coman
Internet-Draft
Intended status: Informational
Expires: December 30, 2013

P. van der Stok
consultant

June 28, 2013

CoAp Management Interfaces
draft-vanderstok-core-comi-00

Abstract

The draft describes an interface based on CoAP to manage constrained devices. Access to existing MIBs is included. The proposed integration of CoAP with SNMP reduces the code size by removing an important part of the SNMP code. The draft lists a set of CoMI objects that describe the operational settings of the applications on the device. A majority of them is taken over from other drafts to provide one uniform CoMI based access.

Note

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 30, 2013.

Copyright Notice

Table of Contents

1. Requirements 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].

2. Introduction

The Constrained RESTful Environments (CoRE) working group aims at

Machine to Machine (M2M) applications such as smart energy and building control.

Small M2M devices need to be managed in an automatic fashion to handle the large quantities of devices that are expected to be

installed in future installations. The management protocol of choice for Internet is SNMP [RFC3410] as is testified by the large number of Management Information Base (MIB) [RFC3418] specifications currently published [STD0001]. More recently, he NETCONF protocol [RFC6241] was developed with an extended set of messages using XML [XML] as data format. The data syntax is specified with YANG [RFC6020] and a mapping from Yang to XML is specified. In [RFC6643] SMIv2 syntax is expressed in Yang. Contrary to SNMP and also CoAP, NETCONF assumes persistent connections for example provided by SSH. The NETCONF protocol provides operations to retrieve, configure, copy, and delete configuration data-stores. Configuring data-stores distinguishes NETCONF from SNMP which operates on standardized MIBs.

The CoRE Management Interface (CoMI) is intended to work on standardized data-sets in a stateless client-server fashion and is thus closer to SNMP than to NETCONF. Standardized data sets are necessary when small devices from different manufacturers are managed by applications originating from another set of manufacturers. Stateless communication is encouraged to keep communications simple and the amount of state information small in line with the design objectives of 6lowpan [RFC4944] [RFC6775] , RPL [RFC6650] , and CoAP [I-D.ietf-core-coap].

Currently, managed devices need to support two protocols: CoAP and SNMP each with its own security solution. When the MIB can be accessed with the CoAP protocol, the SNMP protocol with its security provisioning can be replaced with the CoAP protocol [I-D.ietf-core-coap] and DTLS [RFC6347]. This arrangement significantly reduces memory requirements, simplifies the stack in the constrained device, and harmonizes applications development. A CoAP device that needs management can include a set of existing MIBs and use CoMI to manipulate them.

The objective of CoMI is to provide a CoAP based Function Set that reads and sets parameter values in devices, and acquires operational values that change with time. It extends SNMP by providing an XML

interface [XML] that allows more complex structures than the flat tables of SNMP.

CoMI is intended for small devices. The XML overhead can be prohibitive. It is therefore recommended to transport EXI [EXI] in the payload. In [EXI-measurement] it is shown that EXI can be an order of magnitude smaller than the equivalent XML representation. Actually, the EXI structure adds the overhead per data unit of an EXI event (indicates the type of the following XML element) with a size that depends on the number of EXI event types present in the schema and its frequency of occurrence. In [JSON-XML] it is shown that memory and CPU usage for sending JSON encoded or XML encoded objects

led on average to a 50% lower resource usage for JSON. Consequently, from a resource utilization point of view EXI seems the right choice.

The end goal of CoMI is to provide information exchange over the CoAP transport protocol in a uniform manner to approach the full management functionality as specified in [I-D.ersue-constrained-mgmt]. This memo collects existing management functions from several sources to uniformize their access.

3. Terminology

Core Management Interface (CoMI) specifies the profile of Function Sets which access CoMI objects with the pupose of managing the operation of constrained devices in a network.

4. CoAP Interface

In CoRE a group of links can constitute a Function Set. The format of the links is specified in [RFC6690]. This note specifies a Management Function Set. CoMI end-points that implement the CoMI management protocol support at least one discoverable management resource of resource type (rt) core.mg with path /mg, where mg is short-hand for management. The mg resource has two sub-resources each accessible with a different path:

- MIB with path /mg/mib and a binary content format
- CoMI with path /mg/comi and an XML, EXI content format.

