Message Processing and Control Model for version 3 of the Simple Network Management Protocol (SNMPv3)

17 June 1997

J. Case
Snmp Research Inc.
case@snmp.com

D. Harrington
Cabletron Systems, Inc.
dbh@cabletron.com

B. Wijnen
IBM T. J. Watson Research
wijnen@vnet.ibm.com

<draft-ietf-snmpv3-v3mpc-model-01.txt>

Status of this Memo

Abstract

This document describes the SNMP version 3 Message Processing and Control Model for use in the SNMP architecture [SNMP-ARCH]. This model defines the SNMP version 3 message format, and the procedure for coordinating the processing of a message in SNMPv3 format.

0. Issues

. Interactions with Applications needs more synchronization with
SNMPv3 applications (orangelets) document
I am using "applications" instead of "orangelets". DaveL c.s.
are talking about applications and not about orangelets
. Interactions with Security Model needs more synchronization with
SNMPv3 USEC document. Statistics counters have already been
moved to this MPC document and have now generic names that
should fit any security model.
. Do we need to keep the security threats in section 5?
. Should statistics counters from ACM be moved to MPC ??
Bert thinks YES.
. Need to check the SNMP-MPC-MIB with a MIB compiler
. Need to decide if we want to enumerate Security and Application
status Codes, so that we can document in which case a specific
statsCounter gets incremented and a reportPDU gets sent.
Could decide that such is implementation matter.
. acknowledgements needs expansion
. sendRequestPdu() primitive
should we allow the application to pass stateReference
which the MPC could save with its state info about an
outstanding SNMP request message, so that when it returns
the response message it can pass back a stateReference that
the application can use to correlate a response to a request.
. scopedPDU defines contextEngineID as SIZE(0|12). Do we need 0?
. should the architecture require that all message formats have
the version number first?
. should globalData parameters be expanded in primitive?
. should we use the name primitive, or call them abstract service
interfaces?
. when matching a response to the outstanding requests, what
info must be checked? msgID, contextEngineID? Does the MPC need
to look at the verb directly?

0.1. Change Log

[version 2.1]
 . spell-check
 . removed references to groupName
 . replaced discussion of threats with reference to [SNMP-ARCH]
 . moved unresolved co-editor questions into Issues list
 . worked on improving consistency internally and with other documents
 . changed errorCode, errorStatus, etc to statusCode.
 . modified 4.7.1. discussion
 . published as draft-ietf-snmpv3-mpc-model-01.txt

[version 2.0]
    . changes as a result of 1st interim meeting
      . some new wording in introduction
      . rewording in overview with a drawing
      . added reportFlag to msgFlags

      . describe overall MPC model: MPC Selection mechanism
      . describe overall MPC model: MPC Multiplexing Layer
      . describe v3MPC model.
      . added the abstract interface definitions for interacting
        with SNMPv3 USEC Model
      . added the abstract interface definitions for interacting
        with applications
      . added MIB definitions for error Counters (statistics)
      . removed references to LPM and Access Control
      . Bert added as editor (thank you for the help, bert - dbh)

1. Introduction

The Architecture for describing Internet Management Frameworks is composed of multiple subsystems:

1) a message processing and control subsystem,
2) a security subsystem,
3) an access control subsystem, and
4) orangelets.

It is important to understand the SNMP architecture and the terminology of the architecture to understand where the model described in this document fits into the architecture and interacts with other subsystems within the architecture. The reader is expected to have read and understood the description of the SNMP architecture, as defined in [SNMP-ARCH].

The Message Processing and Control subsystem of an SNMP engine interacts with the network using SNMP messages, interacts with applications using data elements defined by the Message Processing

and Control model, within the constraints of the Architecture, and interacts with other models within the constraints of the architecture.

A Message Processing and Control model has the responsibility for coordinating the sending and receiving of SNMP messages, and for coordinating the interaction with other subsystems to acquire the desired services for the processing of a message.

