

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2017

M. Veillette, Ed.
Trilliant Networks Inc.
A. Pelov, Ed.
Acklio
A. Somaraju
Tridonic GmbH & Co KG
R. Turner
Landis+Gyr
A. Minaburo
Acklio
October 31, 2016

**CBOR Encoding of Data Modeled with YANG
draft-ietf-core-yang-cbor-03**

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 May 4, 2017.

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](#) [3](#)
- [2. Terminology and Notation](#) [3](#)
 - [2.1. CBOR diagnostic notation](#) [4](#)
- [3. Properties of the CBOR Encoding](#) [5](#)
- [4. Encoding of YANG Data Node Instances](#) [6](#)
 - [4.1. The 'leaf' Data Node](#) [6](#)
 - [4.2. The 'container' Data Node](#) [6](#)
 - [4.2.1. SIDs as keys](#) [7](#)
 - [4.2.2. Member names as keys](#) [8](#)
 - [4.3. The 'leaf-list' Data Node](#) [9](#)
 - [4.4. The 'list' Data Node](#) [9](#)
 - [4.4.1. SIDs as keys](#) [10](#)
 - [4.4.2. Member names as keys](#) [13](#)
 - [4.5. The 'anydata' Data Node](#) [14](#)
 - [4.6. The 'anyxml' Data Node](#) [15](#)
- [5. Representing YANG Data Types in CBOR](#) [15](#)
 - [5.1. The unsigned integer Types](#) [15](#)
 - [5.2. The integer Types](#) [15](#)
 - [5.3. The 'decimal64' Type](#) [16](#)
 - [5.4. The 'string' Type](#) [16](#)
 - [5.5. The 'boolean' Type](#) [17](#)
 - [5.6. The 'enumeration' Type](#) [17](#)
 - [5.7. The 'bits' Type](#) [18](#)
 - [5.8. The 'binary' Type](#) [18](#)
 - [5.9. The 'leafref' Type](#) [19](#)
 - [5.10. The 'identityref' Type](#) [19](#)
 - [5.10.1. SIDs as identityref](#) [20](#)
 - [5.10.2. Name as identityref](#) [20](#)
 - [5.11. The 'empty' Type](#) [21](#)
 - [5.12. The 'union' Type](#) [21](#)
 - [5.13. The 'instance-identifier' Type](#) [22](#)
 - [5.13.1. SIDs as instance-identifier](#) [22](#)
 - [5.13.2. Names as instance-identifier](#) [25](#)
- [6. Security Considerations](#) [26](#)
- [7. IANA Considerations](#) [26](#)
 - [7.1. Tags Registry](#) [26](#)
- [8. Acknowledgments](#) [26](#)
- [9. References](#) [27](#)
 - [9.1. Normative References](#) [27](#)

[9.2. Informative References](#) [27](#)
[Authors' Addresses](#) [28](#)

[1. Introduction](#)

The specification of the YANG 1.1 data modelling language [[RFC7950](#)] defines 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 [[RFC7951](#)].

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 [[RFC7950](#)]:

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

- o schema tree
- o submodule

The following terms are defined in [[RFC7951](#)]:

- o member name
- o name of an identity
- o namespace-qualified

This specification also makes use of the following terminology:

- o 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.
- o delta : Difference between the SID assigned to the current schema node and the SID assigned to the parent.
- o parent: The collection in which a schema node is defined.
- o 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. Table 1 below provides a summary of this notation.

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

Table 1: CBOR diagnostic notation summary

The following extensions to the CBOR diagnostic notation are supported:

- o Comments can be added to the end of each line. Any characters after a Pound sign ('#') outside of a string, up to the end of the line, are treated as a comment.
- o Deltas are represented as numbers preceded by a '+' or '-' sign. The use of the '+' sign for positive deltas represents an extension to the CBOR diagnostic notation as defined by [\[RFC7049\] section 6](#).

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 needs to be provided by some external means when required.

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 two type of keys; SID as defined in [[I-D.ietf-core-sid](#)] and member names as defined in [[RFC7951](#)]. 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 (e.g. RESFCONF, CoMI) can mandate the use of a specific key type.

In order to minimize the size of the encoded data, the proposed mapping avoid any unnecessary meta-information beyond those natively supported by CBOR. For instance, CBOR tags are used solely in the case of the union datatype to distinguish explicitly the use of different YANG datatypes encoded using the same CBOR major type.

