                        Constrained Objects Language
                        draft-veillette-core-cool-01

Abstract

   This document describes a management function set adapted to
   constrained devices and constrained networks (e.g., low-power,
   lossy). CoOL objects (datastores, RPCs, actions and notifications)
   are defined using the YANG modelling language
   [I-D.ietf-netmod-rfc6020bis]. Interactions with these objects are
   performed using the CoAP web transfer protocol [RFC7252]. Payloads
   are encoded using the CBOR data format [RFC7049]. The mapping
   between YANG data models and the CBOR data format is defined in [I-
   D.veillette-core-yang-cbor-mapping].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

   This document defines a CoAP function set for accessing YANG defined
   resources. YANG data models are encoded in CBOR based on the mapping
   rules defined in [I-D.veillette-core-yang-cbor-mapping]. YANG items
   are identified using a compact identifier called Structured
   Identifiers (SIDs) as defined in [I-D.somaraju-core-sid].

   The resulting protocol based on CoAP, CBOR encoded data and
   structured identifiers (SID) has a low implementation footprint and
   low network bandwidth requirements and is suitable for both
   constrained devices and constrained networks as defined by [RFC7228].
   This protocol is applicable to the different management topology
   options described by [I-D.ersue-constrained-mgmt]; centralized,
   distributed and hierarchical.

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]:

   o action

   o data node

   o data tree

   o module

   o notification

   o RPC

   o schema node

o schema tree

o submodule

This specification also makes use of the following terminology:

o candidate configuration datastore: A configuration datastore that
   can be manipulated without impacting the device's current
   configuration and that can be committed to the running
   configuration datastore. Not all devices support a candidate
   configuration datastore.

o CoOL client: The originating endpoint of a request, and the
   destination endpoint of a response.

o CoOL server: The destination endpoint of a request, and the
   originating endpoint of a response.

o delta: Within a list, a delta represents the different between the
   current SID and the SID of the previous entry within this list.
   Within a collection, a delta represents the difference between the
   SID assigned to the current schema node and the SID assigned to
   the parent. When no previous entry or parent exist, the delta is
   set to the absolute SID value.

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

o datastore: Resource used to store and access information.

o endpoint: An entity participating in the CoOL protocol. Multiple
   CoOL endpoints may be accessible using a single CoAP endpoint. In
   this case, each CoOL endpoint is accessed using a distinct URI.

o event stream: Resource used to access notifications generated by a
   CoOL server. Events are defined using the YANG notification
   statement.

o function set: A group of well-known resources that provides a
   particular service.

o object: Within CoOL, an object is a data node, an RPC or an action
   within a datastore resource or a notification within an event
   stream resource.

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

   o resource: Content identified by a URI.

   o running configuration datastore: A configuration datastore holding
      the complete configuration currently active on the device. The
      running configuration datastore always exists.

   o Structured IDentifier (SID). Unsigned integer used to identify
      different YANG items.

3. Architecture

   The CoOL protocol is based on the client-server model. The CoOL
   server is the provider of the datastore resource(s) and the event
   stream resource(s). The CoOL client is the requester of these
   resources.

   CoOL objects are defined using the YANG modeling language [RFC6020].
   Interactions with these objects are performed using the Constrained
   Application Protocol (CoAP) [RFC7252]. Payloads are encoded using
   the Concise Binary Object Representation (CBOR) [RFC7049].

   This specification is applicable to any transport and security
   protocols supported by CoAP. Implementers are free to select the
   most appropriate transport for the targeted applications.

```
    +--------------+           +-------------------------------+
    | CoOL client  |           | CoOL Server                   |
    |              |           | - Datastore resource(s)       |
    |              |           | - Event stream resource(s)    |
    +--------------+           +-------------------------------+
    | CoAP client  | <-------> | CoAP Server                   |
    +--------------+           +-------------------------------+
    |              |           |                               |
    |              |           |                               |
    | Lower layers |           | Lower layers                  |
    |              |           |                               |
    +--------------+           +-------------------------------+
```
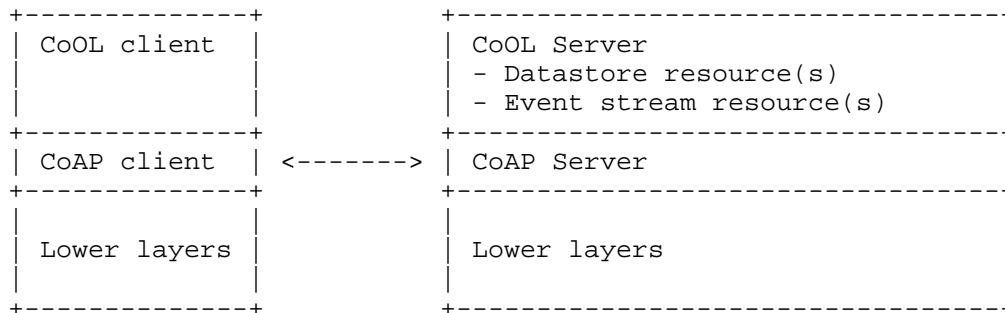
4. Resources

   This section lists the URIs recommended for the different CoOL
   resources. A CoOL server MAY implement a different set of URIs. See
   the Resource discovery section (Section 7.15) for more details on how
   a CoOL client can discover the list of URIs supported by a CoOL
   server using the "/.well-known/core" resource.

   o /c - The default datastore resource

   o /c/c - The candidate configuration datastore resource

o /c/r - The running configuration datastore resource

o /c/b - The backup configuration datastore (use to implement
   rollbacks)

o /c/e - URI used to access the default event stream for this
   device.

o /c/e0, /c/e1, ... - URI used to access alternate event streams.

o /c/0, /c/1, ... - URI used to access a specific endpoint. Each
   end point represents a virtual device which can support any of the
   resources listed above.