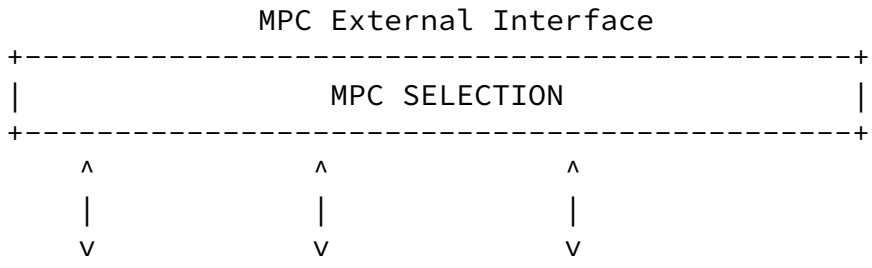
It is the purpose of this document to define:

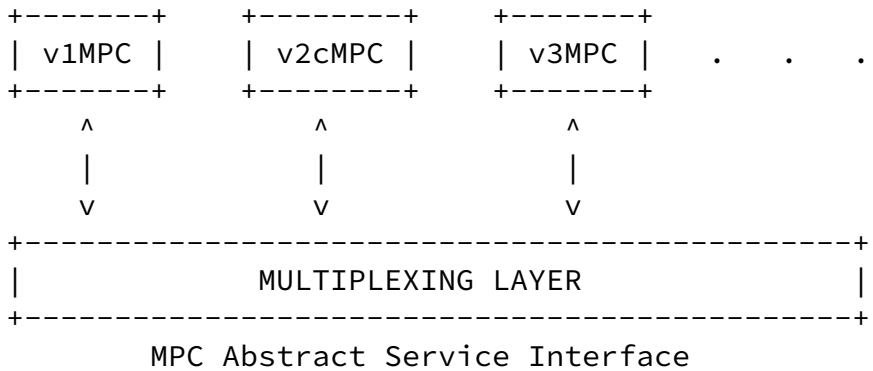
- the overall MPC Model, including the MPC Selection and the MPC Multiplexing Layer
- a specific model of Message Processing and Control subsystem, designated the SNMP version 3 Message Processing and Control (v3MPC) model.

Other (future) documents can add specific models of an MPC for other SNMP versions (like a v1MPC and/or a v2cMPC) which then fit into the overall MPC model described in this document.

2. Overview

The following illustration depicts the MPC:





2.1. MPC External Interface

The SNMP Message Processing and Control model is part of an SNMP meesageEngine which interacts with the network. It can possibly handle multiple SNMP Message formats (like the SNMPv1 [RFC1157], SNMPv2c [RFC1901] and SNMPv3 message formats). The MPC Selection mechanism receives and sends messages from/to the network. For incoming messages it determines the specific message format of an incoming SNMP message and then selects the corresponding version-specific MPC module to further process the SNMP message.

The transport mappings for sending and receiving messages are defined in [RFC1906].

This document defines the SNMP version 3 MPC (v3MPC) model and the corresponding SNMPv3 message format. Other documents may specify additional version-specific MPC models.

The MPC model, while sending and receiving SNMP messages, collects statistics about SNMP messages and the behavior of the SNMP engine. These statistics are maintained in managed objects to make it accessible to remote SNMP engines. This document will define the managed objects, and the MIB module which contains them. This document will also describe how these managed objects might be used to provide useful network management.

2.2. MPC Abstract Service Interface

The SNMP Message Processing and Control model is a specification of a subsystem within an SNMP messageEngine which interacts with other subsystems, requesting services from, and performing services for,

those other subsystems.

The v3MPC module interacts with one or more SNMPv3 Security Models to ensure that the proper security measures (authentication, timeliness checking, en/decryption) are applied to the messages being sent and received. It uses the abstract service interface to the security model, as defined by the SNMP architecture, to request services from such a security module.

