

## Middlebox Communications (MIDCOM) Protocol Evaluation

### Status of This Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

### Copyright Notice

Copyright (C) The Internet Society (2005).

### Abstract

This document provides an evaluation of the applicability of SNMP (Simple Network Management Protocol), RSIP (Realm Specific Internet Protocol), Megaco, Diameter, and COPS (Common Open Policy Service) as the MIDCOM (Middlebox Communications) protocol. A summary of each of the proposed protocols against the MIDCOM requirements and the MIDCOM framework is provided. Compliancy of each of the protocols against each requirement is detailed. A conclusion summarizes how each of the protocols fares in the evaluation.

### Table of Contents

Overview.....	2
Conventions Used in This Document.....	3
1. Protocol Proposals.....	3
1.1. SNMP.....	3
1.2. RSIP.....	5
1.3. Megaco.....	7
1.4. Diameter.....	8
1.5. COPS.....	10
2. Item Level Compliance Evaluation.....	11
2.1. Protocol Machinery.....	11
2.2. Protocol Semantics.....	20
2.3. General Security Requirements.....	27
3. Conclusions.....	29
4. Security Considerations.....	30
5. References.....	31
5.1. Normative References.....	31
5.2. Informative References.....	33
6. Acknowledgements.....	33

Appendix A - SNMP Overview.....	34
Appendix B - RSIP with Tunneling.....	35
Appendix C - Megaco Modeling Approach.....	37
Appendix D - Diameter IPFilter Rule.....	39
Contributors .....	42

## Overview

This document provides an evaluation of the applicability of SNMP (Simple Network Management Protocol), RSIP (Realm Specific Internet Protocol), Megaco, Diameter and COPS (Common Open Policy Service) as the MIDCOM (Middlebox Communications) protocol. This evaluation provides overviews of the protocols and general statements of applicability based upon the MIDCOM framework [2] and requirements [1] documents.

The process for the protocol evaluation was fairly straightforward as individuals volunteered to provide an individual document evaluating a specific protocol. Thus, some protocols that might be considered as reasonably applicable as the MIDCOM protocol are not evaluated in this document since there were no volunteers to champion the work. The individual protocol documents for which there were volunteers were submitted for discussion on the list with feedback being incorporated into an updated document. The updated versions of these documents formed the basis for the content of this WG document.

Section 1 contains a list of the proposed protocols submitted for the purposes of the protocol evaluation with some background information on the protocols and similarities and differences with regards to the applicability to the framework [2] provided.

Section 2 provides the item level evaluation of the proposed protocols against the Requirements [1].

Section 3 provides a summary of the evaluation. A table containing a numerical breakdown for each of the protocols, with regards to its applicability to the requirements, for the following categories is provided: Fully met, Partially met through the use of extensions, Partially met through other changes to the protocol, or Failing to be met. This summary is not meant to provide a conclusive statement of the suitability of the protocols, but rather to provide information to be considered as input into the overall protocol decision process.

In order for this document to serve as a complete evaluation of the protocols, some of the background information and more detailed aspects of the proposals documenting enhancements and applications of the protocols to comply with the MIDCOM framework and requirements are included in Appendices.

## Conventions Used in this Document

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 BCP 14, RFC 2119 [4].

## 1. Protocol Proposals

The following protocols were submitted to the MIDCOM WG for consideration:

- o SNMP
- o RSIP
- o Megaco
- o Diameter
- o COPS

The following provides an overview of each of the protocols and the applicability of each protocol to the MIDCOM framework.

### 1.1. SNMP

This section provides a general statement with regards to the applicability of SNMP as the MIDCOM protocol. A general overview and some specific details of SNMP are provided in Appendix A. This evaluation of SNMP is specific to SNMPv3, which provides the security required for MIDCOM usage. SNMPv1 and SNMPv2c would be inappropriate for MIDCOM since they have been declared Historic, and because their messages have only trivial security. Some specifics with regards to existing support for NAT and Firewall Control are provided in section 1.1.2. The differences between the SNMP framework and the MIDCOM framework are addressed in section 1.1.3.

#### 1.1.1. SNMP General Applicability

The primary advantages of SNMPv3 are that it is a mature, well understood protocol, currently deployed in various scenarios, with mature toolsets available for SNMP managers and agents.

Application intelligence is captured in MIB modules, rather than in the messaging protocol. MIB modules define a data model of the information that can be collected and configured for a managed functionality. The SNMP messaging protocol transports the data in a standardized format without needing to understand the semantics of the data being transferred. The endpoints of the communication understand the semantics of the data.

Partly due to the lack of security in SNMPv1 and SNMPv2c, and partly due to variations in configuration requirements across vendors, few MIB modules have been developed that enable standardized configuration of managed devices across vendors. Since monitoring can be done using only a least-common-denominator subset of information across vendors, many MIB modules have been developed to provide standardized monitoring of managed devices. As a result, SNMP has been used primarily for monitoring rather than for configuring network nodes.

SNMPv3 builds upon the design of widely-deployed SNMPv1 and SNMPv2c versions. Specifically, SNMPv3 shares the separation of data modeling (MIBs) from the protocol to transfer data, so all existing MIBs can be used with SNMPv3. SNMPv3 also uses the SMIV2 standard, and it shares operations and transport with SNMPv2c. The major difference between SNMPv3 and earlier versions is the addition of strong message security and controlled access to data.

SNMPv3 uses the architecture detailed in RFC 3411 [5], where all SNMP entities are capable of performing certain functions, such as the generation of requests, response to requests, the generation of asynchronous notifications, the receipt of notifications, and the proxy-forwarding of SNMP messages. SNMP is used to read and manipulate virtual databases of managed-application-specific operational parameters and statistics, which are defined in MIB modules.

#### 1.1.2. SNMP Existing Support for NAT and Firewall Control

For configuring NATs, a NAT MIB module [16] has been developed. The NAT MIB module meets all of the MIDCOM requirements concerning NAT control with the exception of grouping of policy rules (requirement 2.2.3.). In order to support this, an additional grouping table in the NAT MIB module is required.

Existing work for firewall control with SNMP only considered the monitoring of firewalls and not the configuration. Further work is required towards the development of MIBs for configuring firewalls.

#### 1.1.3. Architectural Differences between SNMP and MIDCOM

The SNMP management framework provides functions equivalent to those defined by the MIDCOM framework, although there are a few architectural differences.

Traditionally, SNMP entities have been called Manager and Agent. Manager and agent are now recognized as entities designed to support particular configurations of SNMPv3 functions. A traditional manager

is an entity capable of generating requests and receiving notifications, and a traditional agent is an entity capable of responding to requests and generating notifications. The SNMP use of the term agent is different from its use in the MIDCOM framework: The SNMP Manager corresponds to the MIDCOM agent and the SNMP Agent corresponds to the MIDCOM PDP. The SNMP evaluation assumes that the MIDCOM PDP (SNMP Agent) is physically part of the middlebox, which is allowed by the MIDCOM framework as described in section 6.0 of [2]. Thus, for the purpose of this evaluation, the SNMP agent corresponds to the Middlebox.

While this evaluation is based on the assumption that the SNMP agent corresponds to the middlebox, SNMP does not force such a restriction.

Proxy means many things to many people. SNMP can be deployed using intermediate entities to forward messages, or to help distribute policies to the middlebox, similar to the proxy capabilities of the other candidate protocols. Since proxy adds configuration and deployment complexity and is not necessary to meet the specified MIDCOM requirements, the use of a proxy agent or mid-level manager is not considered in this evaluation. Further details on SNMP proxy capabilities are provided in Appendix A.

Although the SNMP management framework does not have the concept of a session, session-like associations can be established through the use of managed objects. In order to implement the MIDCOM protocol based on SNMP, a MIDCOM MIB module is required. All requests from the MIDCOM agent to the Middlebox would be performed using write access to managed objects defined in the MIDCOM MIB module. Replies to requests are signaled by the Middlebox (SNMP agent), by modifying the managed objects. The MIDCOM agent (SNMP manager) can receive this information by reading or polling, if required, the corresponding managed object.

## 1.2. RSIP

The RSIP framework and detailed protocol are defined in RFC 3102 [17] and RFC 3103 [18] respectively.

### 1.2.1. Framework Elements in Common to MIDCOM and RSIP

The following framework elements are common to MIDCOM and RSIP listed by their MIDCOM names, with the RSIP name indicated in parenthesis:

- o Hosts
- o Applications
- o Middleboxes (RSIP gateways)
- o Private domain (private realm)

- o External domain (public realm)
- o Middlebox communication protocol (RSIP)
- o MIDCOM agent registration (host registration)
- o MIDCOM session (RSIP session)
- o MIDCOM Filter (local / remote address and port number(s) pairs)

#### 1.2.2. MIDCOM Framework Elements Not Supported by RSIP

The following MIDCOM framework elements are not supported by RSIP:

- o Policy actions and rules. RSIP always implicitly assumes a permit action. To support MIDCOM, a more general and explicit action parameter would have to be defined. RSIP requests specifying local / remote address and port number(s) pairs would have to be extended to include an action parameter, in MIDCOM rules.
- o MIDCOM agents. RSIP makes no distinction between applications and agents; address assignment operations can be performed equally by applications and agents.
- o Policy Decision Points. RSIP assumes that middleboxes grant or deny requests with reference to a policy known to them; the policy could be determined jointly by the middlebox and a policy decision point; such joint determination is not addressed by the RSIP framework, nor is it specifically precluded.