The MIB resource is provided to enable a smooth transition from SNMP to CoMI. The binary content is composed of BER encoded SNMP PDUs. The CoMI resource provides access to the CoMI objects. CoMI objects include MIBs and the objects described in Section 7. XML schemas describe the structure of the CoMI objects. It is expected that given the verbosity of XML, CoMI messages will mostly use EXI. The service provided by the CoMI server is to transfer from/to the constrained devices: (1) BER encoded SNMP PDUs from/to MIBs, and (2) EXI encoded data from/to CoMI objects including MIB objects. The profile of the management function set, with IF=core.comi, is shown in the table below

name	path	RT	Data Type
Management	/mg	core.mg	n/a
MIB binary	/mg/mib	core.mg.mib	application/octet

CoMI objects	/mg/comi	core.mg.comi	application/exi
CoMI MIB	/mg/comi/mib	core.mg.comi.mib	application/exi

It is intended that CoMI schemas will be developed independently for different management subjects or contexts. That means that the value of the EXI event has a different meaning dependent on the accompanying schema file. The set of schemas that will be used is annouced to the device, such that it can prepare the parsing tables in advance.

5. MIB binary Function Set

The MIB binary Function Set provides a CoAP interface to transport BER encoded SNMP PDUs. Section 5.1 and Section 5.2 explain the structure of SNMP PDUs and their transport with CoMI.

5.1. SNMP architecture

The architecture of the Internet Standard management framework consists of:

- A data definition language that is referred to as Structure of Management Information (SMI)[RFC2578].

- The Management Information Base (MIB) which contains the information to be managed and is defined for each specific function to be managed [RFC3418].

- A protocol definition referred to as Simple Network Management Protocol (SNMP) [RFC3416].

- Security and administration that provides SNMP message based security on the basis of the user-based security model [RFC3414].

Separation in modules was motivated by the wish to respond to the evolution of Internet. The protocol part (SNMP) and data definition part (MIB) are independent of each other. The separation has enabled the progressive passage from SNMPv1 via SNMPv2 to SNMPv3. This draft leverages this separation to replace the SNMP protocol with a CoAP based protocol.

The SNMP protocol supports seven types of access supported by as many Protocol Data Unit (PDU) types:

- Get Request, transmits a list of OBJECT-IDENTIFIERs to be paired with values.

- GetNext Request, transmits a list of OBJECT-IDENTIFIERs to which lexicographic successors are returned for table traversal.

- GetBulk Request, transmits a list of OBJECT-IDENTFIERs and the maximum number of expected paired values.

- Response, returns an error or the (OBJECT-IDENTIFIER, value) pairs for the OBJECT-IDENTIFIERs specified in Get, GetNext, GetBulk, Set, or Inform Requests.

- Set Request, transmits a list of (OBJECT-IDENTIFIERs, value) pairs to be set in the specified MIB object.

- Trap, sends an unconfirmed message with a list of (OBJECT-IDENTIFIERs, value) pairs to a notification requesting end-point.

- Inform Request, sends a confirmed message with a list of (OBJECT-IDENTIFIERs, value) pairs to a notification requesting end-point.

The binding of the notification to a destination (left open in SNMP protocol) is discussed in Section 7.

5.2. CoMI binding

The payload is structured as specified by the ASN-1 notation presented in [RFC3416]. The request PDUs: Get, GetNext, GetBulk, Set, and Inform are sent with a Confirmable CoAP message. The Response is piggy backed in the returned Acknowledgement message. The Trap is sent as a NON-confirmable CoAP message. The URI of the destination CoMI end-point contains the destination identifier and the path /mg/bin. The request PDUs: Get, GetNext, GetBulk are sent with CoAP message type GET, all other request PDUs are sent with CoAP message type PUT. Strictly speaking the request-id in the SNMP PDU has the same function as the CoAP token. Preferably, the value of request-id should be identical to the CoAP token. An example request looks like:

```
REQ: GET example.com/mg/mib/ (Content-Format: application/octet-stream)
<BER encoded SNMP PDU>

RES: 2.05 Content (Content-Format: application/octet-stream)
<BER encoded SNMP PDU>
```


The BER encoding implies that OBJECT-IDENTIFIER is encoded as a sequence of numbers which defines the place of the object in the ISO object tree [RFC2578].