The v3MPC is designed to guarantee certain behaviors and to meet a particular set of requirements. It will attempt to achieve its goals by controlling the processing of messages, sometimes by defining a fixed behavior that must be adhered to by a conforming implementation of this model, sometimes by defining options which can be specified by the developer or by the user to cause processing to meet optional requirements.

This document describes how the internal interactions are coordinated and constrained by the v3MPC model.

Once an incoming message has passed the version-specific MPC processing (like the v3MPC processing) and is ready for processing by an application, the MPC uses a Multiplexing Layer which hides the differences between the various message formats of the different SNMP versions. It uses the abstract service interface to the applications, as defined by the SNMP architecture, to request or provide services from or to the applications.

3. Transport Mappings

The MPC model recognizes the transport mappings defined in RFC 1906, and may support additional transport mappings defined in other documents.

During installation, an SNMP engine acting in an agent role is configured with one or more transport service addresses

> These parameters may be specified explicitly, or they may be specified implicitly as the same set of network-layer addresses configured for other uses by the device together with the well-known transport-layer "port" information for the appropriate transport domain [RFC1906]. The agent listens on each of these transport service addresses for messages.

4. The SNMPv3 message format

This is the format of a message to be processed by the v3MPC.

```
DEFINITIONS ::= BEGIN

snmpV3Message ::= SEQUENCE {
        -- administrative parameters (globalData)
        version    INTEGER { snmpv3 (3) },
        msgID      INTEGER (0..2147483647),
        mms        Integer32 (484..2147483647),

        msgFlags OCTET STRING (SIZE(1)),
                   --  .... ..00   noAuth/noPriv
                   --  .... ..01   auth/noPriv
                   --  .... ..10   auth/priv
                   --  .... ..11   reserved
                   --  .... .1..   reportableFlag
                   --  .... 1...   reportFlag

        securityModel SnmpSecurityModel,

        -- security model-specific parameters
        -- format defined by Security Model
        securityParameters OCTET STRING (SIZE(0..2048)),

        -- local-processing model-specific data
        data  ScopedPduData
    }

    ScopedPduData ::= CHOICE {
            plaintext ScopedPDU,
            encrypted OCTET STRING    -- encrypted value
        }

    scopedPDU ::= SEQUENCE {
            contextEngineID  OCTET STRING (SIZE(0|12)),
            contextName OCTET STRING (SIZE(0..32)),
```

```
            data
                ANY -- e.g. PDUs as defined in RFC1905
        }
END
```

4.1. SNMP version

The SNMP version field set to snmpv3(3) identifies the message as an snmpV3Message.

4.2. msgID

The msgID is used by the v3MPC to coordinate the processing of the

message by different subsystem models within the architecture.

Note that the request-id [RFC1905] used during local processing identifies the request, not the message that carried the request, and therefore might not be equal to the msgID.

4.3. MMS

The maximum message size supported by the sender of the message. That is the maximum message size that the sender can accept when another SNMP engine sends an SNMP message (be it a response or any other message) to the sender of this message.

When a request message is being generated, the MMS is provided by the engine which generates the message. When a response message is being generated, the MMS from the request message is used by the v3MPC to determine the maximum size that is acceptable to send.

4.4. msgFlags

The msgFlags field contains flags to control the processing of the message.

If the reportableFlag is set, then reportPDUs are allowed to be returned to the sender under those conditions which cause the generation of reportPDUs. If the reportableFlag is zero, then a reportPDU must not be sent. The reportableFlag should always be zero when the message is a report, a response, or a trap.

The reportFlag is set for report-PDUs so that the v3MPC can recognize

such a reportPDU and process it internally.

It is the v3MPC module that generates and processes reportPDUs.

If the auth flag is set, then the security implementation is required to identify the securityIdentity on whose behalf a request is generated and to authenticate such identification. If the auth flag is zero, authentication of the securityIdentity is not required.