For example:

o /c/1 is the default datastore resource for endpoint 1

o /c/1/c is the candidate datastore resource for endpoint 1

o /c/1/r is the running configuration datastore resource for
   endpoint 1

o /c/1/b is the backup configuration datastore resource for endpoint
   1

o /c/1/e is the default event stream resource for endpoint 1

o /c/1/e0 is an alternate event stream resource for endpoint 1

All these resources are optional at the exception of the default
datastore resource. The CoAP response code 4.04 (Not Found) MUST be
returned when a CoOL client tries to access a resource that is
unavailable.

RPCs commit and cancel-commit defined in ietf-cool YANG module are
available to perform the following operations on datastores:

o Immediate or differed commit of a candidate or backup datastore.

o Confirmed commit

o Cancel of a different or confirmed commit.

5. Operations

   This section defines the different interactions supported between a
   CoOL client and a CoOL server.

5.1. GET - Retrieving all data nodes of a datastore

   The GET method is used by CoOL clients to retrieve the entire
   contents of a datastore. Implementation of this function is optional
   and dependent of the capability of the CoOL server to transfer a
   relatively large response.

   To retrieve all instantiated data nodes of a datastore resource, a
   CoOL client sends a CoAP GET request to the URI of the targeted
   datastore. If the request is accepted by the CoOL server, a 2.05
   (Content) response code is returned. The payload of the GET response
   MUST carry a CBOR array containing the contents of the targeted
   datastore. The CBOR array MUST contain a list of pairs of delta and
   associated value. A delta represents the difference between the
   current SID and the SID of the previous pair within the CBOR array.
   Each value is encoded using the rules defined by [I-D.veillette-core-
   yang-cbor-mapping].

   If the request is rejected by the CoOL server, a 5.01 Not implemented
   or 4.13 Request Entity Too Large response code is returned.

   Example:

   In this example, the CoOL server returns a datastore containing the
   following data nodes defined in the YANG module "ietf-system"
   [RFC7317] and YANG module "ietf-interfaces" [RFC7223]:

   o "/interfaces/interface" (SID 1529)

   o "/interfaces/interface/description" (SID 1530)

   o "/interfaces/interface/enabled" (SID 1531)

   o "/interfaces/interface/name" (SID 1533)

   o "/interfaces/interface/type" (SID 1534)

   o "/system-state/clock" (SID 1708)

   o "/system-state/clock/boot-datetime" (SID 1709)

   o "/system-state/clock/current-datetime" (SID 1710)

   o "/system/clock/timezone/timezone-utc-offset/timezone-utc-offset"
      (SID 1721)

   CoAP Request:

   GET /c

   CoAP response:

```
 2.05 Content Content-Format(application/cool+cbor)
 [
   1529,
     {
      4 : "eth0",              # name (SID 1533)
      1 : "Ethernet adaptor",  # description (SID 1530)
      5 : 1179,                # type (SID 1534), identity ethernetCsmacd
      2 : true                 # enabled (SID 1531)
     },
   179,                        # clock (SID 1708)
     {
       1 : "2015-02-08T14:10:08Z09:00",  # boot-datetime (SID 1709)
       2 : "2015-04-04T09:32:51Z09:00"   # current-datetime (SID 1710)
     }
   13, 60                      # timezone-utc-offset (SID 1721)
 ]
```

5.2. FETCH - Retrieving specific data nodes

   The FETCH method is used by the CoOL client of retrieve a subset of
   the data nodes within a datastore.

   To retrieve a list of data node instances, the CoOL client sends a
   CoAP FETCH request to the URI of the targeted datastore. The payload
   of the FETCH request contains the list of data node(s) instance to be
   retrieved. This list is encoded using a CBOR array, each entry
   containing an "instance-identifier" as defined by [I-D.veillette-
   core-yang-cbor-mapping]. Within each "instance-identifier", data
   nodes are identified using SIDs as defined by [I-D.somaraju-core-
   sid].

   SIDs within the list of "instance-identifier" are encoded using
   delta. A delta represents the different between the current SID and
   the SID of the previous entry within this list. The delta of the
   first entry within the list is set to the absolute SID value (current
   SID minus zero).

   On successful processing of the CoAP request, the CoOL server MUST
   return a CoAP response with a response code 2.05 (Content).

When a single data node is requested, the payload of the GET response
MUST carry the data node instance requested encoded using the rules
defined in [I-D.veillette-core-yang-cbor-mapping].

When a multiple data nodes are requested, the payload of the GET
response MUST carry a CBOR array containing the data node instance(s)
requested. Each entry within this array MUST be encoding using the
rules defined in [I-D.veillette-core-yang-cbor-mapping].

When a collection is returned (YANG container, YANG list or YANG list
instance), delta(s) are computed using the requested SID as parent.

The CBOR value undefined (0xf7) must be returned for each data node
requested but not currently available.

5.2.1. Example #1 - Simple data node

   In this example, a CoOL client retrieves the leaf "/system-
   state/clock/current-datetime" (SID 1704) and the container "/system/
   clock" (SID 1719) containing the leaf "/system/clock/timezone/
   timezone-utc-offset/timezone-utc-offset" (SID 1721). These data
   nodes are defined in the YANG module "ietf-system" [RFC7317].