It is expected that application 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.

4. Encoding of YANG Data 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 [[RFC7950](#)] and CBOR [[RFC7049](#)].

4.1. The 'leaf' Data Node

Leafs MUST be encoded based on the encoding rules specified in [Section 5](#).

4.2. The 'container' Data 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. Each key within the CBOR map is set to a data node identifier, each value is set to the value of this data node instance.

This specification supports two type of keys; SID as defined in [[I-D.ietf-core-sid](#)] encoded using CBOR unsigned or signed integers and member names as defined in [[RFC7951](#)] encoded using CBOR text strings. The use of CBOR byte strings for keys is reserved for future extensions.

[4.2.1.](#) SIDs as keys

Keys implemented using SIDs MUST be encoded using a CBOR unsigned integer (major type 0) or CBOR signed integer (major type 1), depending on the actual value. Keys are set to the delta 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.

The following example shows the encoding of the 'system' container using the SIDs defined in [[I-D.ietf-core-sid](#)] [Appendix C](#).

Definition example from [[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;
      }
    }
  }
}
```


CBOR diagnostic notation:

```
{
  1717 : {
    +2 : "2015-10-02T14:47:24Z-05:00", # current-datetime (SID 1719)
    +1 : "2015-09-15T09:12:58Z-05:00" # boot-datetime (SID 1718)
  }
}
```

CBOR encoding:

```
a1 # map(1)
  19 06b5 # unsigned(1717)
  a2 # map(2)
    02 # unsigned(2)
    78 1a # text(26)
    323031352d31302d30325431343a34373a32345a2d30353a3030
    01 # unsigned(1)
    78 1a # text(26)
    323031352d30392d31355430393a31323a35385a2d30353a3030
```

[4.2.2.](#) 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. Names and namespaces are defined in [\[RFC7951\] section 4](#).

The following example shows the encoding of the 'system' container using names. This example is described in [Section 4.2.1](#).

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

```

4.3. The 'leaf-list' Data 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.

The following example shows the encoding the 'search' leaf-list containing the two entries, "ietf.org" and "ieee.org".

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' Data Node

A list MUST be encoded using a CBOR array data item (major type 4). Each list instance within this CBOR array is encoded using a CBOR map data item (major type 5) based on the same rules as a YANG container as defined in [Section 4.2](#).

4.4.1. SIDs as keys

The following example show the encoding a the 'server' list using the SIDs defined in [[I-D.ietf-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.

The following example shows the encoding of the 'server' list containing two enties. SIDs used in this example are defined in [[I-D.ietf-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.

Definition example from [[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;
  }
}
```

CBOR diagnostic notation:


```
[
  {
    1755 : "NRC TIC server",      # name (SID 1755)
    1757 : {                      # udp (SID 1757)
      +1 : "tic.nrc.ca",        # address (SID 1758)
      +2 : 123                  # port (SID 1759)
    },
    1753 : 0,                    # association-type (SID 1753)
    1754 : false,                # iburst (SID 1754)
    1756 : true                  # prefer (SID 1756)
  },
  {
    1755 : "NRC TAC server",     # name (SID 1755)
    1757 : {                      # udp (SID 1757)
      +1 : "tac.nrc.ca"        # address (SID 1758)
    }
  }
]
```

CBOR encoding:

```
82                                # array(2)
  a5                                # map(5)
    19 06db                          # unsigned(1755)
    6e                                # text(14)
      4e52432054494320736572766572 # "NRC TIC server"
    19 06dd                          # unsigned(1757)
    a2                                # map(2)
      01                              # unsigned(1)
      6a                              # text(10)
        74696332e6e72632e6361        # "tic.nrc.ca"
      02                              # unsigned(2)
      18 7b                          # unsigned(123)
    19 06d9                          # unsigned(1753)
    00                                # unsigned(0)
    19 06da                          # unsigned(1754)
    f4                                # primitive(20)
    19 06dc                          # unsigned(1756)
    f5                                # primitive(21)
  a2                                # map(2)
    19 06db                          # unsigned(1755)
    6e                                # text(14)
      4e52432054414320736572766572 # "NRC TAC server"
    19 06dd                          # unsigned(1757)
    a1                                # map(1)
      01                              # unsigned(1)
      6a                              # text(10)
        74616332e6e72632e6361        # "tac.nrc.ca"
```


4.4.2. Member names as keys

The following example shows the encoding of the 'server' list using names. This example is described in [Section 4.4.1](#).

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)
      4e52432054494320736572766572     # "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)
    696574662d73797374656d3a6173736f6369617469666e2d74797065
  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)
      4e52432054414320736572766572     # "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"

```

4.5. The 'anydata' Data 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:

- o Keys of any inner data nodes MUST be set to valid deltas or member names.

