INTERNET-DRAFT
Intended Status: Standards Track
Expires: May 24, 2014

K. Vaughn
Trevilon LLC
A. Triglia
OSS Nokalva, Inc.
R. Rausch
Transcore, LP
November 20, 2013

Coexistence between Condensed Network Management Protocol (CNMP) and the Simple Network Management Protocol (SNMP) draft-vaughn-cnmp-coex-00

Abstract

The purpose of this document is to describe coexistence between the Condensed Network Management Protocol (CNMP) and the various versions of the Simple Network Management Protocol (SNMP).

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Copyright and License Notice

Table of Contents

1. Introduction 3
2. Conventions 3
3. MIB Modules 3
4. Translating Notification Parameters 3
5. Approaches to Coexistence in a Multi-lingual Network 3
5.1. Access to MIB Data 4
5.2. Multi-lingual implementations 4
5.2.1. Command Generator 4
5.2.2. Command Responder 4
5.2.2.1. Integer Encoding 4
5.2.2.2. New Structures 4
5.2.3. Notification Originator 5
5.2.4. Notification Receiver 5
5.3. Proxy Implementations 5
5.3.1. CNMP Upstream and SNMP Downstream 5
5.3.1.1. Dynamic Object Messages 5
5.3.1.2. Other Messages 5
5.3.2. CNMP Downstream and SNMP Upstream 8
5.3.2.1. Dynamic Objects 8
5.3.2.2. Other Messages 8
5.4. Error Status Mappings 11
6. Security Considerations 12
7. IANA Considerations 12
8. References 12
8.1 Normative References 12
8.2 Informative References 12
Authors' Addresses 13

1. Introduction

The purpose of this document is to describe coexistence between the Condensed Network Management Protocol (CNMP) and the Simple Network Management Protocol (SNMP). For a complete overview of CNMP, see [Intro].

There are four general aspects of coexistence described in this document. Each of these is described in a separate section:

- Conversion of MIB documents between SMIv1 and SMIv2 formats is documented in section 3.

- Mapping of notification parameters is documented in section 4.

- Approaches to coexistence between CNMP and SNMP entities in a multi-lingual network is documented in section 5. This section addresses the processing of protocol operations in multi-lingual implementations, as well as behaviour of proxy implementations.

2. Conventions

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 RFC 2119 [RFC2119].

Within this INTERNET-DRAFT and all referenced documents, CNMP is to be considered another version of SNMP. It is only given a different name because the protocol encoding does not follow the same message format as SNMP messages and the protocol will use a different binding with the transport layer.

3. MIB Modules

MIB modules defined using the SMIv1 may continue to be used with CNMP, once they are updated to the SMIv2 format according to the rules defined in Clause 2 of [RFC3584].

4. Translating Notification Parameters

CNMP Notifications use the same format as SNMPv2 and SNMPv3. SNMPv1 notification parameters can between this format and SNMPv1 according to the rules defined in Clause 3 of [RFC3584].

5. Approaches to Coexistence in a Multi-lingual Network

This document defines how to translate between CNMP and SNMPv3.

Messages can be further translated to SNMPv2 or SNMPv1 according to the rules defined in [RFC3584].

5.1. Access to MIB Data

The part of a CNMP agent which actually accesses instances of MIB objects and which actually initiates generation of notifications SHALL be identical to SNMPv2 Access to MIB Data. The conversion between this access and SNMPv1 Access to MIB Data SHALL be as defined in [RFC3584].

5.2. Multi-lingual implementations

A multi-lingual implementation SHALL conform to the following rules.

5.2.1. Command Generator

A command generator SHALL select an appropriate message version when sending requests to another entity. One way to achieve this is to consult a local database to select the appropriate message version.

In addition, a command generator SHALL 'downgrade' GetBulk requests to GetNext requests when selecting SNMPv1 as the message version for an outgoing request. This is done by simply changing the operation type to GetNext, ignoring any non-repeaters and max-repetitions values, and setting error-status and error-index to zero.

5.2.2. Command Responder

A command responder SHALL follow the same rules as defined in Clause 4.2.2 of [RFC3584] with CNMP being treated as another message format for SNMPv2 Access to MIB Data, with the following additional rule:

5.2.2.1. Integer Encoding

When converting a CNMP ObjectValue which is a byte, short, long, ubyte, or ushort to SNMP, the value SHALL be encoded as an SNMP integer-value.