   CoAP request:

   FETCH /c Content-Format(application/cool+cbor)
   [1704, 15]

   CoAP response:

   2.05 Content Content-Format(application/cool+cbor)
   [
     "2015-10-08T14:10:08Z09:00",   # current-datetime (SID 1704)
     {   # clock (SID 1719)
       2 : 540   # timezone-utc-offset (SID 1721)
     }
   ]

   CoAP requests and responses MUST be encoded in accordance with
   [RFC7252] or [I-D.ietf-core-coap-tcp-tls]. An encoding example is
   shown below:

   CoAP request:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver| T |  TKL  |   Code (0x01)  |           Message ID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Token (0 to 8 bytes) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Opt Delta (12)| Opt Length (1)|      na       | Opt Delta (3) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Opt Length (2)|      '/'       |      'c'      |1 1 1 1 1 1 1 1|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     0x82      |     0x19       |    0x06       |     0xa8       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     0x0f      |
+-+-+-+-+-+-+-+-+
```

CoAP response:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver| T |  TKL  |   Code (0x45)  |           Message ID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Token (0 to 8 bytes) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Opt Delta (12)| Opt Length (1)|      na       |1 1 1 1 1 1 1 1|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     0xa2      |     0x78       |    0x19       |     0x32       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     0x30      |     0x31       |    0x35       |     0x2d       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     0x31      |     0x30       |    0x2d       |     0x30       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     0x38      |     0x54       |    0x31       |     0x34       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     0x3a      |     0x31       |    0x30       |     0x2d       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     0x30      |     0x38       |    0x5a       |     0x30       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     0x39      |     0x3a       |    0x30       |     0x30       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     0xa1      |     0x02       |    0x19       |     0x02       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     0x1c      |
+-+-+-+-+-+-+-+-+
```

5.2.2. Example #2 - Data node instance within a YANG list

   The data type "instance-identifier" allows the selection of an
   instance of a specific data node within a list. In this example, a
   CoOL client retrieves the "/interfaces/interface/type" (SID 1529)
   leaf from the "/interfaces/interface" list. The
   "/interfaces/interface/name" associated to this interface is equal to
   "eth0". This example is based on the YANG module "ietf-interfaces"
   [RFC7223].

   CoAP request:

   FETCH /c Content-Format(application/cool+cbor)
   [[1529, "eth0"]]

   CoAP response:

   2.05 Content Content-Format(application/cool+cbor)
   "Ethernet adaptor"

5.2.3. Example #3 - YANG list

   To retrieve all instances of a list, the CoOL client excludes from
   the "instance-identifier" the key(s) of the targeted list. The list
   returned is encoded using the rules defined in [I-D.veillette-core-
   yang-cbor-mapping] section 4.4.

   In this example, a CoOL client retrieves the list "/interfaces/
   interface" (SID 1529). The response returns contain two instances,
   one for an Ethernet adaptor and one for a WIFI interface.

   CoAP request:

   FETCH /c Content-Format(application/cool+cbor)
   [1529]

   CoAP response:

```
  2.05 Content Content-Format(application/cool+cbor)
  [
    {
      4 : "eth0",            # name (SID 1533)
      1 : "Ethernet adaptor", # description (SID 1530)
      5 : 1179,              # type (SID 1534), identity ethernetCsmacd
      2 : true               # enabled (SID 1531)
    },
    {
      4 : "wlan0",           # name (SID 1533)
      1 : "WIFI ",           # description (SID 1530)
      5 : 1220,              # type (SID 1534), identity ieee80211
      2 : false              # enabled (SID 1531)
    }
  ]
```

5.2.4. Example #4 - YANG list instance

   To retrieve a list instance, the CoOL client MUST use an "instance-
   identifier" with a SID set to the targeted list and the key(s) set to
   the value(s) associated to the targeted instance.

   In this example, the CoOL client requests the instance of the list
   "/interfaces/interface" (SID 1529) associated to the name "eth1".
   The response returned by the CoOL server contains the targeted list
   instance formatted as YANG container.

   CoAP request:

   FETCH /c Content-Format(application/cool+cbor)
   [[1529, "eth1"]

   CoAP response:

```
   2.05 Content Content-Format(application/cool+cbor)
   {
     4 : "eth0"             # name (SID 1533)
     1 : "Ethernet adaptor"  # description (SID 1530)
     5 : 1179               # type (SID 1534), identity ethernetCsmacd
     2 : true               # enabled (SID 1531)
   }
```

5.2.5. Example #5 - YANG list instance filtering

   This "instance-identifier" extension allows the selection of a subset
   of data nodes within a list. This is accomplished by adding an extra
   element to the "instance-identifier". This element contains the
   subset of data nodes to be returned encoded as CBOR array. Each

entry within this CBOR array is set to the delta between the current
SID and the SID of targeted container as specified in the first entry
of the "instance-identifier".

CoOL servers SHOULD implement this "instance-identifier" extension.
When this extension is not supported, the CoOL server MUST ignore the
third element of the "instance-identifier" and return the list
instance as specified by the first two elements of the "instance-
identifier".

In this example, a CoOL client retrieves from within the
"/interfaces/interface" list (SID 1528) the leafs
"/interfaces/interface/type" (SID 1533) and "/interfaces/interface/
enabled" (SID 1530). The CoOL client also includes in this request
the selection of the leaf "/system/hostname" defined in "ietf-system"
[RFC7317].

For example:

CoAP request:

FETCH /c Content-Format(application/cool+cbor)
[[1528, ["eth0"], [5, 2]], 211]

CoAP response:

```
2.05 Content Content-Format(application/cool+cbor)
[
  {
    5 : 1179,              # type (SID 1533), identity ethernetCsmacd
    2 : true               # enabled (SID 1530)
  },
  "datatracker.ietf.org", # hostname (SID 1739)
]
```

5.2.6. Example #6 - All instances of a data node within a YANG list

This "instance-identifier" extension allows the efficient transfer of
all instances of a data node within a YANG list. To retrieve all
instances, the CoOL client excludes form the "instance-identifier"
the key(s) of the list containing the targeted data node.

The response MUST be encoded as a CBOR ARRAY containing the available
instances of the requested data node. This special encoding
minimizes significantly this commonly used type of request.

In this example, a CoOL client retrieves all instances of data node
"/interfaces-state/interface/name" (SID 1532).

Example:

CoAP request:

FETCH /c Content-Format(application/cool+cbor)
[1532]

CoAP response:

2.05 Content Content-Format(application/cool+cbor)
["eth0", "eth1", "wlan0"]

5.3. PUT - Updating all data nodes of a datastore

   The CoAP PUT method is used by CoOL clients to update the content of
   a datastore.