#### 1.2.3. RSIP Framework Elements Not Supported by MIDCOM

The following elements are unique to the RSIP framework. If RSIP were adopted as the basis for the MIDCOM protocol, they could be added to the MIDCOM framework:

- o RSIP client: that portion of the application (or agent) that talks to the RSIP gateway using RSIP.
- o RSIP server: that portion of an RSIP gateway that talks to applications using RSIP.
- o Realm Specific Address IP (RSA-IP) and Realm Specific Address and Port IP (RSAP-IP): RSIP distinguishes between filters that include all ports on an IP address and those that do not.
- o Demultiplexing Fields: Any set of packet header or payload fields that an RSIP gateway uses to route an incoming packet to an RSIP host. RSIP allows a gateway to perform, and an application to control, packet routing to hosts in the private domain based on more than IP header fields.

- o Host-to-middlebox tunnels: RSIP assumes that data communicated between a private realm host and a public realm host is transferred through the private realm by a tunnel between the inner host and the middle box, where it is converted to and from native IP based communications to the public realm host.

#### 1.2.4. Comparison of MIDCOM and RSIP Frameworks

RSIP with tunneling, has the advantage that the public realm IP addresses and port numbers are known to the private realm host application, thus no translation is needed for protocols such as SDP, the FTP control protocol, RTSP, SMIL, etc. However, this does require that an RSIP server and a tunneling protocol be implemented in the middlebox and an RSIP client and the tunneling protocol be implemented in the private realm host. The host modifications can generally be made without modification to the host application or requiring the implementation of a host application agent. This is viewed as a significant advantage over NAT (Network Address Translation).

Further details on the evaluation of RSIP with regards to tunneling in the context of NAT support are available in Appendix B of this document.

### 1.3. Megaco

#### 1.3.1. Megaco Architectural Model

Megaco is a master-slave, transaction-oriented protocol defined in RFC 3015 [20] in which Media Gateway Controllers (MGC) control the operation of Media Gateways (MG). Originally designed to control IP Telephony gateways, it is used between an application-unaware device (the Media Gateway) and an intelligent entity (the Media Gateway Controller) having application awareness.

The Megaco model includes the following key concepts:

1. Terminations: Logical entities on the MG that act as sources or sink of packet streams. A termination can be physical or ephemeral and is associated with a single MGC.
2. Context: An association between Terminations for sharing media between the Terminations. Terminations can be added, subtracted from a Context and can be moved from one Context to another. A Context and all of its Terminations are associated with a single MGC.

3. Virtual Media Gateways: A physical MG can be partitioned into multiple virtual MGs allowing multiple Controllers to interact with disjoint sets of Contexts/Terminations within a single physical device.
4. Transactions/Messages: Each Megaco command applies to one Termination within a Context and generates a unique response. Commands may be replicated implicitly so that they act on all Terminations of a given Context through wildcarding of Termination identifiers. Multiple commands addressed to different Contexts can be grouped in a Transaction structure. Similarly, multiple Transactions can be concatenated into a Message.
5. Descriptors/Properties: A Termination is described by a number of characterizing parameters or Properties, which are grouped in a set of Descriptors that are included in commands and responses.
6. Events and signals: A Termination can be programmed to perform certain actions or to detect certain events and notify the Agent.
7. Packages: Packages are groups of properties, events, etc. associated with a Termination. Packages are simple means of extending the protocol to serve various types of devices or Middleboxes.

#### 1.3.2. Comparison of the Megaco and MIDCOM Architectural Frameworks

In the MIDCOM architecture, the Middlebox plays the role of an application-unaware device being controlled by the application-aware Agent. In the Megaco architecture, the Media Gateway controller serves a role similar to the MIDCOM Agent (MA) and the Media Gateway serves a role similar to the Middlebox (MB). One major difference between the Megaco model and the MIDCOM protocol requirements is that MIDCOM requires that the MIDCOM Agent establish the session. Whereas, the Megaco definition is that a MG (Middlebox) establishes communication with an MGC (MIDCOM Agent).

#### 1.4. Diameter

##### 1.4.1. Diameter Architecture

Diameter is designed to support AAA for network access. It is meant to operate through networks of Diameter nodes, which both act upon and route messages toward their final destinations. Endpoints are characterized as either clients, which perform network access control, or servers, which handle authentication, authorization and accounting requests for a particular realm. Intermediate nodes perform relay, proxy, redirect, and translation services. Design



requirements for the protocol include robustness in the face of bursty message loads and server failures, resistance to specific DOS attacks and protection of message contents, and extensibility including support for vendor-specific attributes and message types.

The protocol is designed as a base protocol in RFC 3588 [24] to be supported by all implementations, plus extensions devoted to specific applications. Messages consist of a header and an aggregation of "Attribute-Value Pairs (AVPs)", each of which is a tag-length-value construct. The header includes a command code, which determines the processing of the message and what other AVP types must or may be present. AVPs are strongly typed. Some basic and compound types are provided by the base protocol specification, while others may be added by application extensions. One of the types provided in the base is the IPFilterRule, which may be sufficient to express the Policy Rules that MIDCOM deals with.

Messaging takes the form of request-answer exchanges. Some exchanges may take multiple round-trips to complete. The protocol is connection-oriented at both the transport and application levels. In addition, the protocol is tied closely to the idea of sessions, which relate sequences of message exchanges through use of a common session identifier. Each application provides its own definition of the semantics of a session. Multiple sessions may be open simultaneously.

#### 1.4.2. Comparison of Diameter With MIDCOM Architectural Requirements

The MIDCOM Agent does not perform the functions of a Diameter client, nor does the Middlebox support the functions of a Diameter server. Thus the MIDCOM application would introduce two new types of endpoints into the Diameter architecture. Moreover, the MIDCOM requirements do not at this time imply any type of intermediate node.

A general assessment might be that Diameter meets and exceeds MIDCOM architectural requirements, however the connection orientation may be too heavy for the number of relationships the Middlebox must support. Certainly the focus on extensibility, request-response messaging orientation, and treatment of the session, are all well-matched to what MIDCOM needs. At this point, MIDCOM is focused on simple point-to-point relationships, so the proxying and forwarding capabilities provided by Diameter are not needed. Most of the commands and AVPs defined in the base protocol are also surplus to MIDCOM requirements.

## 1.5. COPS

Overall, COPS, defined in RFC 2748 [25], and COPS-PR, defined in RFC 3084 [26], have similar compliancy with regards to the MIDCOM protocol requirements. In this document, references to COPS are generally applicable to both COPS and COPS-PR. However, COPS-PR is explicitly identified to meet two of the requirements. The only other major difference between COPS-PR and COPS, as applied to the MIDCOM protocol, would be the description of the MIDCOM policy rule attributes with COPS-PR MIDCOM PIB attributes rather than COPS MIDCOM client specific objects.

### 1.5.1. COPS Protocol Architecture

COPS is a simple query and response protocol that can be used to exchange policy information between a policy server (Policy Decision Point or PDP) and its clients (Policy Enforcement Points or PEPs). COPS was defined to be a simple and extensible protocol. The main characteristics of COPS include the following:

1. The protocol employs a client/server model. The PEP sends requests, updates, and deletions to the remote PDP and the PDP returns decisions back to the PEP.
2. The protocol uses TCP as its transport protocol for reliable exchange of messages between policy clients and a server.
3. The protocol is extensible in that it is designed to leverage self-identifying objects and can support diverse client specific information without requiring modification of the COPS protocol.
4. The protocol was created for the general administration, configuration, and enforcement of policies.
5. COPS provides message level security for authentication, replay protection, and message integrity. COPS can make use of existing protocols for security such as IPSEC [22] or TLS [21] to authenticate and secure the channel between the PEP and the PDP.
6. The protocol is stateful in two main aspects:
  - (1) Request/Decision state is shared and kept synchronized in a transactional manner between client and server. Requests from the client PEP are installed or remembered by the remote PDP until they are explicitly deleted by the PEP. At the same time, Decisions from the remote PDP can be generated asynchronously at any time for a currently installed request state.

- (2) State from various events (Request/Decision pairs) may be inter-associated. The server may respond to new queries differently because of previously installed, related Request/Decision state(s).