If the priv flag is set, then the security implementation is required to protect the data within an SNMP operation from disclosure, i.e. to encrypt the data. If the priv flag is zero, then the security implementation does not need to protect the data using encryption.

It is an explicit requirement of the SNMP Architecture that if privacy is selected, then authentication of the identification is required, i.e. priv flag can only be set if auth flag is also set.

The combination of the auth flag and the priv flag comprises a Level of Security (LoS), as defined in [SNMP-ARCH].

4.5. securityModel

The v3MPC supports the concurrent existence of multiple security models to provide security services for snmpV3Messages. The securityModel identifies which security model should be used to provide security processing for the message. The mapping to the appropriate implementation within an SNMP messageEngine is done in an implementation-dependent manner.

4.6. security parameters

This octet string is not interpreted by the v3MPC. This abstract data element is used by the v3MPC to transfer security-model-specific data from the snmpV3Message to the security subsystem model indicated by the securityModel field in the message. This data is used exclusively by the security model, and the contents and format of the data is defined by the security model.

4.7. scopedPDU

A scopedPDU contains a PDU and information to identify an administratively unique context. The object identifiers in the PDU refer to managed objects which are expected to be accessible within the specified context.

4.7.1. contextEngineID

A contextEngineID is the unique identifier of the SNMP contextEngine that has access to the managed objects referenced in the PDUs.

The v3MPC will compare the contextEngineID to the registered contextEngineIDs to determine to which application the scopedPDU should be forwarded.

4.7.2. contextName

This is the name of the locally-unique context, within the engine specified by the contextEngineID, which realizes the managed objects referenced within the PDUs.

4.7.3. data

The data contains the PDUs. The v3MPC specifies that the PDUs are those as specified in RFC1905.

5. Services of the SNMP Message Processing and Control Model

5.1. Services of the MPC Selection process

5.1.1. Receiving an SNMP message from the network

The MPC Selection process when it receives an SNMP message from the network first increments the snmpInPkts counter [RFC1907]. If the

received message is not the serialization (according to the conventions of [RFC1906]) of a Message value, then the snmpInASNParseErrs counter [RFC1907] is incremented, and the message is discarded without further processing.

The MPC Selection process determines the SNMP version of an incoming message. If the version is not supported by this SNMP messageEngine (e.g. there is no version-specific MPC for this version) then the snmpInBadVersions counter [RFC1907] is incremented, and the message is discarded without further processing.

Based on the SNMP version of the message, the MPC passes the message on to the appropriate version-specific MPC. Before doing so, it caches the origin network address information, so that a possible response can be sent back to the sender of the message.

5.1.2. Sending an SNMP message onto the network

The MPC Selection process gets called by the version-specific MPC to send an SNMP message onto the network to a specified destination.

The MPC selection process sends the message to the specified destination. It then advises the calling version-specific MPC about the success or failure of the sending of the message.

5.2. Services of the v3MPC Model

5.2.1. Interacting with a Security Model to Protect against Threats

Several of the classical threats to network protocols are applicable to the network management problem and therefore would be applicable to any SNMP security model. These are described in [SNMP-ARCH].

5.2.1.1. Receive SNMPv3 messages from the network

Upon receipt of an SNMPv3 message from the network (passed to the v3MPC by the MPC Selection process), this SNMPv3 Message Processing and Control model extracts and caches the version, msgID, MMS, and the msgFlags, determines the LoS, and determines where the block of security parameters start in the message.

The v3MPC, in an implementation-defined manner, establishes a mechanism for coordinating processing regarding this received

\

message, e.g. it may assign a "handle" to the message and it's cached data.