   The URI of the PUT request MUST be set to the URI of the targeted
   datastore.

   The payload of the PUT request MUST carry a CBOR array containing the
   new content of the datastore. The CBOR array MUST contain a list of
   pairs of delta and associated value. A delta represents the
   different between the current SID and the SID of the previous pair
   within the CBOR array. Each value is encoded using the rules defined
   by [I-D.veillette-core-yang-cbor-mapping].

   On successful processing of the CoAP request, the CoOL server MUST
   return a CoAP response with a response code 2.04 (Changed).

   A PUT request MUST be processed as an atomic transaction, if any of
   the data node transferred is rejected for any reason, the entire PUT
   request MUST be rejected and the CoOL server MUST return an
   appropriate error response as defined in section 6.

   Example:

   In this example, a CoOL client sets the default runtime datastore
   with these data nodes:

   o "/system/clock/timezone/timezone-utc-offset/timezone-utc-offset"
      (SID 1721)

   o "/system/ntp/enabled" (SID 1742)

   o "/system/ntp/server" (SID 1743)

   o "/system/ntp/server/name" (SID 1746)

o "/system/ntp/server/prefer" (SID 1747)

o "/system/ntp/server/transport/udp/udp" (SID 1748)

o "/system/ntp/server/transport/udp/udp/address" (SID 1749)

o "/system/ntp/server/transport/udp/udp/port" (SID 1750)

CoAP request:

```
PUT /c/r Content-Format(application/cool+cbor)
[
  1727, 540,                    # timezone-utc-offset (SID 1721)
  15, true,                     # enabled (SID 1742)
  1, [                          # server (SID 1743)
    {
      3 : "tic.nrc.ca",         # name (SID 1746)
      4 : true,                 # prefer (SID 1747)
      5 : {                     # udp (SID 1748)
        6 : "132.246.11.231",   # address (SID 1749)
        7 : 123                 # port (SID 1750)
      }
    },
    {
      3 : "tac.nrc.ca",         # name (SID 1746)
      4 : false,                # prefer (SID 1747)
      5 : {                     # udp (SID 1748)
        6 : "132.246.11.232"    # address (SID 1749)
      }
    }
  ]
]
```

CoAP response:

2.04 Changed

5.4. PATCH - Updating specific data nodes

   The PATCH method is used by CoOL clients to modify a subset of a
   datastore.

   To modify a datastore, the CoOL client sends a CoAP PATH request to
   the URI of the targeted datastore. The payload of the FETCH request
   contains the list of data node instance(s) to be updated, inserted or
   deleted. This list is encoded using a CBOR array and contains a
   sequence of pairs of "instance-identifier" and associated values.

Within each "instance-identifier", data nodes are identified using
SIDs as defined by [I-D.somaraju-core-sid]. SIDs within the list are
encoded as delta.

On reception, the list is processed by the CoOL server as follows:

o If the targeted data instance already exists, this instance is
   replaced by the associated value (not merged). To update only
   some children of a collection, each child data node MUST be
   provided individually.

o If the targeted data instance doesn't exist, this instance is
   created.

o If the targeted data instance already exists but is associated
   with the value "null", this instance is deleted.

On successful processing of the CoAP request, the CoOL server MUST
return a CoAP response with a response code 2.05 (Content).

A PATCH request MUST be processed as an atomic transaction, if any of
the data nodes transferred is rejected for any reasons, the entire
PATCH request MUST be rejected and the CoOL server MUST return an
appropriate error response as defined in section 6.

Example:

In this example, a CoOL client performs the following operations:

o Set "/system/ntp/enabled" to true.

o Remove the server "tac.nrc.ca" from the"/system/ntp/server" list.

o Add the server "NTP Pool server 2" to the list "/system/ntp/
   server".

o Set "prefer" to false for the server "tic.nrc.ca".

CoAP request:

```
PATCH /c/r Content-Format(application/cool+cbor)
[
  1742 , true,                           # enabled (1742)
  [1, "tac.nrc.ca"], null,               # server (SID 1743)
  0,                                     # server (SID 1743)
    {
      3 : "NTP Pool server 2",           # name (SID 1746)
      4 : true,                          # prefer (SID 1747)
      5 : {                              # udp (SID 1748)
        6 : "2620:10a:800f::11",         # address (SID 1749)
      }
    }
  [4, "tic.nrc.ca"], false               # prefer (SID 1747)
]
```

CoAP response:

2.04 Changed

5.5. POST - Protocol operation

   Protocol operations are defined using the YANG "rpc" or YANG "action"
   statements.

   To execute a protocol operation, the CoOL client sents a CoAP POST
   request to the URI of the targeted datastore.

   The payload of the POST request carries a CBOR array with up to two
   entries. The first entry carries the instance-identifier identifying
   the targeted protocol operation. The second entry carries the
   protocol operation input(s). Input(s) are present only if defined
   for the invoked protocol operation and used by the CoOL client.
   Input(s) are encoded using the rules defined for a YANG container,
   deltas are relative to the SID assigned to the protocol operation.

   On successful completion on the protocol operation, the CoOL server
   returns a CoAP response with the response code set to 2.05 (Content).
   When output parameters are returned by the CoOL server, these
   parameter(s) are carried in the CoAP response payload. Output(s) are
   encoded using the rules defined for a YANG container, deltas are
   relative to the SID assigned to the protocol operation.