- 7. The protocol is also stateful in that it allows the server to push configuration information to the client, and then allows the server to remove such state from the client when it is no longer applicable.

#### 1.5.2. Comparison of COPS and the MIDCOM Framework

In the MIDCOM framework, the Middlebox enforces the policy controlled by an application-aware Agent. Thus, when compared to the COPS architecture, the Middlebox serves as the PEP (COPS Client) and the MIDCOM Agent serves as the PDP (COPS Policy Server). One major difference between the COPS protocol model and the MIDCOM protocol requirements is that MIDCOM requires that the MIDCOM Agent establish the session. Whereas, the COPS definition is that a PEP (Middlebox) establishes communication with a PDP (MIDCOM Agent).

### 2. Item Level Compliance Evaluation

This section contains a review of the protocol's level of compliance to each of the MIDCOM Requirements [1]. The following key will be used to identify the level of compliancy of each of the individual protocols:

T = Total Compliance. Meets the requirement fully.

P+ = Partial Compliance+. Fundamentally meets the requirement through the use of extensions (e.g., packages, additional parameters, etc).

P = Partial Compliance. Meets some aspect of the requirement, however, the necessary changes require more than an extension and/or are inconsistent with the design intent of the protocol.

F = Failed Compliance. Does not meet the requirement.

#### 2.1. Protocol Machinery

This section describes the compliancy of the proposed protocols against the protocol machinery requirements from section 2.1 of the requirements document [1]. A short description of each of the protocols is provided to substantiate the evaluation.

### 2.1.1. Ability to Establish Association Between Agent and Middlebox.

SNMP: T, RSIP: P+, Megaco: P, Diameter: T, COPS: P

SNMP: SNMPv3 provides mutual authentication at the user level (where the user can be an application or a host if desired) via shared secrets. Each authenticated principal is associated with a group that has access rights that control the principals ability to perform operations on specific subsets of data. Failure to authenticate can generate a SNMP notification (administrator configurable choice).

RSIP: RSIP allows sessions to be established between middleboxes and applications and MIDCOM agents. Authorization credentials would have to be added to the session establishment request to allow the middlebox to authorize the session requestor.

Megaco: There is a directionality component implicit in this requirement in that the MA initiates the establishment of the authorized session. Megaco defines this association to be established in the opposite direction, i.e., the Middlebox(MG) initiates the establishment. If this restriction is not considered, then Megaco makes the syntax and semantics available for the endpoint to initiate the connection.

Diameter: Although this is out of scope, the Diameter specification describes several ways to discover a peer. Having done so, a Diameter node establishes a transport connection (TCP, TLS, or SCTP) to the peer. The two peers then exchange Capability Exchange Request/Answer messages to identify each other and determine the Diameter applications each supports.

If the connection between two peers is lost, Diameter prescribes procedures whereby it may be re-established. To ensure that loss of connectivity is detected quickly, Diameter provides the Device-Watchdog Request/Answer messages, to be used when traffic between the two peers is low.

Diameter provides an extensive state machine to govern the relationship between two peers.

COPS: COPS does not meet the directionality part of the requirement. The definition of COPS allows a PEP (Middlebox) to establish communication with a PDP (MIDCOM Agent). However, nothing explicitly prohibits a PDP from establishing communication

with a PEP. The PEP could have local policies dictating what action to take when it is contacted by an unknown PDP. These actions, defined in the local policies, would ensure the proper establishment of an authorized association.

#### 2.1.2. Agent Can Relate to Multiple Middleboxes

SNMP: T, RSIP: P, Megaco: T, Diameter: T, COPS: T

SNMP: An SNMP manager can communicate simultaneously with several Middleboxes.

RSIP: RSIP sessions are identified by their IP source and destination addresses and their TCP / UDP port numbers. Thus each RSIP client can communicate with multiple servers, and each server can communicate with multiple clients. However, RSIP did not explicitly include agents in its design. The architecture and semantics of RSIP messages do not preclude agents, thus the RSIP architecture could certainly be extended to explicitly include agents; therefore RSIP is deemed partially compliant to this requirement.

Megaco: Megaco allows an MA to control several Middleboxes. Each message carries an identifier of the endpoint that transmitted the message allowing the recipient to determine the source.

Diameter: Diameter allows connection to more than one peer (and encourages this for improved reliability). Whether the Diameter connection state machine is too heavy to support the number of connections needed is a matter for discussion.

COPS: COPS PDPs are designed to communicate with several PEPs.

#### 2.1.3. Middlebox Can Relate to Multiple Agents

SNMP: T, RSIP: P, Megaco: T, Diameter: T, COPS: T

SNMP: An SNMP agent can communicate with several SNMP managers Simultaneously.

RSIP: Refer to 2.1.2.

Megaco: Megaco has the concept of Virtual Media Gateways (VMG), allowing multiple MGCs to communicate simultaneously with the same MG. Applying this model to MIDCOM would allow the same middlebox (MG) to have associations with multiple MIDCOM Agents (MGCs).

Diameter: Diameter allows connection to more than one peer and encourages this for improved reliability. Whether the Diameter connection state machine is too heavy to support the number of connections needed is a matter for discussion. The Middlebox and Agent play symmetric roles as far as Diameter peering is concerned.

COPS: The COPS-PR framework specifies that a PEP should have a unique PDP in order to achieve effective policy control. The COPS-PR protocol would allow the scenario whereby a PEP establishes communication with multiple PDPs by creating a COPS client instance per PDP.

#### 2.1.4. Deterministic Outcome When Multiple Requests are Presented to the Middlebox Simultaneously

SNMP: T, RSIP: T, Megaco: T, Diameter: T, COPS: T

SNMP: While the architectural design of SNMP can permit race conditions to occur, there are mechanisms defined as part of the SNMPv3 standard, such as view-based access control and advisory locking that can be used to prevent the conditions, and MIB modules may also contain special functionality, such as RMONs OwnerString, to prevent conflicts. Deterministic behavior of SNMP agents when being accessed by multiple managers is important for several management applications and supported by SNMP.

RSIP: All RSIP requests are defined to be atomic. Near simultaneous requests are executed as if they were sequential.

Megaco: Megaco supports the concept of VMGs to make these interactions deterministic and to avoid resource access conflicts. Each VMG has a single owner, in a MGC, and there can be no overlap between the sets of Terminations belonging to multiple VMGs. The Megaco protocol messages also include the identifier of the sending entity, so that the MG can easily determine to whom to send the response or asynchronously report certain events.

Diameter: Diameter depends partly upon the transport protocol to provide flow control when the server becomes heavily loaded. It also has application-layer messaging to indicate that it is too busy or out of space (Diameter\_TOO\_BUSY and Diameter\_OUT\_OF\_SPACE result codes).

COPS: COPS has built-in support for clear state and policy instances. This would allow the creation of well-behaved MIDCOM state machines.

### 2.1.5. Known and Stable State

SNMP: T, RSIP: T, Megaco: T, Diameter: P, COPS: T

SNMP: Requests are atomic in SNMP. MIB modules can define which data is persistent across reboots, so a known startup state can be established. The manager can poll the agent to determine the current state.

RSIP: RSIP assumes that on middlebox start-up no sessions are defined, and thus no allocations have been made. In effect, all resources are released upon restart after failure.

Megaco: Megaco has extensive audit capabilities to synchronize states between the MG and the MGC. Megaco also provides the MGC with the ability to do mass resets, as well as individual resets. The MGC can always release resources in the MG. The MG can also initiate the release of resources by the MGC.

Diameter: Diameter documentation does not discuss the degree of atomicity of message processing, so this would have to be specified in the MIDCOM extension.

COPS: The COPS protocol maintains synchronized states between Middleboxes and MA hence all the states are known on both sides.

### 2.1.6. Middlebox Status Report

SNMP: T, RSIP: T, Megaco: T, Diameter: T, COPS: T

SNMP: The status of a middlebox can be reported using asynchronous communications, or via polling.

RSIP: All RSIP client requests have explicit server responses. Additionally, a client may explicitly request server status using a QUERY request.

Megaco: Megaco has extensive audit capabilities for the MG to report status information to the MGC. It can also report some status updates using the ServiceChange command.

Diameter: Diameter provides a number of response codes by means of which a server can indicate error conditions reflecting status of the server as a whole. The Disconnect-Peer-Request provides a means in the extreme case to terminate a connection with a peer gracefully, informing the other end about the reason for the disconnection.

COPS: The COPS Report message is designed to indicate any asynchronous conditions/events.

#### 2.1.7. Middlebox Can Generate Unsolicited Messages

SNMP: T, RSIP: T, Megaco: T, Diameter: T, COPS: T

SNMP: SNMPv3 supports both confirmed and unconfirmed asynchronous notifications.

