array(2)
      a5                                # map(5)
         44                             # bytes(4)
            06c32032"
         6e                             # text(14)
            4e524320544943207365727665 72 # "NRC TIC server"
         44                             # bytes(4)
            11889c84 "
         a2                             # map(2)
            44                          # bytes(4)
               3158c529
            6a                          # text(10)
               7469632e6e72632e6361     # "tic.nrc.ca"
            44                          # bytes(4)
               34492d05
            18 7b                       # unsigned(123)
         44                             # bytes(4)
            2c2c2ccf
         00                             # unsigned(0)
         44                             # bytes(4)
            1058dc5d
         f4                             # primitive(20)
         44                             # bytes(4)
            390e346a
         f5                             # primitive(21)
      a2                                # map(2)
         44                             # bytes(4)
            06c32032
         6e                             # text(14)
            4e5243205441432073657276 6572 # "NRC TAC server"
         44                             # bytes(4)
            11889c84
         a1                             # map(1)
            44                          # bytes(4)
               3158c529
            6a                          # text(10)
               7461632e6e72632e6361     # "tac.nrc.ca"
```

4.5. The "anydata" Schema Node

An anydata serves as a container for an arbitrary set of schema nodes that otherwise appear as normal YANG-modeled data. An anydata instance is encoded using the same rules as a container, i.e., CBOR map. The requirement that anydata content can be modeled by YANG implies the following:

- Keys MUST be set to valid SIDs, member names or YANG hashes. This rule applies to the key of the anydata node and the key of any inner schema node.

- The CBOR array MUST contain either unique scalar values (as a leaf-list, see Section 4.3), or maps (as a list, see Section 4.4).

- Values MUST follow the encoding rules of one of the datatypes listed in Section 5.

4.6. The "anyxml" Schema Node

An anyxml instance is encoded as a CBOR key/value pair. The key of the anyxml schema node MUST be a valid SID, member name or YANG hash but the value is unrestricted, i.e., the value can be any CBOR encoded content.

5. Representing YANG Data Types in CBOR

5.1. The unsigned integer Types

Leafs of type uint8, uint16, uint32 and uint64 MUST be encoded using a CBOR unsigned integer data item (major type 0).

Definition example [RFC7277]:

```
leaf mtu {
  type uint16 {
    range "68..max";
  }
}
```

CBOR diagnostic notation: 1280

CBOR encoding: 19 0500

5.2. The integer Types

Leafs of type int8, int16, int32 and int64 MUST be encoded using either CBOR unsigned integer (major type 0) or CBOR signed integer (major type 1), depending on the actual value.

Definition example [RFC7317]:

```
   leaf timezone-utc-offset {
     type int16 {
       range "-1500 .. 1500";
     }
   }
```

CBOR diagnostic notation: -300

CBOR encoding: 39 012b

5.3. The "decimal64" Type

Leafs of type decimal64 MUST be encoded using either CBOR unsigned integer (major type 0) or CBOR signed integer (major type 1), depending on the actual value. The position of the decimal point is defined by the fraction-digits YANG statement and is not available in the CBOR encoding.

Definition example [RFC7317]:

```
   leaf my-decimal {
     type decimal64 {
       fraction-digits 2;
       range "1 .. 3.14 | 10 | 20..max";
     }
   }
```

CBOR diagnostic notation: 257 (Represents decimal value 2.57)

CBOR encoding: 19 0101

5.4. The "string" Type

Leafs of type string MUST be encoded using a CBOR text string data item (major type 3).

Definition example [RFC7223]:

```
   leaf name {
     type string;
   }
```

CBOR diagnostic notation: "eth0"

CBOR encoding: 64 65746830

5.5. The "boolean" Type

Leafs of type boolean MUST be encoded using a CBOR true (major type 7, additional information 21) or false data item (major type 7, additional information 20).

Definition example [RFC7317]:

```
leaf enabled {
  type boolean;
}
```

CBOR diagnostic notation: true

CBOR encoding: f5

5.6. The "enumeration" Type

Leafs of type enumeration MUST be encoded using a CBOR unsigned integer data item (major type 0).

Definition example [RFC7317]:

```
leaf oper-status {
  type enumeration {
    enum up { value 1; }
    enum down { value 2; }
    enum testing { value 3; }
    enum unknown { value 4; }
    enum dormant { value 5; }
    enum not-present { value 6; }
    enum lower-layer-down { value 7; }
  }
}
```

CBOR diagnostic notation: 3 (Represents enumeration value "testing")

CBOR encoding: 03

5.7. The "bits" Type

Leafs of type bits MUST be encoded using a CBOR byte string data item (major type 2). Bits position 0 to 7 are assigned to the first byte within the byte string, bits 8 to 15 to the second byte, and subsequent bytes are assigned similarly. Within each byte, bits are assigned from least to most significant.

Definition example [I-D.ietf-netmod-rfc6020bis]:

```
   leaf mybits {
     type bits {
       bit disable-nagle {
         position 0;
       }
       bit auto-sense-speed {
         position 1;
       }
       bit 10-Mb-only {
         position 2;
       }
     }
   }
```

CBOR diagnostic notation: h’05’ (Represents bits disable-nagle and 10-Mb-only set)

CBOR encoding: 41 05

5.8. The "binary" Type

Leafs of type binary MUST be encoded using a CBOR byte string data item (major type 2).

Definition example:

```
   leaf aes128-key {
     type binary {
       length 16;
     }
   }
```

CBOR diagnostic notation: h’1f1ce6a3f42660d888d92a4d8030476e’

CBOR encoding: 50 1f1ce6a3f42660d888d92a4d8030476e

5.9. The "leafref" Type

Leafs of type leafref MUST be encoded using the rules of the schema node referenced by the "path" YANG statement.

Definition example [RFC7223]:

```
typedef interface-state-ref {
  type leafref {
    path "/interfaces-state/interface/name";
  }
}

container interfaces-state {
  list interface {
    key "name";
    leaf name {
      type string;
    }
    leaf-list higher-layer-if {
      type interface-state-ref;
    }
  }
}
```

CBOR diagnostic notation: "eth1"

CBOR encoding: 64 65746831

5.10. The "identityref" Type

This specification supports two approaches for encoding identityref, a SID as defined in [I-D.somaraju-core-sid] or a name as defined in [I-D.ietf-netmod-yang-json] section 6.8.

SIDs as identityref

SIDs are globally unique and may be used as identityref. This approach is both compact and simple to implement. When SIDs are used, identityref MUST be encoded using a CBOR unsigned integer data item (major type 0) and set to a SID allocated from a registered SID range.

Name as identityref

Alternatively, an identityref may be encoded using a name as defined in [I-D.ietf-netmod-yang-json] section 6.8. When names are used, identityref MUST be encoded using a CBOR text string data item (major type 3). If the identity is defined in another module than the leaf node containing the identityref value, the namespace-qualified form MUST be used. Otherwise, both the simple and namespace-qualified forms are permitted.

Definition example [RFC7223]:

```
identity interface-type {
}

identity iana-interface-type {
  base interface-type;
}

identity ethernetCsmacd {
  base iana-interface-type;
}

leaf type {
  type identityref {
    base interface-type;
  }
}
```

SIDs as identityref

Assuming that the identity "iana-if-type:ethernetCsmacd" has been assigned to the SID value 1179.

CBOR diagnostic notation: 1179

CBOR encoding: 19 049b

Name as identityref

CBOR diagnostic notation: "iana-if-type:ethernetCsmacd"

CBOR encoding: 78 1b
69616e612d69662d747970653a65746865726e657443736d616364

5.11. The "empty" Type

Leafs of type empty MUST be encoded using the CBOR null value (major type 7, additional information 22).

Definition example [RFC7277]:

```
leaf is-router {
  type empty;
}
```

CBOR diagnostic notation: null

CBOR encoding: f6

5.12. The "union" Type

Leafs of type union MUST be encoded using the rules associated with one of the types listed.

Definition example [RFC7317]:

```
typedef ipv4-address {
  type string {
  pattern '(([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.){3}
           ([0-9][1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])(%[\p{N}
           \p{L}]+)?';
  }
}

typedef ipv6-address {
  type string {
    pattern '((:|[0-9a-fA-F]{0,4}):)([0-9a-fA-F]{0,4}:){0,5}((([0-9a
             -fA-F]{0,4}:)?(:|[0-9a-fA-F]{0,4}))|(((25[0-5]|2[0-4][0
             -9]|[01]?[0-9]?[0-9])\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0
             -9]?[0-9])))(%[\p{N}\p{L}]+)?';
    pattern '(([^:]+:){6}(([^:]+:[^:]+)|(.*\..*)))|((([^:]+:)*[^:]+)
             ?::(([^:]+:)*[^:]+)?)(%.+)?';
  }
}

typedef ip-address {
  type union {
    type ipv4-address;
    type ipv6-address;
  }
}

leaf address {
  type inet:ip-address;
}
```

CBOR diagnostic notation: "2001:db8:a0b:12f0::1"

CBOR encoding: 74 323030313a6462383a6130623a313266303a3a31

5.13. The "instance-identifier" Type

This specification supports three approaches for encoding an instance-identifier, one based on SIDs as defined in [I-D.somaraju-core-sid], one based on names as defined in [I-D.ietf-netmod-yang-json] section 6.13 and one based on YANG hashes as defined in [I-D.vanderstok-core-comi].

SIDs as instance-identifier

SIDs uniquely identify a data node. For a single instance data node, the SID is sufficient to identify this instance. For a multi-instance data node, a SID is combined with the list key(s) to identify each instance of this data node within the YANG list(s).

Single instance data nodes MUST be encoded using a CBOR unsigned integer data item (major type 0) and set to the targeted data node SID.

Multi-instances data nodes MUST be encoded using a CBOR array data item (major type 4) containing the following entries:

- o The first entry MUST be encoded as a CBOR unsigned integer data item (major type 0) and set to the targeted data node SID.

- o The following entries MUST contain the value of each key required to identify the instance of the targeted data node. These keys MUST be ordered as defined in the "key" YANG statement, starting from top level list, and follow by each of the subordinate list(s).

When SIDs identify a YANG list, the presence of the key(s) for this list is optional. When the key(s) are present, the targeted instance within this list is selected. When the key(s) are absent, the entire YANG list is selected.

Names as instance-identifier

The use of names as instance-identifier is defined in [I-D.ietf-netmod-yang-json] section 6.11. The resulting xpath MUST be encoded using a CBOR text string data item (major type 3).

YANG hashes as instance-identifier

When YANG hashes are used, xpath can be compressed based on the method defined by [I-D.vanderstok-core-comi] sections 4.1.4.1 and 4.1.4.2.

Definition example [RFC7317]:

```
container system {

  leaf contact {
    type string;
  }

  leaf hostname {
    type inet:domain-name;
  }
}
```

First example based on SID

In this example, a field of type instance-identifier identifies the data node "/system/contact" (SID 1728).

```
1728
```

CBOR encoding:

```
19 06c0
```

First example based on name

Same example as above based on names.

```
"/ietf-system:system/contact"
```

CBOR encoding:

```
78 1c 2f20696574662d73797374656d3a73797374656d2f636f6e74616374
```

First example based on YANG hash

Same example assuming data node "/system/contact" is associated to YANG hash 0x09b06d17 or "JsG0X" in base64.

CBOR diagnostic notation:

```
"/JsG0X"
```

CBOR encoding:

```
66 2f4a73473058
```

Definition example [RFC7317]:

```
list user {
  key name;

  leaf name {
    type string;
  }
  leaf password {
    type ianach:crypt-hash;
  }

  list authorized-key {
    key name;

    leaf name {
      type string;
    }
    leaf algorithm {
      type string;
    }
    leaf key-data {
      type binary;
  }
}
```

Second example based on SID

In this example, a field of type instance-identifier identify the data node "/system/authentication/user/authorized-key/key-data" (SID 1721) for the user name "bob" and the authorized-key name "admin".

CBOR diagnostic notation:

```
[1721, "bob", "admin"]
```

CBOR encoding:

```
83                       # array(3)
   19 06b9               # unsigned(1721)
   63                    # text(3)
      626f62             # "bob"
   65                    # text(5)
      61646d696e         # "admin"
```

Second example based on name

Same example as above based on names.

CBOR diagnostic notation:

```
"/ietf-system:system/authentication/user[name='bob']/authorized-key
[name='admin']/key-data"
```

CBOR encoding:

```
78 59
   2f696574662d73797374656d3a73797374656d2f61757468656e74696361
   74696f6e2f757365725b6e616d653d27626f62275d2f617574686f72697a
   65642d6b65795b6e616d653d2761646d696e275d2f6b65792d64617461
```

Second example based on YANG hash

Same example assuming data node "/ietf-system:system/authentication/user/authorized-key/key-data" is associated to YANG hash 0x0d6e7afb or "Nbnr7" in base64.

CBOR diagnostic notation:

```
"/Nbnr7?keys=\"bob\",\"admin\""
```

CBOR encoding:

```
78 19 2f4e626e72373f6b6579733d22626f62222c2261646d696e22
```

Third example based on SID

This third example identify an instance within the list "/system/authentication/user" (SID 1717) corresponding to the user name "jack".

CBOR diagnostic notation:

```
[1717, "jack"]
```

CBOR encoding:

```
82                      # array(2)
   19 06b5              # unsigned(1717)
   64                   # text(4)
      6a61636b          # "jack"
```

Third example based on name

Same example as above based on names.

CBOR diagnostic notation:

```
"/ietf-system:system/authentication/user[name='bob']"
```

CBOR encoding:

```
78 33
   2f696574662d73797374656d3a73797374656d2f61757468656e74696361
   74696f6e2f757365725b6e616d653d27626f62275d
```

Third example based on YANG hash

Same example assuming data node "/ietf-system:system/authentication/user" is associated to YANG hash 0x2677c6c1 or "md8bB" in base64.

CBOR diagnostic notation:

```
"/md8bB?keys=\"bob\""
```

CBOR encoding:

```
71 2f6d643862423f6b6579733d22626f6222
```

6. Security Considerations

The security considerations of [RFC7049] and [I-D.ietf-netmod-rfc6020bis] apply.

This document defines an alternative encoding for data modeled in the YANG data modeling language. As such, this encoding does not contribute any new security issues in addition of those identified for the specific protocol or context for which it is used.

To minimize security risks, software on the receiving side SHOULD reject all messages that do not comply to the rules of this document and reply with an appropriate error message to the sender.

7. Acknowledgments

This document has been largely inspired by the extensive works done by Andy Bierman and Peter van der Stok on [I-D.vanderstok-core-comi]. [I-D.ietf-netmod-yang-json] has also been a critical input to this work. The authors would like to thank the authors and contributors to these two drafts.

The authors would also like to acknowledge the review, feedback, and comments from Ladislav Lhotka and Juergen Schoenwaelder.

8. References

8.1. Normative References

[I-D.ietf-netmod-rfc6020bis]
Bjorklund, M., "The YANG 1.1 Data Modeling Language", draft-ietf-netmod-rfc6020bis-11 (work in progress), February 2016.

[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>.

[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>.

8.2. Informative References

[I-D.ietf-netmod-yang-json]
Lhotka, L., "JSON Encoding of Data Modeled with YANG", draft-ietf-netmod-yang-json-09 (work in progress), March 2016.

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

[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>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <http://www.rfc-editor.org/info/rfc7223>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <http://www.rfc-editor.org/info/rfc7228>.

[RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <http://www.rfc-editor.org/info/rfc7277>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <http://www.rfc-editor.org/info/rfc7317>.

Authors' Addresses

Michel Veillette (editor)
Trilliant Networks Inc.
610 Rue du Luxembourg
Granby, Quebec J2J 2V2
Canada

Phone: +14503750556
Email: michel.veillette@trilliantinc.com

Alexander Pelov (editor)
Acklio
2bis rue de la Chataigneraie
Cesson-Sevigne, Bretagne 35510
France

Email: a@ackl.io

Abhinav Somaraju
Tridonic GmbH & Co KG
Farbergasse 15
Dornbirn, Vorarlberg 6850
Austria

Phone: +43664808926169
Email: abhinav.somaraju@tridonic.com

Randy Turner
Landis+Gyr
30000 Mill Creek Ave
Suite 100
Alpharetta, GA 30022
US

Phone: ++16782581292
Email: randy.turner@landisgyr.com
URI: http://www.landisgyr.com/

Ana Minaburo
Acklio
2bis rue de la chataigneraie
Cesson-Sevigne, Bretagne 35510
France

Email: ana@ackl.io