The SNMPv3 Message Processing and Control model passes the various header fields. the LoS, the whole message and it's length, and the block of security parameters to the implementation of the Security Model specified in the message header. The abstract service interface for this is:

```
processMsg(version, msgID, mms, msgFlags,
           securityModel, securityParameters,
           LoS, wholeMsg, wholeMsgLen)
```

The Security Model, after completion of its processing, returns to the Message Processing and Control implementation a status code and, if the message passed all security checks, a MIID, a cachedSecurityDataReference, a scopedPDUmms, and the scopedPDU itself. The abstract service interface for this is:

```
returnMsg(MIID, cachedSecurityDataReference,
          scopedPDUmms, scopedPDU, statusCode)
```

In the event of an error in security processing (as indicated by the statusCode), the v3MPC module updates the proper error counter, generates a reportPDU and discards the message without further processing.

If successful, the v3MPC adds the cachedSecurityDataReference to it's set of cached information.

If the reportFlag is set, then the v3MPC assumes that this is an SNMPv3 error report and parses the reportPDU. It then takes appropriate action (like resending the request in case of a notInTimeWindows report or advising the original application about a failed request).

If the reportFlag is not set, then the v3MPC determines if this message is a response to an outstanding request. To do so it checks if there is an outstanding request message for the msgID and contextEngineID contained in the message.

The v3MPC passes this message, together with an indication if this is a response message or not, to the MPC Multiplexing layer, which will determine what to do with the payload of the message. If this is a response message, then any cached data for this msgID is discarded, the message processing part of the transaction is complete.

If the Multiplexing layer returns an error in the statusCode, then an appropriate reportPDU is generated and the message (including any cached data) is discarded without further processing.

5.2.1.2. Send SNMPv3 messages to the network

The MPC Multiplexing Layer passes a request to the v3MPC to be sent out as an SNMPv3 message. The v3MPC interacts with the SNMPv3 Security Model to secure the outgoing message. There are 2 types of SNMPv3 messages to consider:

a. An SNMP request or notification. In this case, the abstract service interface is:

```
generateRequestMsg(version, msgID, mms, msgFlags,
                   securityModel, securityParameters,
                   LoS, MIID, engineID, scopedPDU)
```

The v3MPC module fills out the globalData of the SNMPv3 message before calling upon the generateRequestMsg service. It does so as follows:

- the version is set to snmpv3 (3).
- a unique msgID is generated. It is best to use random numbers for the msgID.
- the mms is set to the value of snmpEngineMaxMessageSize.0
- the msgFlags are set:
  - the auth and priv flags are set according to the LoS requested.
  - the reportableFlag is set to 1
  - the other bits of the msgFlags octet are set to zero.
- the securityModel is set to SnmpV3UsecModel

b. An SNMP response. In this case, the abstract service interface is:

```
generateResponseMsg(version, msgID, mms, msgFlags,
                    securityModel, securityParameters,
                    scopedPDU, cachedSecurityDataReference)
```

The v3MPC module fills out the global data based on the cached information it saved from the incoming request to which this is

a response. The cachedSecurityDataReference is also picked up from that cache.

The Security Model will construct the message, and return the completed message to the Message Processing and Control model. The abstract service interface is:

returnGeneratedMsg(wholeMsg, wholeMsgLen, statusCode)

If the statusCode indicates success, then the v3MPC passes the completed message to the MPC Selection process for sending it to the destination. If the sending of the message succeeds, then the MPC Multiplexing Layer is advised that the sending of the SNMP message was successful.

If the statusCode indicates an error, or if the actual sending of the message failed, then the MPC Multiplexing Layer is advised that the sending of the SNMP message failed.

If the message is a request, then the msgID, the target snmpEngineID and the target contextEngineID are saved in a cache, so that when a response is received, it can be matched up to an outstanding request.

5.3. Services of the MPC Multiplexing Layer

5.3.1. Application Registration for handling payloads (PDUs)

Applications must register, with the MPC multilexing layer, the contextEngineIDs for which they wish to receive PDUs delivered in asynchronous messages.
The abstract interfaces for this are:

boolean register_contextEngineID(contextEngineID)

unregister_contextEngineID(contextEngineID)

Only one application can register to provide asynchronous support for a contextEngineID. If a second one tries to register then a FAILED error code is returned. Otherwise a SUCCESS code is

returned.