RSIP: An RSIP server will send an unsolicited DE\_REGISTER\_RESPONSE to force an RSIP host to relinquish all of its bindings and terminate its relationship with the RSIP gateway. An RSIP server can send an asynchronous ERROR\_RESPONSE to indicate less severe conditions.

Megaco: Megaco supports the asynchronous notification of events using the Notify command.

Diameter: The Diameter protocol permits either peer in a connection to originate transactions. Thus the protocol supports Middlebox-originated messages.

COPS: The COPS Report message is designed to indicate any asynchronous conditions/events.

#### 2.1.8. Mutual Authentication

SNMP: T, RSIP: T, Megaco: T, Diameter: T, COPS: T

SNMP: SNMPv3 meets this requirement. SNMPv3 supports user authentication and explicitly supports symmetric secret key encryption between MIDCOM agent (SNMP manager) and Middlebox (SNMP agent), thus supporting mutual authentication. The default authentication and encryption methods are specified in RFC 3414 [11] (MD5, SHA-1, and DES). Different users at the same management application (MIDCOM agent) can authenticate themselves with different authentication and encryption methods, and additional methods can be added to SNMPv3 entities as needed.

RSIP: This requirement can be met by operating RSIP over IPsec as described in RFC 3104 [19]. The RSIP framework recommends all communication between an RSIP host and gateway be authenticated. Authentication, in the form of a message hash appended to the end of each RSIP protocol packet, can serve to authenticate the RSIP host and gateway to one another, provide message integrity, and



avoid replay attacks with an anti-replay counter. However, the message hash and replay counter parameters would need to be defined for the RSIP protocol.

Megaco: Megaco provides for the use of IPSec [22] for all security mechanisms including mutual authentication, integrity check and encryption. Use of IKE is recommended with support of RSA signatures and public key encryption.

Diameter: The Diameter base protocol assumes that messages are secured by using either IPSec or TLS [21]. Diameter requires that when using the latter, peers must mutually authenticate themselves.

COPS: COPS has built-in message level security for authentication, replay protection, and message integrity. COPS can also use TLS or IPSec.

#### 2.1.9. Termination of session by either party

SNMP: T, RSIP: T, Megaco: T, Diameter: T, COPS: T

SNMP: Each SNMPv3 message is authenticated and authorized, so each message could be considered to have its own session, which automatically terminates after processing. Processing may be stopped for a number of reasons, such as security, and a response is sent.

Either peer may stop operating, and be unavailable for further operations. The authentication and/or authorization parameters of a principal may be changed between operations if desired, to prevent further authentication or authorization for security reasons.

Additionally, managed objects can be defined for realizing sessions that persist beyond processing of a single message. The MIB module would need to specify the responsibility for cleanup of the objects following normal/abnormal termination.

RSIP: An RSIP client may terminate a session with a DE\_REGISTER\_REQUEST. An RSIP server may terminate a session with an unsolicited DE\_REGISTER\_RESPONSE, and then respond to subsequent requests on the session with a REGISTER\_FIRST error.

Megaco: The Megaco protocol allows both peers to terminate the association with proper reason code.

Diameter: Either peer in a connection may issue a Disconnect-Peer-Request to end the connection gracefully.

COPS: COPS allows both the PEP and PDP to terminate a session.

#### 2.1.10. Indication of Success or Failure

SNMP: T, RSIP: T, Megaco: T, Diameter: T, COPS: T

SNMP: Each operation request has a corresponding response message that contains an error status to indicate success or failure. For complex requests that the middlebox cannot complete immediately, the corresponding MIB module may be designed to also provide asynchronous notifications of the success or failure of the complete transaction, and/or may provide pollable objects that indicate the success or failure of the complete transaction. For example, see ifAdminStatus and ifOperStatus in RFC 2863 [28].

RSIP: All RSIP requests result in a paired RSIP response if the request was successful or an ERROR\_RESPONSE if the request was not successful.

Megaco: Megaco defines a special descriptor called an Error descriptor that contains the error code and an optional explanatory string.

Diameter: Every Diameter request is matched by a response, and this response contains a result code as well as other information.

COPS: The COPS Report message directly fulfills this requirement.

#### 2.1.11. Version Interworking

SNMP: T, RSIP: T, Megaco: T, Diameter: T, COPS: T

SNMP: SNMP has a separation of the protocol to carry data, and the data that defines additional management functionality. Additional functionality can be added easily through MIBs. Capability exchange in SNMP is usually uni-directional. Managers can query the middlebox (SNMP agent) to determine which MIBs are supported. In addition, multiple message versions can be supported simultaneously, and are identified by a version number in the message header.

RSIP: Each RSIP message contains a version parameter.

Megaco: Version interworking and negotiation are supported both for the protocol and any extension Packages.

Diameter: The Capabilities Exchange Request/Answer allows two peers to determine information about what each supports, including protocol version and specific applications.

COPS: The COPS protocol can carry a MIDCOM version number and capability negotiation between the COPS client and the COPS server. This capability negotiation mechanism allows the COPS client and server to communicate the supported features/capabilities. This would allow seamless version interworking.

#### 2.1.12. Deterministic Behaviour in the Presence of Overlapping Rules

SNMP: T, RSIP: T, Megaco: P, Diameter: T, COPS: T

SNMP: Rulesets would be defined in MIBs. The priority of rulesets, and the resolution of conflict, can be defined in the MIB module definition. The SNMPConf policy MIB defines mechanisms to achieve deterministic behavior in the presence of overlapping rule sets.

RSIP: All requests for allocation of IP addresses, or ports or both resulting in rule overlap are rejected by an RSIP server with a LOCAL\_ADDR\_INUSE error.

Megaco: This is met with the help of a model that separates Megaco protocol elements from the overlapping Policy rules (see Appendix C). However, new behavior for the Megaco protocol elements needs to be specified as part of a new MIDCOM specific Package.

Diameter: The IPFilterRule type specification, which would probably be used as the type of a Policy Rule AVP, comes with an extensive semantic description providing a deterministic outcome, which the individual Agent cannot know unless it knows all of the Policy Rules installed on the Middlebox. Rules for the appropriate direction are evaluated in order, with the first matched rule terminating the evaluation. Each packet is evaluated once. If no rule matches, the packet is dropped if the last rule evaluated was a permit, and passed if the last rule was a deny. The IPFilterRule format and further details on its applicability to this requirement are provided in Appendix D.

COPS: The COPS protocol provides transactional-based communication between the PEP and PDP, hence the behavior is totally deterministic provided the middlebox state machine is designed correctly. The COPS protocol features encourage and support good state machine design.

## 2.2. Protocol Semantics

This section contains the individual protocols as evaluated against the protocol semantic requirements from section 2.2 of the requirements document [1]. A short description of each of the protocols is provided to substantiate the evaluation.

### 2.2.1. Extensibility

SNMP: T, RSIP: T, Megaco: T, Diameter: T, COPS: T

SNMP: Extensibility is a basic feature of the SNMP management Framework.

RSIP: All RSIP messages consist of three mandatory fields (protocol version, message type, and message length) and a sequence of parameterType / length / value 3-tuples. New messages may be defined by defining new values for the message type field. New parameter types may be defined, and existing messages may be extended, by defining new parameterType values. If new messages, parameters, or both are added in a non-backward compatible way, a new value of the protocol version field may be defined. This may be desirable even if the additions are backward compatible.

Megaco: Megaco is easily extensible through new Packages, which allow definition of new attributes and behavior of a Termination.

Diameter: Diameter provides a great deal of flexibility for extensions, including allowance for vendor-defined commands and AVPs and the ability to flag each AVP as must-understand or ignorable if not understood.

COPS: The COPS protocol is extensible, since it was designed to separate the Protocol from the Policy Control Information.

### 2.2.2. Support of Multiple Middlebox Types

SNMP: T, RSIP: P+, Megaco: T, Diameter: P+, COPS: T

SNMP: SNMP explicitly supports managing different device types with different capabilities. First the managed object called sysObjectID from basic MIB-II [3] identifies the type of box. For boxes with variable capabilities, SNMP can check the availability of corresponding MIBs.

RSIP: All types of middleboxes are supported so long as the ruleset action is permit. Other actions would require the definition of a new RSIP message parameter with values for permit and the other desired actions.

Megaco: Megaco can support multiple Middlebox types on the same interface either by designing the properties representing the Policy Rules to provide this support, or by using multiple terminations in the same session, each representing one type of action. In the latter case, the Megaco Context can be used as a convenient means of managing the related terminations as a group. However, the inherent idea of flow between terminations of a context is irrelevant and would have to be discarded.

Diameter: Any necessary additional AVPs or values must be specified as part of the MIDCOM application extension (see <2.2.8> below).

COPS: COPS allows a PDP to provide filters and actions to multiple PEP functions through a single COPS session.