5.5.1. Example #1 - RPC

   This example is based on the "activate-software-image" RPC defined in
   [I-D.ietf-netmod-rfc6020bis], assuming that this RPC is assigned to
   SID 1932, leaf image-name to SID 1933 and leaf status to SID 1934.
   These SIDs are defined strictly for the purpose of this example.

```
rpc activate-software-image { input { leaf image-name { type string;
} } output { leaf status { type string; } } }
```

CoAP request:

```
POST /c Content-Format(application/cool+cbor)
[
  1932,
  {
    1 : "acmefw-2.3"                    # image-name (SID 1933)
  }
]
```

CoAP response:

```
2.05 Content
{
  2 : "installed"                      # status (SID 1934)
}
```

5.5.2. Example #2 - Action

   This example is based on the "reset" action defined in
   [I-D.ietf-netmod-rfc6020bis] assuming that this action is assigned to
   SID 1902, leaf reset-at to SID 1903 and leaf reset-finished-at to SID
   1904. These SIDs are defined strictly for the purpose of this
   example.

```
list server { key name; leaf name { type string; } action reset {
input { leaf reset-at { type yang:date-and-time; mandatory true; } }
output { leaf reset-finished-at { type yang:date-and-time; mandatory
true; } } } }
```

CoAP request:

```
POST /c Content-Format(application/cool+cbor)
[
  [1902, "myServer"],
  {
    1 : "2016-02-08T14:10:08Z09:00"    # reset-at (SID 1903)
  }
]
```

CoAP response:

```
   2.05 Content
   {
     2 : "2016-08T14:10:08Z09:18"          # reset-finished-at (SID 1904)
   }
```

5.6. Event stream

WARNING
This section requires more work to address the following identified issues:

* Retrieval of past events (e.g. start-time, stop-time)
* Retrieval of specific events (e.g. filter)
* Configuration persistence
* Configuration of by a third entity (configuration tool)
* Support of multicast
* Event congestion-avoidance
* Transfer reliability

The current solution based on the observe CoAP option can be augmented
or completely replaced by a future version of this draft.

   Notifications are defined using the YANG "notification" statement.
   Subscriptions to an event stream and notification reporting are
   performed using an event stream resource. When multiple event stream
   resources are supported, the list of notifications associated with
   each stream is either pre-defined or configured in the CoOL server.
   CoOL clients MAY subscribe to one or more event stream resources.

   To subscribe to an event stream resource, a CoOL client MUST send a
   CoAP GET with the Observe CoAP option set to 0. To unsubscribe, a
   CoOL client MAY send a CoAP reset or a CoAP GET with the Observe
   option set to 1. For more information on the observe mechanism, see
   [RFC7641].

   Each notification transferred by a CoOL server to each of the
   registered CoOL clients is carried in a CoAP response with a response
   code set to 2.05 (Content). Each CoAP response MUST carry in its
   payload at least one notification but MAY carry multiple. Each
   notification is carried in a notification-payload defined in ietf-
   cool, see Appendix A. The notification-payload supports different
   meta-data associated to this notification, such as the notification
   identifier, event timestamp, sequence number, severity level and
   facility. All of these meta information are optional with the
   exception of the notification identifier.

   The CoAP response payload is encoded using the rules defined for the
   PUT request. When multiple notifications are reported, the CoAP

response payload carries a CBOR array, with each entry containing a
notification.

This example is based on the "link-failure" and "interface-enabled"
notifications defined in [I-D.ietf-netmod-rfc6020bis] assuming the
following SID assignment:

o "/link-failure" (SID 1942)

o "/link-failure/if-name" (SID 1943)

o "/link-failure/admin-status" (SID 1944)

o "/interfaces/interface/interface-enabled" (SID 1538)

o "/interfaces/interface/interface-enabled/by-user" (SID 1539)

These SIDs are defined strictly for the purpose of this example.

```
notification link-failure {
  leaf if-name {
    type leafref {
      path "/interface/name";
    }
  }
  leaf admin-status {
    type leafref {
      path "/interface[name = current()/../if-name]/admin-status";
    }
  }
}

container interfaces {
  list interface {
    key "name";

    leaf name {
      type string;
    }

    notification interface-enabled {
      leaf by-user {
        type string;
      }
    }
  }
}
```

In this example, a CoOL client starts by registering to the default
event stream resource "/c/e".

CoAP request:

GET /c/e observe(0) Token(0x9372)

The CoOL server confirms this registration by returning a first CoAP
response. The payload of this CoAP response may be empty or may
carry the last notification reported by this server.

CoAP response:

2.05 Content Observe(52) Token(0xD937)

After detecting an event, the CoOL server sends its first
notification to the registered CoOL client.

CoAP response:

```
2.05 Content Observe(53) Token(0xD937)
Content-Format(application/cool+cbor)
[
  1010 , [1538, "eth0"],              # _id (SID 1010)
  1,{                                 # content (SID 1011)
    1 : "bob"                         # by-user  (SID 1539)
  }
  5 , "2016-03-08T14:10:08Z09:00",    # timestamp (SID 1015)
]
```

To optimize communications or because of other constraints, the CoOL
server might transfer multiple notifications in a single CoAP
response.

CoAP response:

```
   2.05 Content Observe(52) Token(0xD937)
   Content-Format(application/cool+cbor)
   [
     [
       1010 , [1538, "eth0"],       # _id = interface-enabled (SID 1010)
        1,{                         # content (SID 1011)
          1 : "jack"               # by-user (SID 1539)
        }
        5 , "2016-03-12T15:49:51Z09:00",  # timestamp (SID 1015)
     ],
     [
       1010 , 1942,                 # _id = link-failure (SID 1010)
        1,{                         # content (SID 1011)
          1 : "eth0",              # if-name (SID 1943)
          1 : 1                    # admin-status = up (SID 1944)
        }
        5 , "2016-03-12T15:50:06Z09:00",  # timestamp (SID 1015)
     ]
   ]
```

6. CoAP compatibility

6.1. Working with Uri-Host, Uri-Port, Uri-Path, and Uri-Query

   Uri-Query is not currently used by this protocol. Uri-Host, Uri-Port
   and Uri-Path MUST be used as specified by [RFC6690] to target the
   CoOL resources as defined by section 3.

6.2. Working with Location-Path and Location-Query

   This version of CoOL doesn't support the creation of resources
   (datastore or event stream). For this reason, the use of Location-
   Path and Location-Query is not required.