When converting an SNMP integer-value to CNMP, the value SHALL be encoded according to the rules defined in Clause 5.2.2. of [PDU].

5.2.2.2. New Structures

Structures that are new to CNMP do not need to be converted. For example, a GetDynObjX structure can only be generated by CNMP manager. If the agent does not support CNMP, it will ignore the request. A CNMP agent will never receive a GetDynObjX structure from

a known SNMP version. Thus, no conversion rules are necessary.

5.2.3. Notification Originator

A notification originator SHALL follow the same rules as defined in Clause 4.2.2 of [RFC3584] with CNMP being treated as another message format for SNMPv2 Access to MIB Data.

5.2.4. Notification Receiver

There are no special requirements of a notification receiver.

5.3. Proxy Implementations

A proxy implementation may be used to enable communication between one entity that only supports CNMP and another entity that only supports SNMP. This is accomplished in a proxy forwarder application by performing translations on messages. The following sections describe the translations between CNMP and SNMPv3; the rules defined in Clause 4.3 of [RFC3584] MAY be used to further translate to any older version of SNMP.

In the following sections, the 'Upstream' refers to the link between the command generator or notification receiver and the proxy, and 'Downstream' refers to the link between the proxy and the command responder or notification originator, regardless of the PDU type or direction.

NOTE: The proxy may change the port number used for the exchanges in a implementation dependent manner.

5.3.1. CNMP Upstream and SNMP Downstream

5.3.1.1. Dynamic Object Messages

If a Dynamic Object Class PDU (a.k.a. a Simple Transportation Management Protocol [STMP] message) is received, the proxy SHALL forward the received message without change.

5.3.1.2. Other Messages

If a Normal Object Class PDU is received, the proxy SHALL attempt to translate and forward the received message according to the following rules.

5.3.1.2.1. Caching

If the proxy receives an Confirmed Class PDU, the proxy SHALL cache

the following information:
* pduType
* msgID
* request-id, if present
* variable bindings, if present

The proxy may clear messages from the cache in an implementation dependent manner.

5.3.1.2.2. msgVersion

The msgVersion field SHALL be changed to indicate the appropriate version for the translated message.

5.3.1.2.3. msgID

If the proxy receives a message containing an InformResponse-PDU from the Upstream entity, the proxy SHALL search its cache for a pduType of InformRequest-PDU and a msgID where the lowest 16-bits match the value of the incoming msgID. The proxy SHALL forward the cached value of msgID.

For all other messages, the msgID SHALL be truncated to a 16-bit unsigned value before forwarding.

5.3.1.2.4. msgMaxSize

If the msgMaxSize value is greater than 65535, the forwarded msgMaxSize SHALL be 65535; otherwise, the msgMaxSize value shall be forwarded unchanged.

5.3.1.2.5. msgFlags

The msgFlags shall be forwarded unchanged.

5.3.1.2.6. msgSecurityModel

The msgSecurityModel shall be translated in an implementation dependent manner.

5.3.1.2.7. msgSecurityParameters

The msgSecurityParameters shall be updated according to the selected msgFlags, i.e., authenticated and encrypted messages will be 1) authenticated, 2) unencrypted, 3) translated, 4) re-encrypted, and 5) re-authenticated.

5.3.1.2.8. contextEngineID

The contextEngineID, if present, shall be forwarded unchanged. If the contextEngineID is absent, the proxy SHALL determine the default contextEngineID in an implementation dependent manner and forward that value.

5.3.1.2.9. contextName

The contextName, if present, shall be forwarded unchanged. If the contextName is absent, the proxy SHALL determine the default contextName in an implementation dependent manner and forward that value.

5.3.1.2.10. request-id

If the proxy receives a message containing an InformResponse-PDU from the Upstream entity, the proxy SHALL search its cache for a pduType of InformRequest-PDU and a msgID where the lowest 16-bits match the value of the incoming msgID. The proxy SHALL forward the cached value of request-id.

For any other message received from the Upstream entity, the request-id in the Downstream message SHALL be the same as the msgID contained in the Upstream message.

For any message received from the Downstream entity, the request-id value SHALL NOT be forwarded.