### 2.2.3. Ruleset Groups

SNMP: T, RSIP: P+, Megaco: T, Diameter: T, COPS: T

SNMP: This requirement can be realized via the SNMP management framework by an appropriate definition of a MIB module. The SNMPConf WG has already defined an SNMP Policy MIB that permits the definitions of policy rulesets and grouping of rulesets.

RSIP: RSIP currently only allows one IP address, or address and port range, to be assigned to a bind-ID. RSIP could implement rulesets as required by adding an optional bind-ID parameter to the ASSIGN\_REQUESTs to extend an existing ruleset rather than creating a new one. Similarly, the FREE\_REQUESTs would have to be extended by adding optional, local and remote, address and port parameters.

Megaco: The Megaco context can be used to group terminations to be managed together. For example, all of the terminations, each representing an instantiation of a Policy Rule, can be deleted in one command by doing a wildcarded Subtract from the context. However, the inherent idea of media flows between terminations of a context would be irrelevant in this application of the protocol.

Diameter: Diameter allows message syntax definitions where multiple instances of the same AVP (for example, a Policy Rule AVP whose syntax and low-level semantics are defined by the IPFilterRule type definition) may be present. If a tighter grouping is

required, the set of Diameter base types includes the Grouped type. MIDCOM can choose how to make use of these capabilities to meet the ruleset group requirement when defining its application extension to the Diameter protocol.

COPS: The COPS-PR Handle State may be used to associate the set of closely related policy objects. As the Middlebox learns additional requirements, the Middlebox adds these resource requirements under the same handle ID, which constitutes the required aggregation.

#### 2.2.4. Lifetime Extension

SNMP: P+, RSIP: T, Megaco: T, Diameter: T, COPS: P+

SNMP: This requirement can be realized via the SNMP management framework by an appropriate definition of a MIB module. The SNMPConf WG has developed a Policy MIB module that includes a pmPolicySchedule object with a modifiable lifetime.

RSIP: A client may request an explicit lease time when a request is made to assign one or more IP addresses, ports or both. The server may grant the requested lease time, or assign one if none was requested. Subsequently, the lease time may be extended if a client's EXTEND\_REQUEST is granted by the server.

Megaco: The MG can report the imminent expiry of a policy rule to the MGC, which can then extend or delete the corresponding Termination.

Diameter: The Diameter concept of a session includes the session lifetime, grace period, and lifetime extension. It may make sense to associate the Diameter session with the lifetime of a MIDCOM Policy Rule, in which case support for lifetime extension comes ready-made.

COPS: COPS allows a PDP to send unsolicited decisions to the PEP. However, the unsolicited events will be relevant to the COPS MIDCOM specific client or the MIDCOM specific PIB which needs to be defined. This would allow the PDP to extend the lifetime of an existing ruleset.

### 2.2.5. Handling of Mandatory/Optional Nature of Unknown Attributes

SNMP: T, RSIP: T, Megaco: P+, Diameter: P+, COPS: T

SNMP: Unknown attributes in a read operation are flagged as exceptions in the Response message, but the rest of the read succeeds. In a write operation (a SET request), all attributes are validated before the write is performed. If there are unknown attributes, the request fails and no writes are done. Unknown attributes are flagged as exceptions in the Response message, and the error status is reported.

RSIP: All options of all requests are fully specified. Not understood parameters must be reported by an ERROR\_RESPONSE with an EXTRA\_PARM error value, with the entire request otherwise ignored.

Megaco: Megaco entities provide Error codes in response messages. If a command marked "Optional" in a transaction fails, the remaining commands will continue. However, the specified requirement deals with rules of processing properties that need definition in new Package.

Diameter: Indication of the mandatory or optional status of AVPs is fully supported, provided it is enabled in the AVP definition. No guidance is imposed regarding the return of diagnostic information for optional AVPs.

COPS: COPS provides for the exchange of capabilities and limitations between the PEP and PDP to ensure well-known outcomes are understood for scenarios with unknown attributes. There is also clear error handling for situations when the request is rejected.

### 2.2.6. Actionable Failure Reasons

SNMP: T, RSIP: P+, Megaco: T, Diameter: T, COPS: T

SNMP: The SNMPv3 protocol returns error codes and exception codes in Response messages, to permit the requestor to modify their request. Errors and exceptions indicate the attribute that caused the error, and an error code identifies the nature of the error encountered.

If desired, a MIB can be designed to provide additional data about error conditions either via asynchronous notifications or polled objects.

RSIP: RSIP defines a fairly large number of very specific error values. It is anticipated that additional error values will also have to be defined along with the new messages and parameters required for MIDCOM.

Megaco: The MG can provide Error codes in response messages allowing the MGC to modify its behavior. Megaco uses transaction identifiers for correlation between a response and a command. If the same transaction id is received more than once, the receiving entity silently discards the message, thus providing some protection against replay attacks.

Diameter: Diameter provides an extensive set of failure reasons in the base protocol.

COPS: COPS uses an error object to identify a particular COPS protocol error. The error sub-code field may contain additional detailed COPS client (MIDCOM Middlebox) specific error codes.

#### 2.2.7. Multiple Agents Operating on the Same Ruleset.

SNMP: T, RSIP: P, Megaco: P, Diameter: T, COPS: P

SNMP: The SNMP framework supports multiple managers working on the same managed objects. The View-based Access Control Model (VACM, RFC 3415 [14]) even offers means to customize the access rights of different managers in a fine-grained way.

RSIP: RSIP neither explicitly permits nor precludes an operation on a binding by a host that had not originally create the binding. However, to support this requirement, the RSIP semantics must be extended to explicitly permit any authorized host to request operations on a binding; this does not require a change to the protocol.

Megaco: If the Megaco state machine on the Middle Box is decoupled from the Middle Box policy rule management, this requirement can be met with local policies on the Middle Box. However, this violates the spirit of the Megaco protocol, thus Megaco is considered partially compliant to this requirement.

Diameter: The Diameter protocol, as currently defined, would allow multiple agents to operate on the same ruleset.

COPS: It is possible to use COPS to operate the same resource with multiple agents. An underlying resource management function, separate from the COPS state machine, on the Middlebox will handle the arbitration when resource conflicts happen.



### 2.2.8. Transport of Filtering Rules

SNMP: P+, RSIP: P+, Megaco: P+, Diameter: P+, COPS: P+

SNMP: This requirement can be met by an appropriate definition of a MIDCOM MIB module. SMI, the language used for defining MIB modules, is flexible enough to allow the implementation of a MIB module to meet the semantics of this requirement.

RSIP: To support this requirement, a new optional enumeration parameter, transportProtocol, can be added to the RSIP ASSIGN\_REQUESTs. When the parameter is included, the binding created applies only to the use of the bound addresses and ports, by the specific transportProtocol. When the parameter is not included, the binding applies to the use of all the bound addresses and ports, by any transport protocol, thus maintaining backward compatibility with the current definition of RSIP.

Megaco: Megaco protocol can meet this requirement by defining a new property for the transport of filtering rules.

Diameter: While Diameter defines the promising IPFilterRule data type (see 2.1.12 above), there is no existing message, which would convey this to a Middlebox along with other required MIDCOM attributes. A new MIDCOM application extension of Diameter would have to be defined.

COPS: The COPS protocol can meet this requirement by using a COPS MIDCOM specific client or a MIDCOM specific PIB.

### 2.2.9. Mapped Port Parity

SNMP: P+, RSIP: P+, Megaco: P+, Diameter: P+, COPS: P+

SNMP: This requirement can be met by an appropriate definition of a MIDCOM MIB module.

RSIP: To support this requirement, a new optional boolean parameter, portOddity, can be added to the RSIP ASSIGN\_REQUESTs. If the parameter is TRUE, the remote port number of the binding created would have the same oddity as the local port. If the parameter is not specified, or is FALSE, the remote port's oddity is independent of the local port's oddity, thus maintaining backward compatibility with the current definition of RSIP.

Megaco: Megaco can be easily extended using a MIDCOM specific Package to support this feature.

Diameter: This capability is not part of the current IPFilterRule type definition. Rather than modify the IPFilterRule type, MIDCOM could group it with other AVPs which add the missing information.

COPS: The COPS protocol has all the flexibility to meet this requirement by using a COPS MIDCOM specific client or a MIDCOM specific PIB.

#### 2.2.10. Consecutive Range of Port Numbers

SNMP: P+, RSIP: T, Megaco: P+, Diameter: P+, COPS: P+

SNMP: This requirement can be met by an appropriate definition of a MIDCOM MIB module. SMI, the language used for defining MIB modules, is flexible enough to allow the implementation of a MIB module to meet the semantics of this requirement.

RSIP: The ports parameter of the RSIP ASSIGN\_REQUESTs specifically allows multiple, consecutive port numbers to be specified.

Megaco: Megaco can be easily extended using a MIDCOM specific Package to support this feature.

Diameter: This capability is not part of the current IPFilterRule type definition. Rather than modify the IPFilterRule type, MIDCOM could group it with other AVPs which add the missing information.