6.3. Working with Accept

   This option is not required since this protocol supports a single
   content format, "application/cool+cbor".

6.4. Working with Max-Age

   This option MUST be supported as specified by [RFC6690].

6.5. Working with Proxy-Uri and Proxy-Scheme

   This option MUST be supported as specified by [RFC6690].

6.6. Working with If-Match, If-None-Match and ETag

   This option MUST be supported as specified by [RFC6690]. Each ETag
   is associated to all schema nodes within a datastore.

6.7. Working with Size1, Size2, Block1 and Block2

   When the UDP transport is used and a large payload need to be
   transferred, support of the CoAP block transfer as defined by
   [I-D.ietf-core-block] is recommended.

   6.8. Working with Observe

   A CoOL server MAY support state change notifications to some or all
   its leaf data nodes. When supported the CoOL server MUST implement
   the Server-Side requirements defined in [RFC7641] section 3 and the
   CoOL client MUST implement the Client-Side requirements defined in
   [RFC7641] section 4.

   To start observing a leaf data node, a CoOL client MUST send a CoAP
   FETCH with the Observe CoAP option set to 0.

   The payload of the FETCH request carries a CBOR array of instance-
   identifier. The first entry MUST be set to the "instance-identifier"
   of the data node instance observed. The following entries are
   optional and allow the selection of coincidental values, data nodes
   reported at the same time as the observed data node. Coincidental
   values are included in each notification reported, but changes to
   these extra data nodes MUST not trigger notification messages.

   A subscription can be terminated by the CoOL client by returning a
   CoAP Reset message or by sending a GET request with an Observe CoAP
   option set to deregister (1). More details are available in
   [RFC7641].

   Example:

   In this example, a CoOL client subscribes to state changes of the
   data node "/system/ntp/enabled" (SID = 1742) and requests that data
   node "/system/hostname" (SID 1739) is reported as coincidental value.

   A first response is immediately returned by the CoOL server to
   confirm the subscription and to report the current values of the
   requested data nodes.

   Subsequent responses are returned by the CoOL server each time the
   state of data node "/system/ntp/enabled" changes.

CoAP request:

```
FETCH /c Content-Format(application/cool+cbor) Observe(0)
[ [1742, "tic.nrc.ca"], -3 ]
```

CoAP response:

```
2.05 Content Content-Format(application/cool+cbor) Observe(2631)
[
  false,                    # enabled (SID 1742)
  "tic"                     # hostname (SID 1739)
]
```

CoAP response:

```
2.05 Content Content-Format(application/cool+cbor) Observe(2632)
[
  true,                     # enabled (SID 1742)
  "tic"                     # hostname (SID 1739)
]
```

6.8. Working with resource discovery

   The "/.well-known/core" resource is used by CoOL clients to discover
   resources implemented by CoOL servers. Each CoOL server MUST have an
   entry in the "/.well-known/core" resource for each datastore resource
   and event stream resource supported.

   Resource discovery can be performed using a CoAP GET request. If
   successful, the CoAP response MUST have a response code set to 2.05
   (Content), a Content-Format set to "application/link-format", and a
   payload containing a list of web links.

   To enable discovery of specific resource types, the CoAP server MUST
   support the query string "rt".

   Link format and the "/.well-known/core" resource are defined in
   [RFC6690].

   Example:

   CoAP request:

   GET /.well-known/core

   CoAP response:

```
2.05 Content Content-Format(application/link-format)
</c>;rt="cool.datastore",
</c/r>;rt="cool.datastore",
</c/b>;rt="cool.datastore",
</c/e>;rt="cool.event-stream",
```

In this example, a CoOL client retrieves the list of all resources available on a CoOL server.

Alternatively, the CoOL client may query for a specific resource type. In this example, the CoOL client queries for resource type (rt) "cool.datastore".

CoAP request:

```
GET /.well-known/core?rt=cool.datastore
```

CoAP response:

```
2.05 Content Content-Format(application/link-format)
</c>;rt="cool.datastore",
```

7. Error Handling

All CoAP response codes defined by [RFC7252] MUST be accepted and processed accordingly by CoOL clients. Optionally, client errors (CoAP response codes 4.xx) or server errors (CoAP response codes 5.xx) MAY have a payload providing further information about the cause of the error. This payload contains the " error-payload" container (SID 1006) defined in the "ietf-cool" YANG module, see Appendix A.

Example:

CoAP response:

```
4.00 Bad Request (Content-Format: application/cool+cbor)
[
  1006 , {
    1 : 2,                              # error-code, SID 1007
    2 : "Unknown data node 69687"       # error-text, SID 1008
  }
]
```

8. Security Considerations

   This application protocol relies on the lower layers to provide
   confidentiality, integrity, and availability. A typical approach to
   archive these requirements is to implement CoAP using the DTLS
   binding as defined in [RFC7252] section 9. Other approaches are
   possible to fulfill these requirements, such as the use of a network
   layer security mechanism as discussed in
   [I-D.bormann-core-ipsec-for-coap] or a link layer security mechanism
   for exchanges done within a single sub-network.

   In some applications, different access rights to objects (data nodes,
   protocol operations and notifications) need to be granted to
   different CoOL clients. Different solutions are possible, such as
   the implementation of Access Control Lists (ACL) using YANG module(s)
   or the use of an authorization certificate as defined in [RFC5755].
   These access control mechanisms need to be addressed in complementary
   specifications.

   The Security Considerations section of CoAP [RFC7252] is especially
   relevant to this application protocol and should be reviewed
   carefully by implementers.

9. IANA Considerations

9.1. "FETCH" CoAP Method Code

   This draft makes use of the PATCH CoAP method as defined in
   [I-D.bormann-core-coap-fetch]. This method needs to be registered in
   the CoAP Method Codes sub-registry as defined in [RFC7252] section
   12.1.1.