The unregister primitive does not return an error code. If unregister primitive is called for a non-registered contextEngineID, then the request is ignored.

5.3.2. Sending the payload of an SNMP Message to an application

The MPC Model determines by which application a scopedPDU should be processed.

If the message is a response to an outstanding request, then the response is passed to the application that issued the request. The abstract service interface for that is:

```
processResponsePdu(contextEngineID, contextName, PDU, LoS,
                   status_code)
```

If the message is not a response to an outstanding request, the MPC checks, in an implementation-dependent manner, which application registered for the contextEngineID contained in the scoped PDU. If no application registered for it, then the snmpMPCUnknownContextEngineIDs counter is incremented and an statusCode is returned to the calling version-specific MPC.

Otherwise the registered application is called to handle the scopedPDU. The abstract service interface is:

```
processPdu(contextEngineID, contextName, PDU, PDU-MMS,
           LoS, securityModel, MIID, stateReference)
```

When such an application finishes processing the PDU they must return a statusCode and possibly a response. The abstract service interface is:

```
returnPdu(contextEngineID, contextName, PDU, LoS,
          stateReference, statusCode)
```

If the statusCode indicates an error, then the stateReference data is discarded, a possible error counter is incremented and the error

code is passed on to the version-specific MPC for further processing.

5.3.3. Applications sending SNMP requests

When an application wants to send an SNMP request to another SNMP engine, it can call upon the services of the MPC Multiplexing Layer. The abstract service interface is:

```
   sendRequestPdu(TDomain, TAddress, snmpVersion,
                  LoS, securityModel, MIID, contextEngineID,
                  contextName, PDU)
```

The MPC keeps state information about where the request came from and then passes the message up to the version-specific MPC for further processing.

When an application wants to send an SNMP message that will not result in an SNMP response message (like a trap), it can call upon the services of the MPC Multiplexing Layer. The abstract service interface is:

```
   sendPdu(TDomain, TAddress, snmpVersion,
           LoS, securityModel, MIID, contextEngineID,
           contextName, PDU)
```

The MPC passes the message on to the version-specific MPC and then discards the information.