6. CoMI Objects Function Set

Two XML, EXI based interfaces are supported by CoMI: (1) Reading/ Writing CoMI objects with path /mg/comi/object, (2) Reading/Writing

MIB variables with path /mg/comi/mib/variable.

6.1. CoMI object binding

The information, represented by the CoMI objects, is composed of the values of configuration parameters in a device. A CoMI object has a name that identifies it uniquely within the device. The type of a CoMI object MUST be described with an XML schema. CoMI object names and schemas are standardized by the IETF or by other relevant Standards Developing Organizations (SDO). The CoMI object name is expressed in the path. The attributes of the CoMI object that need to be set or read are either expressed in the EXI payload of the message or in the path of the destination URI. By expressing the attribute in the path the data transfer is limited to one resource. An EXI message can express a selection of attributes that need to be set or read. For example, consider the object with name "Alarms" of type "alarmCount":

```
<xs:complexType name="alarmCount">
  <xs:sequence>
    <xs:element name="Priority1" type="xs:integer"
                    minOccurs="0" maxOccurs="1"/>
   <xs:element name="Priority2" type="xs:integer"
                    minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
<xs:complexType>
```

In the coming examples the format of the EXI contents follows the syntax proposed in [EXI-primer]. Retrieving the contents of attribute Priority1 by expressing the attribute in the path is possible with:


```
REQ: GET example.com/mg/comi/Alarms/Priority1
```

(Content-Format: application/exi)

RES: 2.05 Content (Content-Format: application/exi)
SE(alarmCount)SE(Priority1) value EE EE

Retrieving the contents of Priority1 and Priority2 attributes by referring to the object as a whole in the URI is shown in the following example:

REQ: GET example.com/mg/comi/Alarms

RES: 2.05 Content (Content-Format: application/exi)
SE(alarmCount) SE(priority1) value EE SE(priority2) value EE EE

It is possible that the size of the specified object is too large to fit in a single message. CoMI gives the possibility to send the contents of the objects in several fragments with a maximum size. The "sz" link-format attribute [RFC6690] specifies the maximum size in bytes. The returned data MUST terminate with an EE EXI event. The sequel can be asked by sending the same request but with an offset that is equal to the already received number of EE EXI events. The offset is specified with the (new to define) "of" link-format attribute.

6.2. CoMI MIB Function Set

The Function Set allows the access to one MIB variable per CoAP message. The schema is shown in Appendix A. The identifier is based on the human readable name and not on the sequence of numbers that identifies the object in the ISO object hierachy.

TODO: Distinguish CoMI objects of same type and thus with identical names.

A request to read the value of a MIB variable is sent with a confirmable CoAP GET message. A request to set a value is sent with a confirmable CoAP PUT message. The Response is piggybacked to the CoAP ACK message corresponding with the Request. An example request looks like:

REQ: GET example.com/mg/comi/mib/variable

RES: 2.05 Content (Content-Format: application/exi)

SE(MIB)SE(name) variable EE SE(value) value EE EE

A MIB variable may be composed of rows with each row composed of typed values. Limits to the bulk transport are expressed with the "sz" link-format attribute [RFC6690] and the "of" link-format attribute.

When a variable with the specified name cannot be processed by the SNMP server, CoAP Error code 5.01 is returned with the SNMP error code transported in the payload. Two types of error code can be returned: exception or error-status as specified in Appendix A, according to the rules of [RFC3416]. For example when MIB "variable" does not exist:

REQ: GET example.com/mg/comi/mib/variable

RES: 5.01 Not Implemented (Content-Format: application/exi)
SE(exception)noSuchObject EE

The TRAP is sent with a non confirmable CoAP PUT message to a destination specified in Section 7.

7. CoMI objects

Setting up parameter values and establishing relations between devices during commissioning of a managed network is an important aspect of the deployment of CoMI objects. Draft [I-D.ietf-core-interfaces] describes the binding of end-points to end-points on remote devices, and draft [I-D.ietf-core-groupcomm] describes the enabling of multicast messages. A homogeneous approach to the associated resources is the subject of this section. A list of objects describing different aspects of commissioning comprise:

- Binding table as described in [I-D.ietf-core-interfaces], schema presented in Appendix B.2.
- Multicast address enabling as described in [I-D.ietf-core-groupcomm], schema presented in Appendix B.1.
- SNMP notification destinations as referred to in [RFC3416], schema presented in Appendix B.2.
- Names of files containing the schemas to be expected, schema

presented in Appendix B.3.