9.2. "PATCH" CoAP Method Code

   This draft makes use of the PATCH CoAP method as defined in
   [I-D.vanderstok-core-patch]. This method needs to be registered in
   the CoAP Method Codes sub-registry as defined in [RFC7252] section
   12.1.1.

10. Acknowledgments

   This document have 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-netconf-restconf] have 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 thank Carsten Bormann for his help
during the development of this document and his useful comments
during the review process.

11. References

11.1. Normative References

   [I-D.bormann-core-coap-fetch]
             Bormann, C., "CoAP FETCH Method", draft-bormann-core-coap-
             fetch-00 (work in progress), October 2015.

   [I-D.ietf-core-block]
             Bormann, C. and Z. Shelby, "Block-wise transfers in CoAP",
             draft-ietf-core-block-18 (work in progress), September
             2015.

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

   [I-D.vanderstok-core-patch]
             Stok, P. and A. Sehgal, "Patch Method for Constrained
             Application Protocol (CoAP)", draft-vanderstok-core-
             patch-02 (work in progress), October 2015.

   [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>.

   [RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link
             Format", RFC 6690, DOI 10.17487/RFC6690, August 2012,
             <http://www.rfc-editor.org/info/rfc6690>.

   [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>.

   [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>.

   [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>.

   [RFC7641] Hartke, K., "Observing Resources in the Constrained
              Application Protocol (CoAP)", RFC 7641,
              DOI 10.17487/RFC7641, September 2015,
              <http://www.rfc-editor.org/info/rfc7641>.

11.2. Informative References

   [I-D.bormann-core-ipsec-for-coap]
              Bormann, C., "Using CoAP with IPsec", draft-bormann-core-
              ipsec-for-coap-00 (work in progress), December 2012.

   [I-D.ersue-constrained-mgmt]
              Ersue, M., Romascanu, D., and J. Schoenwaelder,
              "Management of Networks with Constrained Devices: Problem
              Statement, Use Cases and Requirements", draft-ersue-
              constrained-mgmt-03 (work in progress), February 2013.

   [I-D.ietf-core-coap-tcp-tls]
              Bormann, C., Lemay, S., Technologies, Z., and H.
              Tschofenig, "A TCP and TLS Transport for the Constrained
              Application Protocol (CoAP)", draft-ietf-core-coap-tcp-
              tls-01 (work in progress), November 2015.

   [I-D.ietf-netconf-restconf]
              Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF
              Protocol", draft-ietf-netconf-restconf-09 (work in
              progress), December 2015.

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

   [RFC5755] Farrell, S., Housley, R., and S. Turner, "An Internet
              Attribute Certificate Profile for Authorization",
              RFC 5755, DOI 10.17487/RFC5755, January 2010,
              <http://www.rfc-editor.org/info/rfc5755>.

   [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>.

Appendix A. File "ietf-cool.yang"

   Module containing the different definitions required by the CoOL
   protocol.

```
  module ietf-cool {
    namespace "urn:ietf:ns:cool";
    prefix cool;

    organization
      "IETF Core Working Group";

    contact
      "Ana Minaburo
       <mailto:ana@ackl.io>

       Abhinav Somaraju
       <mailto:abhinav.somaraju@tridonic.com>

       Alexander Pelov
       <mailto:a@ackl.io>

       Michel Veillette
       <mailto:michel.veillette@trilliantinc.com>

       Randy Turner
       <mailto:Randy.Turner@landisgyr.com>";

    description
      "This module contains the different definitions required
       by the CoOL protocol.";

    revision 2016-01-01 {
      description
       "Initial revision.";
     reference
       "draft-veillette-core-cool";
    }
```

```
     // List of useful derived YANG data types for constrained devices

     typedef sid {
       type uint32;
       description
         "Structure Identifier value (SID).";
     }

     typedef utc-time {
       type uint32;
       description
         "Unsigned 32-bit value representing the number of seconds
         since 0 hours, 0 minutes, 0 seconds, on the 1st of January,
         2000 UTC (Universal Coordinated Time).";
     }

     // Error payload

     container error-payload {
       description
         "Optional payload of a client error (CoAP response 4.xx)
          or server error (CoAP response 5.xx).";

       leaf error-code {
         mandatory true;
         type enumeration {
           enum ok {
             value 0;
             description
               "The requested edit have been performed successfully.";
           }

           enum error {
             value 1;
             description "Unspecified error.";
           }

           enum malformed {
             value 2;
             description "Malformed CBOR payload.";
           }

           enum invalid {
             value 3;
             description "The value specified in the request can't be
                          apply to the target data node.";
           }
```

```
        enum doesNotExist {
          value 4;
          description "The target data node instance specified in
                      the request doesn't exist.";
        }

        enum alreadyExist {
          value 5;
          description "The target data node instance specified in
                      the request already exists.";
        }

        enum readOnly {
          value 6;
          description "Attempt to update a read-only data node.";
        }
      }
    }

    leaf error-text {
      mandatory false;
      type string;
      description "Textual descriptions of the error.";
    }
  }

  // Notification payload

  identity facility-type {
    description
      "A facility code is used to specify the type of process that
      is logging the message. Notifications from different facilities
      may be handled differently. Other YANG module may add new
      facility type as needed.";
  }

  identity os {
    base facility-type;
  }

  identity protocol-stack {
    base facility-type;
  }

  identity security {
    base facility-type;
  }
```

```
     identity hardware-monitoring {
       base facility-type;
     }

     identity application {
       base facility-type;
     }

     container notification-payload {
       leaf _id {
         mandatory true;
         type instance-identifier;
         description
           "Identifier associated to the notification reported.";
       }

       leaf timestamp {
         mandatory false;
         type utc-time;
         description
           "Event timestamp. Support of this field is optional
            since its not expected that all implementations have
            implement a real time clock and if so, this clock is
            available at all time.";
       }

       leaf sequence-number {
         mandatory false;
         type uint32;
         description
           "Sequence number associated to each event created by CoOL
            server within a specific event stream.";
       }

       leaf severity-level {
         reference "RFC 5424";
         mandatory false;
         type enumeration {
           enum emergency {
             value 0;
             description
               "System is unusable.";
           }
           enum alert {
             value 1;
             description
               "Should be corrected immediately.";
           }
```

```
        enum critical {
          value 2;
          description
            "Critical conditions.";
        }
        enum error {
          value 3;
          description
            "Error conditions.";
        }
        enum warning {
          value 4;
          description
            "May indicate that an error will occur if action is
            not taken.";
        }
        enum notice {
          value 5;
          description
            "Events that are unusual, but not error conditions.";
        }
        enum informational {
          value 6;
          description
            "Normal operational messages that require no action.";
        }
        enum debug {
          value 7;
          description
            "Information useful to developers for debugging the
            application.";
        }
      }
      description
        "Severity associated with this event.";
    }

    leaf facility {
      mandatory false;
      type identityref {
        base facility-type;
      }
      description
        "Type of process that is logging the message.";
      reference "RFC 5424";
    }

    leaf content {
```

```
        mandatory false;
        type anydata;
        description
          "Notification container as defined by the notification YANG
          statement.";
      }
    }

    rpc commit {
      description
        "Used to commit the changes present in a candidate datastore on
        the runtime datastore specify by the URI used to execute this
        operation.";
      input {
        leaf datastore {
          description
            "Path of the datastore resource used as the source of the
            commit operation. When not present, the default candidate
            datastore resource is used.";
          type string;
          mandatory false;
        }

        leaf commit-date-time {
          description
            "When specified, the commit operation is postponed at the
            specified date and time. When not present, the commit is
            performed on reception of this RPC. Supports of this feature
            is optional.";
          type utc-time;
          mandatory false;
        }

        leaf confirm-timeout {
          description
            "When present, a confirming commit MUST be received within
            this period after the start of the commit process.
            A confirming commit is a commit RPC without the
            confirm-timeout field presents. Supports of this feature
            is optional.";
          type string;
          mandatory false;
        }
      }
    }

    rpc cancel-commit {
      description
```

```
          "Cancels an ongoing scheduled or confirmed commit.";
      }
    }
```