COPS: The COPS protocol has all the flexibility to meet this requirement by using a COPS MIDCOM specific client or a MIDCOM specific PIB.

#### 2.2.11. More Precise Rulesets Contradicting Overlapping Rulesets

SNMP: P+, RSIP: P+, Megaco: P+, Diameter: T, COPS: P+

SNMP: This requirement can be met by an appropriate definition of a MIDCOM MIB module.

RSIP: To support this requirement, a new optional boolean parameter, overlapOK, can be added to the RSIP ASSIGN\_REQUESTs. If the parameter is TRUE, the binding may overlap with an existing binding. If the parameter is unspecified, or is FALSE, the binding will not overlap with an existing binding, thus maintaining backward compatibility with the current definition of RSIP.

Megaco: This requirement would be met if the policy in the Middlebox allows contradictory, overlapping policy rules to be installed.

Diameter: Allowed by the IPFilterRule semantics described in Appendix D.

COPS: The COPS protocol has all the flexibility to meet this requirement by using a COPS MIDCOM specific client or a MIDCOM specific PIB.

## 2.3. General Security Requirements

This section contains the individual protocols as evaluated against the General Security requirements from section 2.3 of the requirements document [1]. A short description of each of the protocols is provided to substantiate the evaluation.

### 2.3.1. Message Authentication, Confidentiality and Integrity

SNMP: T, RSIP: T, Megaco: T, Diameter: T, COPS: T

SNMP: SNMPv3 includes the User-based Security Model (USM, RFC 3414 [11]), which defines three standardized methods for providing authentication, confidentiality, and integrity. Additionally, USM has specific built-in mechanisms for preventing replay attacks including unique protocol engine IDs, timers and counters per engine and time windows for the validity of messages.

RSIP: This requirement can be met by operating RSIP over IPsec. The RSIP framework recommends all communication between an RSIP host and gateway be authenticated. Authentication, in the form of a message hash appended to the end of each RSIP protocol packet, can serve to authenticate the RSIP host and gateway to one another, provide message integrity, and avoid replay attacks with an anti-replay counter. However, the message hash and replay counter parameters would need to be defined for the RSIP protocol.

Megaco: Megaco provides for these functions with the combined usage of IPSEC [22] or TLS [21].

Diameter: Diameter relies on either IPSEC or TLS for these functions.

COPS: COPS has built-in message level security for authentication, replay protection, and message integrity. COPS can also use TLS or IPsec, thus reusing existing security mechanisms that have interoperated in the markets.

### 2.3.2. Optional Confidentiality Protection

SNMP: T, RSIP: T, Megaco: T, Diameter: T, COPS: T

SNMP: SNMPv3 includes the User-based Security Model, which defines three standardized methods for providing authentication, confidentiality, and integrity, and is open to add further methods. The method to use can be optionally chosen.

RSIP: Refer to 2.3.1.

Megaco: Refer to 2.3.1

Diameter: Implementation support of IPSEC ESP (RFC 2406 [23]) in Diameter applications is not optional. Deployment of either IPSEC or TLS is optional.

COPS: Refer to 2.3.1.

### 2.3.3. Operate Across Untrusted Domains

SNMP: T, RSIP: T, Megaco: T, Diameter: T, COPS: T

SNMP: The User-based Security Model of SNMPv3 defines three standardized methods for providing authentication, confidentiality, and integrity, and it is open to add further methods. These methods operate securely across untrusted domains.

RSIP: Refer to 2.3.1.

Megaco: Refer to 2.3.1.

Diameter: The Diameter specification [24] recommends the use of TLS [21] across untrusted domains.

COPS: Refer to 2.3.1

### 2.3.4. Mitigates Replay Attacks on Control Messages

SNMP: T, RSIP: T, Megaco: T, Diameter: T, COPS: T

SNMP: The User-based Security Model for SNMPv3 has specific built-in mechanisms for preventing replay attacks including unique protocol engine IDs, timers and counters per engine and time windows for the validity of messages.

RSIP: Refer to 2.3.1

**Megaco:** Megaco commands and responses include matching transaction identifiers. The recipient receiving the same transaction id multiple times would discard the message, thus providing some protection against replay attacks. If even stronger protection against replay attack is needed, Megaco provides for the use of IPsec or TLS.

**Diameter:** Diameter requires that implementations support the replay protection mechanisms of IPSEC.

**COPS:** Refer to 2.3.1

### 3. Conclusions

The overall statistics with regards to the number of Fully Compliant, Partially Compliant (P+ and P) and Failing Compliancy requirements for each of the protocols is summarized in table 1.

	T	P+	P	F
-----	-----	-----	-----	-----
SNMP	22	5	0	0
RSIP	17	7	3	0
Megaco	19	5	3	0
Diameter	21	5	1	0
COPS	20	5	2	0

Table 1: Totals across all Requirements

In considering the P+ category of compliancy, an important aspect is the mechanism for support of extensibility. The extension mechanism provided by SNMP and COPS-PR using MIBs and PIBs respectively, provides extensions with no impact to the protocol. Diameter extensions require protocol changes, thus has a higher impact, although the extensions can be handled by other Diameter entities without being understood. Megaco's extension mechanisms of packages also requires protocol changes that must be understood by both sending and receiving entities, also being considered higher impact. The RSIP extension mechanism has the largest impact on the existing protocol and is based upon defining the necessary new parameters.

The SNMP management framework meets all the specified MIDCOM protocol requirements with the appropriate design of a MIDCOM MIB module. SNMP is a proven technology with stable and proven development tools, already has extensions defined to support NAT configuration and policy-based management. SNMPv3 is a full standard, is more mature and has undergone more validation than the other protocols in

the evaluation, and has been deployed to manage large-scale real-world networks (e.g., DOCSIS cable modem networks). The applicability of SNMP to the MIDCOM framework has a restriction in that it assumes the MIDCOM PDP is part of the Middlebox.

RSIP fully meets many of the MIDCOM requirements. However, it does require additions and extensions to meet several of the requirements. RSIP would also require several framework elements to be added to the MIDCOM framework as identified in section 1.2.3. In addition, the tunneling required for RSIP as described in section 1.2.4, results in RSIP not being acceptable by the WG as the MIDCOM protocol.

Megaco fully meets most of the key requirements for the MIDCOM Protocol. Additional extensions in the form of a new Termination / Package definition would be required for MIDCOM to meet several of the requirements. In order to meet the remaining requirements, modeling the underlying Middlebox resources (e.g., filters, policy rules) as separate elements from the Megaco entities might allow the usage of the protocol as-is, satisfying some of the resource access control requirements.

The Diameter evaluation indicated a good overall fit. Some partially met requirements were identified that could be addressed by a new application extension. However, the Diameter architecture may be too heavy for the MIDCOM application and clearly much of the Diameter base is not needed. In addition, Diameter is the only protocol, at the time of this evaluation, for which the RFCs had not yet been published. Other than these reservations, the protocol is a good fit to MIDCOM requirements.

The COPS evaluation indicates that the protocol meets the majority of the MIDCOM protocol requirements by using the protocol's native extension techniques, with COPS-PR being explicitly required to meet requirements 2.1.3 and 2.2.3. In order to fully satisfy one partially met requirement, 2.1.1, the COPS model would need to allow a PDP to establish communication with a PEP. While not explicitly prohibited by the COPS model, this would require additions, in the form of local policy, to ensure the proper establishment of an authorized association.

#### 4. Security Considerations

Security considerations for the MIDCOM protocol are covered by the comparison against the specific Security requirements in the MIDCOM requirements document [1] and are specifically addressed by section 2.1.8 and section 2.3.

## 5. References

### 5.1. Normative References

- [1] Swale, R., Mart, P., Sijben, P., Brim, S., and M. Shore, "Middlebox Communications (MIDCOM) Protocol Requirements", RFC 3304, August 2002.
- [2] Srisuresh, P., Kuthan, J., Rosenberg, J., Molitor, A., and A. Rayhan, "Middlebox Communications Architecture and Framework", RFC 3303, August 2002.
- [3] Rose, M. and K. McCloghrie, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", STD 17, RFC 1213, March 1991.
- [4] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [5] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing SNMP Management Frameworks", STD 62, RFC 3411, December 2002.
- [6] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.
- [7] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.
- [8] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.
- [9] Presuhn, R. (Ed.), "Transport Mappings for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3417, December 2002.
- [10] Case, J., Harrington D., Presuhn R., and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3412, December 2002.
- [11] 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.
- [12] Presuhn, R. (Ed.), "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3416, December 2002.