The object with type "binding table" contains a sequence of bindings. The contents of bindings contains the methods, location, the interval specifications, and the step value as suggested in [I-D.ietf-core-interfaces]. The method "notify" has been added to the binding methods "poll", "obs" and "push", to cater for the binding of SNMP notifications to a target.

The object of type "MCgroups" contains a sequence of MC addresses to be enabled on the interface, specified by MCentry. In these schemas both the host name (group name) and the corresponding IP address are stored. The IP address has been added for those networks where no access to DNS is possible and only the IP address is available. Once the DNS is available and the IP address is brittle, it is recommended to use the host name and not rely on the value of the IP address.

The object of type "Schema-files" contains a sequence of schema files describing the data structure tranportable in CoMI messages.

8. security Considerations

TODO: follows CoAP security provisioning.

9. IANA Considerations

TODO

10. Acknowledgements

Mehmet Ersue and Bert Wijnen explained the encoding aspects of PDUs transported under SNMP. The draft has benefited from comments by Dee Denteneer, Esko Dijk, and Michael van Hartskamp.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

11.2. Informative References

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, December 2002.

[RFC3414] Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", STD 62, RFC 3414, December 2002.

[RFC3416] Presuhn, R., "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3416, December 2002.

[RFC3418] Presuhn, R., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3418, December 2002.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6643] Schoenwaelder, J., "Translation of Structure of Management Information Version 2 (SMIv2) MIB Modules to YANG Modules", RFC 6643, July 2012.

[RFC6650] Falk, J. and M. Kucherawy, "Creation and Use of Email

Feedback Reports: An Applicability Statement for the Abuse Reporting Format (ARF)", RFC 6650, June 2012.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, August 2012.

[RFC6775] Shelby, Z., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, November 2012.

[I-D.ietf-core-coap]
Shelby, Z., Hartke, K., and C. Bormann, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-17 (work in progress), May 2013.

[I-D.ietf-core-groupcomm]
Rahman, A. and E. Dijk, "Group Communication for CoAP", draft-ietf-core-groupcomm-09 (work in progress), May 2013.

[I-D.ietf-core-interfaces]
Shelby, Z. and M. Vial, "CoRE Interfaces", draft-ietf-core-interfaces-00 (work in progress), June 2013.

[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.

[STD0001] , "Official Internet Protocols Standard", Web http://www.rfc-editor.org/rfcxx00.html, .

[EXI] , "Efficient XML Interchange", Web http://www.w3.org/xml/exi, .

[XML] , "Extensible Markup Language (XML)", Web http://www.w3.org/xml, .

[EXI-primer]
Peintner, D. and S. Pericas-Geertsen, "EXI primer", Web

http://www.w3.org/TR/exi-primer, december 2009.

[EXI-measurement]
White, G., KangaSharju, J., Williams, S., and D. Brutzman, "Efficient XML Interchange Measurements Note", Web http://www.w3.org/TR/2007/WD-exi-measurements-20070725, July 2007.

[JSON-XML]
Nurseitov, N., Paulson, M., Reynolds, R., and C. Inzurieta, "Comparison of JSON and XML Data Interchange Formats: A Case Study", Web http://www.cs.montana.edu/izurieta/pubs/caine2009.pdf, 2009.

Appendix A. XML Schema for CoMI MIB

This appendix describes the XML schema that defines the payload contents for MIB requests via the CoMI Function Set. It is assumed that MIB variables are referred by name and not by the oid.

TODO: The schema needs to be updated to define basic types and notifications. Access may be more sophisticated than described here.