Appendix B. File "ietf-cool@2016-01-01.sid"

   Following is the ".sid" file generated for the "ietf-cool" YANG
   module. See [I-D.somaraju-core-sid] for more details on SID and
   ".sid" file.

```
   {
     "assignment-ranges": [
       {
         "entry-point": 1000,
         "size": 100
       }
     ],
     "module-name": "ietf-cool",
     "module-revision": "2016-01-01",
     "items": [
       {
         "type": "identity",
         "assigned": "2016-03-08T21:59:45Z",
         "label": "ietf-cool:application",
         "sid": 1000
       },
       {
         "type": "identity",
         "assigned": "2016-03-08T21:59:45Z",
         "label": "ietf-cool:facility-type",
         "sid": 1001
       },
       {
         "type": "identity",
         "assigned": "2016-03-08T21:59:45Z",
         "label": "ietf-cool:hardware-monitoring",
         "sid": 1002
       },
       {
         "type": "identity",
         "assigned": "2016-03-08T21:59:45Z",
         "label": "ietf-cool:os",
         "sid": 1003
       },
       {
         "type": "identity",
         "assigned": "2016-03-08T21:59:45Z",
         "label": "ietf-cool:protocol-stack",
```

```
        "sid": 1004
      },
      {
        "type": "identity",
        "assigned": "2016-03-08T21:59:45Z",
        "label": "ietf-cool:security",
        "sid": 1005
      },
      {
        "type": "node",
        "assigned": "2016-03-08T21:59:45Z",
        "label": "/error-payload",
        "sid": 1006
      },
      {
        "type": "node",
        "assigned": "2016-03-08T21:59:45Z",
        "label": "/error-payload/error-code",
        "sid": 1007
      },
      {
        "type": "node",
        "assigned": "2016-03-08T21:59:45Z",
        "label": "/error-payload/error-text",
        "sid": 1008
      },
      {
        "type": "node",
        "assigned": "2016-03-08T21:59:45Z",
        "label": "/notification-payload",
        "sid": 1009
      },
      {
        "type": "node",
        "assigned": "2016-03-08T21:59:45Z",
        "label": "/notification-payload/_id",
        "sid": 1010
      },
      {
        "type": "node",
        "assigned": "2016-03-08T21:59:45Z",
        "label": "/notification-payload/content",
        "sid": 1011
      },
      {
        "type": "node",
        "assigned": "2016-03-08T21:59:45Z",
        "label": "/notification-payload/facility",
```

```
        "sid": 1012
      },
      {
        "type": "node",
        "assigned": "2016-03-08T21:59:45Z",
        "label": "/notification-payload/sequence-number",
        "sid": 1013
      },
      {
        "type": "node",
        "assigned": "2016-03-08T21:59:45Z",
        "label": "/notification-payload/severity-level",
        "sid": 1014
      },
      {
        "type": "node",
        "assigned": "2016-03-08T21:59:45Z",
        "label": "/notification-payload/timestamp",
        "sid": 1015
      },
      {
        "type": "rpc",
        "assigned": "2016-03-08T21:59:45Z",
        "label": "/cancel-commit",
        "sid": 1016
      },
      {
        "type": "rpc",
        "assigned": "2016-03-08T21:59:45Z",
        "label": "/commit",
        "sid": 1017
      },
      {
        "type": "rpc",
        "assigned": "2016-03-08T21:59:45Z",
        "label": "/commit/input/commit-date-time",
        "sid": 1018
      },
      {
        "type": "rpc",
        "assigned": "2016-03-08T21:59:45Z",
        "label": "/commit/input/confirm-timeout",
        "sid": 1019
      },
      {
        "type": "rpc",
        "assigned": "2016-03-08T21:59:45Z",
        "label": "/commit/input/datastore",
```

```
         "sid": 1020
      }
   ]
}
```

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