- [13] Levi, D., Meyer, P., and B. Stewart, "SNMPv3 Applications", STD 62, RFC 3413, December 2002.
- [14] 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.
- [15] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction to Version 3 of the Internet-Standard Network Management Framework", RFC 3410, December 2002.
- [16] Rohit, R., Srisuresh, P., Raghunarayan, R., Pai, N., and C. Wang, "Definitions of Managed Objects for Network Address Translators (NAT)", RFC 4008, March 2005.
- [17] Borella, M., Lo, J., Grabelsky, D., and G. Montenegro, "Realm Specific IP: Framework", RFC 3102, October 2001.
- [18] Borella, M., Grabelsky, D., Lo, J., and K. Taniguchi, "Realm Specific IP: Protocol Specification", RFC 3103, October 2001.
- [19] Montenegro, G. and M. Borella, "RSIP Support for End-to-end Isec", RFC 3104, October 2001.
- [20] Cuervo, F., Greene, N., Rayhan, A., Huitema, C., Rosen, B., and J. Segers, "Megaco Protocol Version 1.0", RFC 3015, October 2001.
- [21] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.
- [22] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.
- [23] Kent, S. and R. Atkinson, "IP Encapsulating Security Payload", RFC 2406, November 1998.
- [24] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, September 2003.
- [25] Durham, D. (Ed.), Boyle, J., Cohen, R., Herzog, S., Rajan, R., and A. Sastry, "The COPS (Common Open Policy Service) Protocol", RFC 2748, January 2000.
- [26] Chan, K., Seligson, J., Durham, D., Gai, S., McCloghrie, K., Herzog, S., Reichmeyer, F., Yavatkar, R., and A. Smith, "COPS Usage for Policy Provisioning", RFC 3084, March 2001.



## 5.2. Informative References

- [27] Raz, D., Schoenwalder, J., and B. Sugla, "An SNMP Application Level Gateway for Payload Address Translation", RFC 2962, October 2000.
- [28] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.

## 6. Acknowledgements

The editor would like to acknowledge the constructive feedback provided by Joel M. Halpern on the individual protocol evaluation contributions. In addition, a thanks to Elwyn Davies, Christopher Martin, Bob Penfield, Scott Brim and Martin Stiemerling for contributing to the mailing list discussion on the document content.

## Appendix A - SNMP Overview

The SNMP Management Framework presently consists of five major components:

- o An overall architecture, described in RFC 3411 [5]. A more detailed introduction and applicability statements for the SNMP Management Framework can be found in RFC 3410 [15].
- o Mechanisms for describing and naming objects and events for the purpose of management. The current version of this Structure of Management Information (SMI) is called SMIV2 and described in RFC 2578 [6], RFC 2579 [7] and RFC 2580 [8].
- o Message protocols for transferring management information. The current version of the message protocol is called SNMPv3 and described in RFC 3412 [10], RFC 3414 [11] and RFC 3417 [9].
- o Protocol operations for accessing management information. The current version of the protocol operations and associated PDU formats is described in RFC 3416 [12].
- o A set of fundamental applications described in RFC 3413 [13] and the view-based access control mechanism described in RFC 3415 [14].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the mechanisms defined in the SMI.

### A.1 SNMPv3 Proxy Forwarding

SNMPv3 proxy forwarding (RFC 3413 [13]) provides a standardized mechanism to configure an intermediate node to forward SNMP messages. A command generating entity sends requests to a proxy forwarding entity that forwards the request to a third entity.

One SNMP entity may serve both functions as the SNMP agent to monitor and configure the node on which it is resident, and as an intermediate node in a proxy relationship to permit monitoring and configuration of additional entities.

Each entity is identified by a unique engineID value, specifically to support proxy between addressing domains and/or trust domains. An SNMPv3 message contains two engineIDs- one to identify the database to be used for message security, and one to identify the source (or target) of the contained data. Message security is applied between the originator and the proxy, and then between the proxy and the

end-target. The PDU contains the engineID of the node whose data is contained in the message, which passes end-to-end, unchanged by the proxy.

SNMPv3 proxy was designed to provide a standard SNMP approach to inserting an intermediate node in the middle of communications for a variety of scenarios. SNMPv3 proxy can support crossing addressing domains, such as IPv4 and IPv6, crossing SNMP version domains, such as SNMPv3 and SNMPv1, crossing security mechanism domains, such as DES and AES, and for providing a single point of management contact for a subset of the network, such as managing a private network through a NAT device or a VPN endpoint.

## A.2 Proxies Versus Application Level Gateways

Proxies are generally preferred to Application Level Gateways for SNMP. ALGs typically modify the headers and content of messages. SNMP is a protocol designed for troubleshooting network (mis-) configurations. Because an operator needs to understand the actual configuration, the translation of addresses within SNMP data causes confusion, hiding the actual configuration of a managed device from the operator. ALGs also introduce security vulnerabilities, and other complexities related to modifying SNMP data.

SNMP Proxies can modify message headers without modifying the contained data. This avoids the issues associated with translating the payload data, while permitting application level translation of addresses.

The issues of ALGs versus proxies for SNMP Payload Address Translation are discussed at length in RFC 2962 [27].

## Appendix B - RSIP with Tunneling

NAT requires ALGs (Application Layer Gateways) in middleboxes without MIDCOM, and application modifications or agents for middleboxes with MIDCOM.

Support for NAT without tunneling could easily be added to the RSIP control protocol. NAT would be defined as a new, null tunnel type. Support for the NAT null tunnels could be implemented in hosts, or in applications or application agents.

If support for NAT null tunnels were implemented in hosts, no modifications to applications would be required, and no application agents or ALGs would be required. This has obvious advantages. In addition to the NAT null tunnel, the host would have to implement an RSIP / MIDCOM client (or a STUN client) and the middlebox would have

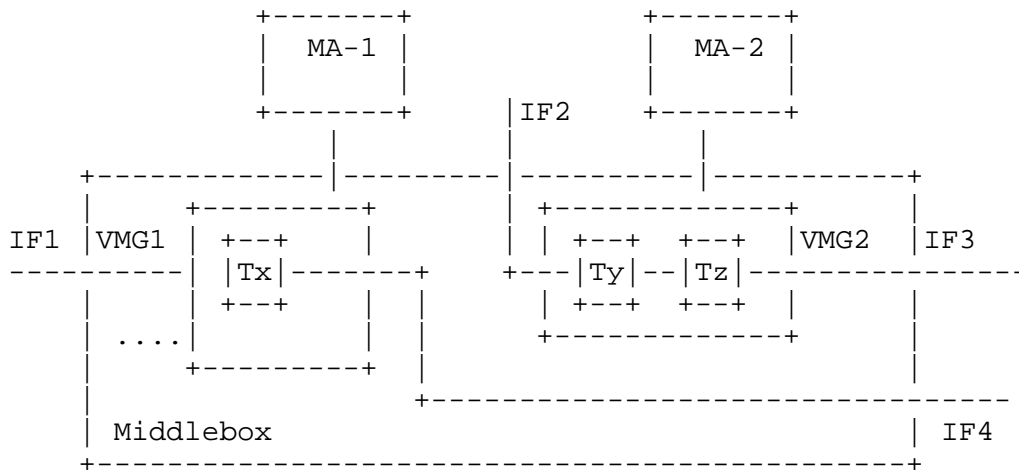
to implement an RSIP / MIDCOM server, or a STUN server would have to be available beyond the middlebox. Note that the STUN client / server approach may not work with all types of middleboxes.

If support for NAT null tunnels were NOT implemented in hosts, then applications would have to be modified, or application agents or ALGs would have to be implemented. This has the advantage over tunnels (whether null or not) of not requiring modification to hosts, but would require the modification of host applications or the implementation of application agents, both of which would include an RSIP / MIDCOM client, and the implementation of an RSIP/MIDCOM server in the middlebox. Again, in some situations, STUN could be used instead of RSIP / MIDCOM.

Tunneled or not, an RSIP / MIDCOM server is needed in the middlebox. Tunneled, the host needs to be modified, but not the application. Untunneled, an agent must be added or the application must be modified, but there would be no host modifications. The advantages/disadvantages of tunneling would need to be evaluated in considering RSIP.