5.3.1.2.11. PDU Translations

1) If the proxy receives a message containing a Response-PDU from the Downstream entity, the proxy SHALL search its cache for a msgID that matches the value of the incoming msgID.

   a) If a match is not found, the proxy SHALL NOT forward the message and processing of the message is complete.

   b) If the pduType of the cached message is SetNoReply-PDU, the proxy SHALL NOT forward the message and processing of the message is complete.

   c) If the received message contains a non-zero error-status value, the proxy SHALL forward the error-status value and error-index value in an ErrorResponse-PDU. The errorVariable SHALL be set to "null" (0.0), if the error-index is zero; otherwise, it SHALL be set to the referenced ObjectName.

   d) Otherwise, if the pduType of the cached message is a SetRequest-PDU, the proxy SHALL forward a SetResponse-PDU.

e) Otherwise, the proxy SHALL forward a GetResponse-PDU containing the variable bindings in the received message.

i) The proxy SHALL attempt to translate the ObjectName into the form used in the matched request. If any error occurs in this process, the forwarded message SHALL directly forward the ObjectName by using the fullOID form.

ii)The proxy SHALL translate integer-values according to the rules in 5.2.2.1.

f) Otherwise, the message SHALL NOT be forwarded.

2) If the proxy receives a message containing a Notification Class PDU from the Downstream entity, the proxy SHALL forward the translated PDU contents.

3) If the proxy receives a message containing a Read Class or a Write Class PDU from the Upstream entity, the proxy SHALL forward the translated PDU contents.

a) CNMP PDUs SHALL map to their corresponding SNMP PDU, with the exception that a CNMP SetNoReply-PDU SHALL map to an SNMP SetRequest-PDU.

b) Missing ObjectValue fields from CNMP Read Class PDUs SHALL be replaced with NULL values (i.e., the unSpecified encoding option).

4) If the proxy receives a message containing an InformResponse-PDU from the Upstream entity, the proxy SHALL forward the translated PDU contents as a Response-PDU filling in the variable-bindings field based on the cached information.

5.3.2. CNMP Downstream and SNMP Upstream

5.3.2.1. Dynamic Objects

If a Dynamic Object Class PDU (a.k.a. a Simple Transportation Management Protocol [STMP] message) is received, the proxy SHALL forward the received message without change.

5.3.2.2. Other Messages

If a Normal Object Class PDU is received, the proxy SHALL attempt to translate and forward the received message according to the following rules.

5.3.2.2.1. Caching

If the proxy receives an Confirmed Class PDU, the proxy SHALL cache the following information:
* pduType
* msgID
* request-id, if present
* variable bindings, if present

The proxy may clear messages from the cache in an implementation dependent manner.

5.3.2.2.2. msgVersion

The msgVersion field SHALL be changed to indicate the appropriate version for the translated message.

5.3.2.2.3. msgID

If the proxy receives a message containing an Response Class PDU from the Downstream entity, the proxy SHALL search its cache for a Read or Write Class PDU with a msgID where the lowest 16-bits match the value of the incoming msgID. The proxy SHALL forward the cached value of msgID.

For all other messages, the msgID SHALL be truncated to a 16-bit unsigned value before forwarding.

5.3.2.2.4. msgMaxSize

If the msgMaxSize value is greater than 65535, the forwarded msgMaxSize SHALL be 65535; otherwise, the msgMaxSize value shall be forwarded unchanged.

5.3.2.2.5. msgFlags

The msgFlags shall be forwarded unchanged.

5.3.2.2.6. msgSecurityModel

The msgSecurityModel shall be translated in an implementation dependent manner.

5.3.2.2.7. msgSecurityParameters

The msgSecurityParameters shall be updated according to the selected msgFlags, i.e., authenticated and encrypted messages will be 1) authenticated, 2) unencrypted, 3) translated, 4) re-

encrypted, and 5) re-authenticated.

5.3.2.2.8. contextEngineID

The contextEngineID, if present, shall be forwarded unchanged. If the contextEngineID is absent, the proxy SHALL determine the default contextEngineID in an implementation dependent manner and forward that value.

5.3.2.2.9. contextName

The contextName, if present, shall be forwarded unchanged. If the contextName is absent, the proxy SHALL determine the default contextName in an implementation dependent manner and forward that value.

5.3.2.2.10. request-id

If the proxy receives a message containing an Response Class PDU from the Downstream entity, the proxy SHALL search its cache for a Read or Write Class PDU with a msgID where the lowest 16-bits match the value of the incoming msgID. The proxy SHALL forward the cached value of request-id.

For any other message received from the Downstream entity, the request-id in the Upstream message SHALL be the same as the msgID contained in the Downstream message.

For any message received from the Upstream entity, the request-id value SHALL NOT be forwarded.