6. Definitions

6.1. Definitions for the SNMP Message Processing and Control Model

```
SNMP-MPC-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, snmpModules,
    Counter32                                   FROM SNMPv2-SMI;
    TEXTUAL-CONVENTION,
    TDomain, TAddress                           FROM SNMPv2-TC;
    MODULE-COMPLIANCE, OBJECT-GROUP             FROM SNMPv2-CONF;
    snmpEngineID, SnmpSecurityModel,
    SnmpAdminString                             FROM IMF-MIB;


snmpMPCMIB MODULE-IDENTITY
    LAST-UPDATED "9706170000Z"     -- 17 June 1997, midnight
    ORGANIZATION "SNMPv3 Working Group"
    CONTACT-INFO "WG-email:   snmpv3@tis.com
                  Subscribe:  majordomo@tis.com
                              In message body:  subscribe snmpv3

                  Chair:      Russ Mundy
                              Trusted Information Systems
                  postal:     3060 Washington Rd
                              Glenwood MD 21738
                  email:      mundy@tis.com
                  phone:      301-854-6889

                  Co-editor:  Dr. Jeffrey Case
                              Snmp Research International, Inc.
                  postal:
                  phone:

                  Co-editor   Dave Harrington
                              Cabletron Systems, Inc
                  postal:     Post Office Box 5005
                              MailStop: Durham
                              35 Industrial Way
                              Rochester NH 03867-5005
                  email:      dbh@cabletron.com
                  phone:      603-337-7357

                  Co-editor:  Bert Wijnen
                              IBM T. J. Watson Research
                  postal:     Schagen 33
                              3461 GL Linschoten
                              Netherlands
                  email:      wijnen@vnet.ibm.com
                  phone:      +31-348-432-794
```

```
                "
    DESCRIPTION  "The snmp MPC MIB"
    ::= { snmpModules xx }

-- Administrative assignments

snmpMPCMIBObjects      OBJECT IDENTIFIER ::= { snmpMPCMIB 1 }
snmpMPCMIBConformance  OBJECT IDENTIFIER ::= { snmpMPCMIB 2 }

-- Statistics for MPC Model ****************************************

snmpMPCStats           OBJECT IDENTIFIER ::= { snmpMPCMIBObjects 1 }

snmpMPCStatsUnknownContextEngineIDs OBJECT-TYPE
    SYNTAX       Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION "The total number of packets received by the SNMP
                 engine which were dropped because they referenced a
                 contextEngineID that was not known to the engine
                 (e.g. was not registered by any application).
                "
    ::= { snmpMPCStats 1 }

snmpMPCStatsUnsupportedLoS OBJECT-TYPE
    SYNTAX       Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION "The total number of packets received by the SNMP
                 engine which were dropped because they requested a
                 Level of Security that was unknown to the engine or
                 otherwise unavailable.
                "
    ::= { snmpMPCStats 2 }

snmpMPCStatsNotInTimeWindows OBJECT-TYPE
    SYNTAX       Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION "The total number of packets received by the SNMP
                 engine which were dropped because they appeared
                 outside of the engine's window.
                "
    ::= { snmpMPCStats 3 }
```

```
snmpMPCStatsUnknownSecurityIdentities OBJECT-TYPE
    SYNTAX       Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION "The total number of packets received by the SNMP
```



```
                engine which were dropped because they referenced a
                security model specific securityIdentity that was
                not known to the engine.
               "
    ::= { snmpMPCStats 4 }


snmpMPCStatsAuthenticationErrors OBJECT-TYPE
    SYNTAX       Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION "The total number of packets received by the SNMP
                engine which were dropped because they could not be
                authenticated (i.e. the MAC did not match).
               "
    ::= { snmpMPCStats 5 }

snmpMPCStatsPrivacyErrors OBJECT-TYPE
    SYNTAX       Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION "The total number of packets received by the SNMP
                engine which were dropped because they could not be
                decrypted.
               "
    ::= { snmpMPCStats 6 }

-- Conformance information

snmpMPCMIBCompliances  OBJECT IDENTIFIER ::=
                               { snmpMPCMIBConformance 1 }
snmpMPCMIBGroups       OBJECT IDENTIFIER ::=
                               { snmpMPCMIBConformance 2 }
```

```
-- Compliance statements

snmpMPCMIBCompliance MODULE-COMPLIANCE
    STATUS       current
    DESCRIPTION "The compliance statement for SNMP entities which
                 implement the SNMP MPC MIB.
                "

    MODULE    -- this module
        MANDATORY-GROUPS { snmpMPCBasicGroup }

    ::= { snmpMPCMIBCompliances 1 }

snmpMPCBasicGroup OBJECT-GROUP
    OBJECTS {
              snmpMPCStatsUnknownContextEngineIDs,
```



```
              snmpMPCStatsUnsupportedLoS,
              snmpMPCStatsNotInTimeWindows,
              snmpMPCStatsUnknownSecurityIdentities,
              snmpMPCStatsAuthenticationErrors,
              snmpMPCStatsPrivacyErrors
            }
    STATUS       current
    DESCRIPTION "A collection of objects providing for remote monitoring
                 of an implementation of an SNMP Message Processing and
                 Control model.
                "
    ::= { snmpMPCMIBGroups 1 }

END
```

7. Security Consideration

The SNMP Message Processing and Control model coordinates the processing of messages to provide a level of security for network management messages and to direct the SNMP Message Payload to the proper SNMP application.

The level of security actually provided is primarily determined by the specific Security Model implementations and the specific SNMP application implementations incorporated into this framework. Applications have access to data which is not secured. Applications should take reasonable steps to protect the data from disclosure, and

when they send data across the network, they should obey the LoS and call upon the Access Control Model services to apply access control.