- o 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](#)).
- o Values MUST follow the encoding rules of one of the datatypes listed in [Section 5](#).

[4.6](#). The 'anyxml' Data Node

An anyxml schema node is used to serialize an arbitrary CBOR content, i.e., its value can be any CBOR binary object.

[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).

The following example shows the encoding of leaf 'mtu' set to 1280 bytes.

Definition example from [[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.

The following example shows the encoding of leaf 'timezone-utc-offset' set to -300 minutes.

Definition example from [[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 a decimal fraction as defined in [\[RFC7049\] section 2.4.3](#).

The following example shows the encoding of leaf 'my-decimal' set to 2.57.

Definition example from [\[RFC7317\]](#):

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

CBOR diagnostic notation: 4([-2, 257])

CBOR encoding: c4 82 21 19 0101

5.4. The 'string' Type

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

The following example shows the encoding of leaf 'name' set to "eth0".

Definition example from [\[RFC7223\]](#):

```
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).

The following example shows the encoding of leaf 'enabled' set to 'true'.

Definition example from [[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 (major type 0) or CBOR signed integer (major type 1), depending on the actual value. Enumeration values are either explicitly assigned using the YANG statement 'value' or automatically assigned based on the algorithm defined in [[RFC7950](#)] [section 9.6.4.2](#).

The following example shows the encoding of leaf 'oper-status' set to 'testing'.

Definition example from [[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

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 are either explicitly assigned using the YANG statement 'position' or automatically assigned based on the algorithm defined in [\[RFC7950\] section 9.7.4.2](#).

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.

The following example shows the encoding of leaf 'mybits' with the 'disable-nagle' and '10-Mb-only' flags set.

Definition example from [\[RFC7950\]](#):

```
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'

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).

The following example shows the encoding of leaf 'aes128-key' set to 0x1f1ce6a3f42660d888d92a4d8030476e.

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.

The following example shows the encoding of leaf 'interface-state-ref' set to the value "eth1".

Definition example from [[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.ietf-core-sid](#)] or a name as defined in [[RFC7951](#)] [section 6.8](#).

5.10.1. 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.

The following example shows the encoding of leaf 'type' set to the value 'iana-if-type:ethernetCsmacd' (SID 1180).

Definition example from [[RFC7317](#)]:

```
identity interface-type {
}

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

identity ethernetCsmacd {
  base iana-interface-type;
}

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

CBOR diagnostic notation: 1180

CBOR encoding: 19 049c

5.10.2. Name as identityref

Alternatively, an identityref may be encoded using a name as defined in [[RFC7951](#)] [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. Names and namespaces are defined in [[RFC7951](#)] [section 4](#).

The following example shows the encoding of the identity 'iana-if-type:ethernetCsmacd' using its name. This example is described in [Section 5.10.1](#).

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

CBOR encoding: 78 1b

69616e612d69662d747970653a657468657226e657443736d616364

5.11. The 'empty' Type

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

The following example shows the encoding of leaf 'is-router' when present.

Definition example from [[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. When used in a union, the following YANG datatypes are prefixed by CBOR tag to avoid confusion between different YANG datatypes encoded using the same CBOR major type.

- o bits
- o enumeration
- o identityref
- o instance-identifier

See [Section 7.1](#) for more information about these CBOR tags.

The following example shows the encoding of leaf 'ip-address' when set to "2001:db8:a0b:12f0::1".

Definition example from [[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 two approaches for encoding an instance-identifier, one based on SIDs as defined in [\[I-D.ietf-core-sid\]](#) and one based on names as defined in [\[RFC7951\] section 6.11](#).

5.13.1. SIDs as instance-identifier

SIDs uniquely identify a data node. In the case of a single instance data node, a data node defined at the root of a YANG module or submodule or data nodes defined within a container, the SID is sufficient to identify this instance.

In the case of a data node member of a YANG list, a SID is combined with the list key(s) to identify each instance 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.

Data nodes member of a YANG list 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).

First example:

The following example shows the encoding of a leaf of type instance-identifier which identify the data node "/system/contact" (SID 1737).

Definition example from [[RFC7317](#)]:

```
container system {  
    leaf contact {  
        type string;  
    }  
    leaf hostname {  
        type inet:domain-name;  
    }  
}
```

CBOR diagnostic notation: 1737

CBOR encoding: 19 06c9

Second example:

The following example shows the encoding of a leaf of type instance-identifier which identify the data node instance "/system/authentication/user/authorized-key/key-data" (SID 1730) for user name "bob" and authorized-key "admin".

Definition example from [[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;
  }
}
```

CBOR diagnostic notation: [1730, "bob", "admin"]

CBOR encoding:

```
83                   # array(3)
 19 06c2             # unsigned(1730)
 63                   # text(3)
    626f62           # "bob"
 65                   # text(5)
    61646d696e       # "admin"
```

Third example:

The following example shows the encoding of a leaf of type instance-identifier which identify the list instance "/system/authentication/user" (SID 1726) corresponding to the user name "jack".

CBOR diagnostic notation: [1726, "jack"]

CBOR encoding:


```
82                    # array(2)
  19 06be            # unsigned(1726)
  64                    # text(4)
    6a61636b        # "jack"
```

5.13.2. Names as instance-identifier

The use of names as instance-identifier is defined in [\[RFC7951\]](#) [section 6.11](#). The resulting xpath MUST be encoded using a CBOR text string data item (major type 3).

First example:

This example is described in [Section 5.13.1](#).

CBOR diagnostic notation: `"/ietf-system:system/contact"`

CBOR encoding:

```
78 1c 2f20696574662d73797374656d3a73797374656d2f636f6e74616374
```

Second example:

This example is described in [Section 5.13.1](#).

CBOR diagnostic notation:

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

CBOR encoding:

```
78 59
  2f696574662d73797374656d3a73797374656d2f61757468656e74696361
  7469666e2f757365725b6e616d653d27626f62275d2f617574686f72697a
  65642d6b65795b6e616d653d2761646d696e275d2f6b65792d64617461
```

Third example:

This example is described in [Section 5.13.1](#).

CBOR diagnostic notation:

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

CBOR encoding:

78 33

2f696574662d73797374656d3a73797374656d2f61757468656e74696361
746966e2f757365725b6e616d653d27626f62275d

6. Security Considerations

The security considerations of [\[RFC7049\]](#) and [\[RFC7950\]](#) 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. IANA Considerations

7.1. Tags Registry

This specification requires the assignment of CBOR tags for the following YANG datatypes. These tags are added to the Tags Registry as defined in [section 7.2 of \[RFC7049\]](#).

Tag	Data Item	Semantics	Reference
40	bits	YANG bits datatype	RFC XXXX
41	enumeration	YANG enumeration datatype	RFC XXXX
42	identityref	YANG identityref datatype	RFC XXXX
43	instance-identifier	YANG instance-identifier datatype	RFC XXXX

// RFC Ed.: update Tag values using allocated tags if needed and remove this note // RFC Ed.: replace XXXX with RFC number and remove this note

8. Acknowledgments

This document has been largely inspired by the extensive works done by Andy Bierman and Peter van der Stok on [\[I-D.vanderstok-core-comi\]](#). [\[RFC7951\]](#) 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.

9. References

9.1. 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>>.
- [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>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", [RFC 7950](#), DOI 10.17487/RFC7950, August 2016, <<http://www.rfc-editor.org/info/rfc7950>>.

9.2. Informative References

- [I-D.ietf-core-sid] Somaraju, A., Veillette, M., Pelov, A., Turner, R., and A. Minaburo, "YANG Schema Item Identifier (SID)", [draft-ietf-core-sid-00](#) (work in progress), October 2016.
- [I-D.vanderstok-core-comi] Stok, P., Bierman, A., Veillette, M., and A. Pelov, "CoAP Management Interface", [draft-vanderstok-core-comi-10](#) (work in progress), October 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>>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", [RFC 7951](#), DOI 10.17487/RFC7951, August 2016, <<http://www.rfc-editor.org/info/rfc7951>>.

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