5.3.2.2.11. PDU Translations

1) If the proxy receives a message containing a Read Class or a Write Class PDU from the Upstream entity, the proxy SHALL forward the translated PDU contents.

   a) The proxy MAY use more compact forms to encode the ObjectName.

   b) The proxy SHALL translate integer-values according to the rules in 5.2.2.1.

2) If the proxy receives a message containing an Response-PDU from the Upstream entity, the proxy SHALL search its cache for a msgID that matches the value of the incoming msgID.

   a) If a match is not found, the proxy SHALL NOT forward the message and processing of the message is complete.

b) If the pduType of the cached message is InformRequest-PDU, the proxy SHALL forward the translated PDU contents as a InformResponse-PDU.

c) Otherwise, the proxy SHALL NOT forward the message and processing of the message is complete.

3) If the proxy receives a message containing a Response Class PDU from the Downstream entity, the proxy SHALL search its cache for a Read or Write Class PDU with a msgID where the lowest 16-bits match the value of the incoming msgID.

a) If a match is not found, the proxy SHALL NOT forward the message and processing of the message is complete.

b) If the received message contains an ErrorResponse-PDU, the proxy SHALL forward the error-status value and error-index value in an Response-PDU. The proxy SHALL include the variable-bindings information from the cache.

c) Otherwise, if the received message contains an SetResponse-PDU, the proxy SHALL forward a Response-PDU. The proxy SHALL include the variable-bindings information from the cache.

d) Otherwise, if the received message contains an GetResponse-PDU,the proxy SHALL forward a Response-PDU containing the information from the VarBinds.

i) The proxy SHALL translate all ObjectNames into their full OIDs.

ii)The proxy SHALL translate long, short, byte, ubyte, and ushort values to integer-values.

f) Otherwise, the message SHALL NOT be forwarded.

4) If the proxy receives a message containing a Notification Class PDU from the Downstream entity, the proxy SHALL forward the translated PDU contents.

5.4. Error Status Mappings

All error codes SHALL be passed without translation, with the exception of the following CNMP error codes that do not exist in SNMPv3. These error codes SHALL be translated as follows.

CNMP error-status	SNMPv3 error-status
invalidName	inconsistentName
processTimeout	resourceUnavailable

6. Security Considerations

Security issues for each protocol are defined within the context of each protocol definition. When protocols co-exist, the security of the system will be defined by the weakest link in the system.

It is recommended that the implementors consider the security features as provided by the CNMP and SNMPv3 framework. Specifically, the use of the User-based Security Model STD 62, RFC 3414 [RFC3414], the Transport Security Model [RFC5590], and the View-based Access Control Model STD 62, RFC 3415 [RFC3415] is recommended.

It is then a customer/user responsibility to ensure that the entity giving access to a MIB is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change) them.

7. IANA Considerations

This document has no IANA actions.

8. References

8.1 Normative References

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

[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.

[RFC3415] Wijnen, B., Presuhn, R., and K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3415, December 2002.

[RFC3584] Frye, R., Levi, D., Routhier, S., and B. Wijnen, "Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework", BCP 74, RFC 3584, August 2003.

[RFC5590] Harrington, D. and J. Schoenwaelder, "Transport Subsystem for the Simple Network Management Protocol (SNMP)",

RFC 5590, June 2009.

[PDU] Vaughn, K., "Protocol Operations for Condensed Network Management Protocol (CNMP)", Internet-Draft draft-vaughn-cnmp-pdu-00, November 2013.

8.2 Informative References

[Intro] Vaughn, K., "Document Roadmap for Condensed Network Management Protocol (CNMP)", Internet-Draft draft-vaughn-cnmp-intro-00, November 2013.

[STMP] "Transport Management Protocols", published by American Association of State Highway Officials (AASHTO), Institute of Transportation Engineers (ITE), and National Electrical Manufacturers Association (NEMA). NTCIP (National Transportation Communications for ITS Protocol) 1103v02.17, July 2010.

Authors' Addresses

Kenneth Vaughn
Trevilon LLC
6606 FM 1488 RD
STE 148-503
Magnolia, TX 77316
USA

Phone: +1-571-331-5670
Email: kvaughn@trevilon.com

Alessandro Triglia
OSS Nokolva, Inc.
1 Executive Drive
Suite 450
Somerset, NJ 08873

Email: sandro@oss.com

Robert Rausch
Transcore, LP
192 Technology Parkway
Suite 500
Norcross, GA 30092

Email: robert.rausch@transcore.com