## Appendix C - Megaco Modeling Approach

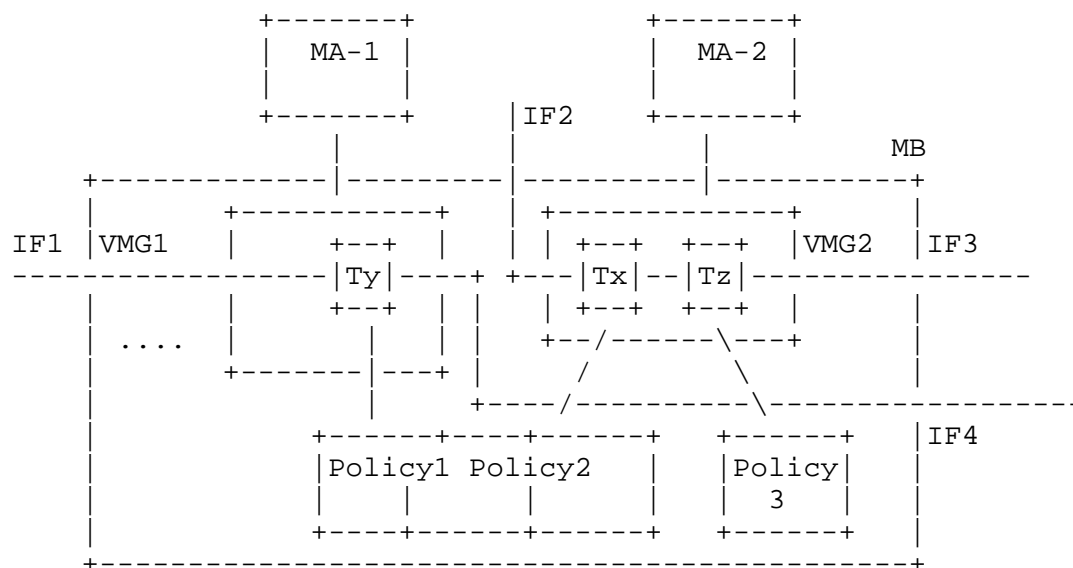
To model the Middlebox functions such as firewall, NAT etc., a new Middlebox Termination type needs to be defined within Megaco. If policy-rule overlap or modification by multiple Agents is NOT required, then a policy rule is equivalent to a Termination (see Figure 1). The various components of a Policy rule such as filter, action, life-time, creator etc. are described as various properties of a Termination. Use of the Virtual Media Gateway (VMG) concept allows for conflict-free interaction of multiple MA's with the same MB.



Tx: Termination x = Policy rule x  
 Ty: Termination y = Policy rule y  
 Tz: Termination z = Policy rule z  
 MA: MIDCOM Agent  
 IF: Interface

Figure 1.

If it is required to allow multiple agents manipulate the same Middlebox resource (e.g., a Policy rule or a filter), the latter needs to be kept separate from the Termination (the Policy rule is manipulated by the MA by manipulating the properties of the associated Termination). For example, if overlapping policy rule manipulation is required, then a Termination shall be associated with a single policy rule, but a policy rule may be associated with more than one Termination. Thus, a Termination can share a policy rule with another Termination, or have a policy rule partially overlapping with that of another Termination. This model allows two MAs, controlling two distinct Terminations (see Figure 2), manipulate the same or overlapping policy rules. In Figure 2, policy rules 1 and 2 are overlapping and they are shared by MA-1 and MA-2.



Tx: Termination x  
 Ty: Termination y  
 Tz: Termination z  
 MA: MIDCOM Agent  
 IF: Interface  
 MB: Middlebox

Figure 2.

This requires that the Agent and the Middlebox adhere to the following principles:

- (1) Only one Termination has read/write access to a filter at any time.

- (2) When the policy rule is being modified by a new agent (i.e., not the one that created the policy) the Middlebox makes a policy decision and decides whether to accept the requested modification or not. In the case the modification is accepted the initial MIDCOM agent may be notified.

#### Appendix D - Diameter IPFilter Rule

The IPFilterRule format is derived from the OctetString AVP Base Format. It uses the UTF-8 encoding and has the same requirements as the UTF8String. Packets may be filtered based on the following information that is associated with it:

Direction	(in or out)
Source and destination IP address	(possibly masked)
Protocol	
Source and destination port	(lists or ranges)
TCP flags	
IP fragment flag	
IP options	
ICMP types	

Rules for the appropriate direction are evaluated in order, with the first matched rule terminating the evaluation. Each packet is evaluated once. If no rule matches, the packet is dropped if the last rule evaluated was a permit, and passed if the last rule was a deny.

IPFilterRule filters MUST follow the format:

action dir proto from src to dst [options]

action        permit - Allow packets that match the rule.  
              deny    - Drop packets that match the rule.

dir            "in" is from the terminal, "out" is to the terminal.

proto         An IP protocol specified by number. The "ip" keyword means any protocol will match.

src and dst   <address/mask> [ports]

The <address/mask> may be specified as:

`ipno`            An IPv4 or IPv6 number in dotted-quad or canonical IPv6 form. Only this exact IP number will match the rule.

`ipno/bits`    An IP number as above with a mask width of the form 1.2.3.4/24. In this case, all IP numbers from 1.2.3.0 to 1.2.3.255 will match. The bit width **MUST** be valid for the IP version and the IP number **MUST NOT** have bits set beyond the mask.

For a match to occur, the same IP version must be present in the packet that was used in describing the IP address. To test for a particular IP version, the bits part can be set to zero. The keyword "any" is 0.0.0.0/0 or the IPv6 equivalent. The keyword "assigned" is the address or set of addresses assigned to the terminal. For IPv4, a typical first rule is often "deny in ip! assigned"

The sense of the match can be inverted by preceding an address with the not modifier (!), causing all other addresses to be matched instead. This does not affect the selection of port numbers.

With the TCP, UDP and SCTP protocols, optional ports may be specified as:

`{port|port-port}[,ports[,...]]`

The '-' notation specifies a range of ports (including boundaries).

Fragmented packets that have a non-zero offset (i.e., not the first fragment) will never match a rule that has one or more port specifications. See the frag option for details on matching fragmented packets.



options:

frag      Match if the packet is a fragment and this is not the first fragment of the datagram. frag may not be used in conjunction with either tcpflags or TCP/UDP port specifications.

ipoptions spec

Match if the IP header contains the comma separated list of options specified in spec. The supported IP options are:

ssrr (strict source route), lsrr (loose source route), rr (record packet route) and ts (timestamp). The absence of a particular option may be denoted with a '!'.

tcptoptions spec

Match if the TCP header contains the comma separated list of options specified in spec. The supported TCP options are:

mss (maximum segment size), window (tcp window advertisement), sack (selective ack), ts (rfc1323 timestamp) and cc (rfc1644 t/tcp connection count). The absence of a particular option may be denoted with a '!'.

established

TCP packets only. Match packets that have the RST or ACK bits set.

setup     TCP packets only. Match packets that have the SYN bit set but no ACK bit.

tcpflags spec

TCP packets only. Match if the TCP header contains the comma separated list of flags specified in spec. The supported TCP flags are:

fin, syn, rst, psh, ack and urg. The absence of a particular flag may be denoted with a '!'. A rule that contains a tcpflags specification can never match a fragmented packet that has a non-zero offset. See the frag option for details on matching fragmented packets.

#### icmptypes types

ICMP packets only. Match if the ICMP type is in the list types. The list may be specified as any combination of ranges or individual types separated by commas. Both the numeric values and the symbolic values listed below can be used. The supported ICMP types are:

echo reply (0), destination unreachable (3), source quench (4), redirect (5), echo request (8), router advertisement (9), router solicitation (10), time-to-live exceeded (11), IP header bad (12), timestamp request (13), timestamp reply (14), information request (15), information reply (16), address mask request (17) and address mask reply (18).

There is one kind of packet that the access device **MUST** always discard, that is an IP fragment with a fragment offset of one. This is a valid packet, but it only has one use, to try to circumvent firewalls.

An access device that is unable to interpret or apply a deny rule **MUST** terminate the session. An access device that is unable to interpret or apply a permit rule **MAY** apply a more restrictive rule. An access device **MAY** apply deny rules of its own before the supplied rules, for example to protect the access device owner's infrastructure.

The rule syntax is a modified subset of ipfw(8) from FreeBSD, and the ipfw.c code may provide a useful base for implementations.

#### Contributors

The following identifies the key contributors who provided the primary content for this document in the form of individual documents for each protocol:

##### RSIP:

Jim Renkel

##### SNMP:

Juergen Quittek  
NEC Europe Ltd.  
EMail: quittek@ccrle.nec.de

David Harrington  
Co-chair SNMPv3 WG  
EMail: dbh@enterasys.com

Megaco:

Sanjoy Sen

Cedric Aoun  
Nortel  
EMail: cedric.aoun@nortel.com

Tom Taylor  
Nortel  
EMail: taylor@nortel.com

Diameter:

Tom Taylor  
Nortel  
EMail: taylor@nortel.com

COPS:

Cedric Aoun  
Nortel  
EMail: cedric.aoun@nortel.com

Kwok-Ho Chan  
Nortel  
EMail: khchan@nortel.com

Louis-Nicolas Hamer

Reinaldo Penno  
EMail: rpenno@juniper.net

Sanjoy Sen

Author's Address

Mary Barnes  
Nortel  
2201 Lakeside Blvd.  
Richardson, TX USA

Phone: 1-972-684-5432  
EMail: mary.barnes@nortel.com

## Full Copyright Statement

Copyright (C) The Internet Society (2005).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

## Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at [ietf-ipr@ietf.org](mailto:ietf-ipr@ietf.org).

## Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet  
gement