8. Glossary

LoS
MIID
MMS
MPC
PDU
SNMP
ScopedPDU
ScopedPduData
SnmpAdminString
SnmpSecurityModel
SnmpV3UsecModel
TAddress
TDomain
USEC
auth
cachedSecurityDataReference
contextEngine
contextEngineID
contextName
engineID
generateRequestMsg
generateResponseMsg
globalData
meesageEngine
messageEngine
MIID
mms
msgFlags
msgID
noAuth
noPriv
notInTimeWindows
orangelets
plaintext
priv
processMsg
processPdu
processResponsePdu
reportFlag
reportPDU
reportPDUs
reportableFlag
responsePDU
returnGeneratedMsg
returnMsg
returnPdu
scopedPDU
scopedPDUmms
securityIdentity

securityModel
securityParameters
sendPdu
sendRequestPdu
snmpEngineID
snmpEngineMaxMessageSize.0
snmpModules
snmpV3Message
snmpV3Messages
snmpVersion
snmpv3
stateReference
statsCounter
statusCode
v1MPC - an MPC model designed to be compatible with RFC1157
v2cMPC - an MPC model designed to be compatible with RFC1901
v3MPC - the MPC model described in this document
wholeMsg - a complete message
wholeMsgLen - the length of a complete message

9. References

[RFC1901] The SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S., Waldbusser, "Introduction to Community-based SNMPv2", RFC 1901, January 1996.

[RFC1902] The SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S., Waldbusser, "Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1902, January 1996.

[RFC1905] The SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S., Waldbusser, "Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1905, January 1996.

[RFC1906] The SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1906, January 1996.

[RFC1907] The SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1907 January 1996.

[RFC1908] The SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework", RFC 1908, January 1996.

[SNMP-ARCH] The SNMPv3 Working Group, Harrington, D., Wijnen, B., "An Architecture for describing Internet Management Frameworks", draft-ietf-snmpv3-next-gen-arch-02.txt, June 1997.

[SNMPv3-MPC] The SNMPv3 Working Group, Wijnen, B., Harrington, D., "Message Processing and Control Model for version 3 of the Simple Network Management Protocol (SNMPv3)", draft-ietf-snmpv3-mpc-01.txt, June 1997.

[SNMPv3-ACM] The SNMPv3 Working Group, Wijnen, B., Harrington, D., "Access Control Model for Version 3 of the Simple Network Management Protocol (SNMPv3)", draft-ietf-snmpv3-acm-00.txt, June 1997.

[SNMPv3-USEC] The SNMPv3 Working Group, Blumenthal, U., Wijnen, B. "User-Based Security Model for version 3 of the Simple Network Management Protocol (SNMPv3)", draft-ietf-snmpv3-usec-01.txt, June 1997.

10. Editor's Addresses

Co-editor: Dr. Jeffrey Case
Snmp Research International, Inc.
postal:
email: case@snmp.com
phone:

Co-editor Dave Harrington
Cabletron Systems, Inc
postal: Post Office Box 5005
MailStop: Durham
35 Industrial Way
Rochester NH 03867-5005
email: dbh@cabletron.com
phone: 603-337-7357

Co-editor: Bert Wijnen
IBM T. J. Watson Research

postal: Schagen 33
3461 GL Linschoten
Netherlands
email: wijnen@vnet.ibm.com
phone: +31-348-432-794

11. Acknowledgements

This document builds on the work of the SNMP Security and Administrative Framework Evolution team, comprised of

David Harrington (Cabletron Systems Inc.)
Jeff Johnson (Cisco)
David Levi (SNMP Research Inc.)

John Linn (Openvision)
Russ Mundy (Trusted Information Systems) chair
Shawn Routhier (Epilogue)
Glenn Waters (Nortel)
Bert Wijnen (IBM T.J. Watson Research)

\

Table of Contents