Creation and deletion of columns and rows is not catered for. The access modes are not visible.

```
<xs:simpleType name="exception">
   <xs:restriction base="xs:string">
     <xs:enumeration value="noSuchObject"/>
     <xs:enumeration value="noSuchInstance"/>
     <xs:enumeration value="endOfMibView"/>
   </xs:restriction>
</xs:simpleType>

<xs:simpleType name="error-status">
   <xs:restriction base="xs:string">
     <xs:enumeration value="noError"/>
     <xs:enumeration  value="tooBig"/>
     <xs:enumeration  value="noSuchName"/>
     <xs:enumeration  value="badValue"/>
```

```
        <xs:enumeration  value="readOnly"/>
        <xs:enumeration  value="genErr"/>
        <xs:enumeration  value="noAccess"/>
        <xs:enumeration  value="wrongType"/>
        <xs:enumeration  value="wrongLength"/>
        <xs:enumeration  value="wrongEncoding"/>
        <xs:enumeration  value="wrongValue"/>
        <xs:enumeration  value="noCreation"/>
        <xs:enumeration  value="inconsistentValue"/>
        <xs:enumeration  value="resourceUnavailable"/>
        <xs:enumeration  value="commitFailed"/>
        <xs:enumeration  value="undoFailed"/>
        <xs:enumeration  value="authorizationError"/>
        <xs:enumeration  value="notWritable"/>
        <xs:enumeration  value="inconsistentName"/>
      </xs:restriction>
   </xs:simpleType>


   <xs:complexType name ="MIBscalar">
   <xs:sequence>
      <xs:element name="identifier" type="xs:string"/>
     <choice>
           <xs:element name="value" type="xs:integer"/>
           <xs:element name="value" type="xs:string"/>
     </choice>
   </xs:sequence>
   </xs:complexType>

   <xs:complexType name="entryType">
```



```
   <xs:sequence>
        <xs:element name="Type" type="MIBscalar"
                    minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
   </xs:complexType>


   <xs:complexType name="MIBTable">
   <xs:sequence>
        <xs:element name="Entry" type="entryType"
                    minOccurs="0" maxOccurs="unbounded"/>
```

```
</xs:sequence>
</xs:complexType>
```

Appendix B. XML Schema for CoMI objects

This appendix describes the XML schema that defines the payload contents for requests via the CoMI Function Set to the CoMI objects for multicast group, binding table, and SNMP notifications. For the SNMP notifications the Binding Method table specification of [I-D.ietf-core-interfaces] has been extended with "notify".

B.1. Schema for MC groups

MCentries are stored in MCGroups

```
<xs:complexType name="MCentry">
    <xs:element name="groupName" type="xs:string"/>
    <xs:element name="IPaddress" type="xs:string"/>
</xs:complexType>

<xs:complexType name="MCgroups">
<xs:sequence>
      <xs:element name="MCentry" type="MCentry"
                minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
```

B.2. Schema for CoAP binding

Binding table contains several simple Bindings, composed of timing parameters and Function signature.

```
<xs:complexType name="CoAPmethod">
    <xs:restriction base="xs:string">
       <xs:enumeration value="GET"/>
       <xs:enumeration value="PUT"/>
       <xs:enumeration value="POST"/>
    </xs:restriction>
</xs:complexType>

<xs:complexType name="bindingMethod">
    <xs:restriction base="xs:string">
    <xs:enumeration value="poll"/>
    <xs:enumeration value="obs"/>
    <xs:enumeration value="push"/>
    <xs:enumeration value="notify"/>
    </xs:restriction>
</xs:complexType>


<xs:complexType name="invocation">
    <xs:element name="hostname" type="xs:string"/>
    <xs:element name="pathname" type="xs:string"/>
    <xs:element name="IPaddress" type="xs:string"/>
    <xs:element name="bindingMethod" type="bindingMethod"/>
    <xs:element name="CoAPmethod" type="CoAPmethod"/>
</xs:complexType>

<xs:complexType name="simpleBinding">
     <xs:element name="method" type="invocation"/>
     <xs:element name="minPeriod" type="xs:integer"/>
     <xs:element name="maxPeriod" type="xs:integer"/>
     <xs:element name="changeStep" type="xs:integer"/>
</xs:complexType>

<xs:complexType name="binding Table">
    <xs:sequence>
        <xs:element name="simpleBinding" type="simpleBinding"
                    minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

B.3. Valid Schemas

File names are stored in Schema

```
<xs:complexType name="Schema-files">
<xs:sequence>
      <xs:element name="Schema" type="xs:string"
                minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
```

Author's Address

Peter van der Stok
consultant

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